eman ta zabal zazu
Universidad del País Vasco
Euskal Herriko Unibertsitatea
INFORMATIKA FAKULTATEA
FACULTAD DE INFORMÁTICA

Grado en Ingeniería Informática
Computación

Trabajo de Fin de Grado

Gestión ágil de Efemérides en el entorno de Julia

Autor
AITOR IGLESIAS HERNÁNDEZ

2022

eman ta zabal zazu
Universidad del País Vasco
Euskal Herriko Unibertsitatea
INFORMATIKA FAKULTATEA
FACULTAD DE INFORMÁTICA

Grado en Ingeniería Informática
Computación

Trabajo de Fin de Grado

Gestión ágil de Efemérides en el entorno de Julia

Autor
AITOR IGLESIAS HERNÁNDEZ

Directores
JOSEBA MAKAZAGA
ANDER MURUA

Resumen

En el entorno de Julia, la comunidad JuliaAstro ofrece la posibilidad de acceder a distintas bases de datos correspondientes a las Efemérides [28] de los cuerpos celestes: JPL de la NASA, INPOP europeo... Estas bases de datos son enormes y si se pretende acceder a la información de pocos cuerpos (sol y luna) en un periodo limitado, el acceso a las enormes bases de datos puede ralentizar el proceso en el que se esté trabajando.

El objetivo de este proyecto es acceder a la información de estas bases de datos una única vez para generar los polinomios que se van a utilizar para aproximar las posiciones y velocidades de los cuerpos de interés. Una vez que se tengan los polinomios, se podrá calcular localmente tanto la posición como la velocidad de dichos cuerpos en los instantes de interés, sin tener que acceder a las bases de datos externas. Los polinomios de interpolación que se van a utilizar son los polinomios de interpolación de Chebyshev [8][10] , por lo que se necesitará la información correspondiente a los nodos de Chebyshev para poder generar los polinomios. Esta información se obtendrá con ayuda de APIs de gestión de Efemérides como pueden ser Horizons o SPICE. Además se realiza una gestión ágil de ficheros con la información de los polinomios que se han generado, para de esta manera no tener que acceder a estas APIs más que una única vez.

Tabla de contenidos

Índice de figuras

Introducción

Este proyecto tiene como finalidad crear un paquete de gestión ágil de Efemérides [28]. Las Efemérides son una tablas con diferente información sobre cuerpos celestes, esta información puede ser sus coordenadas, sus velocidades, o incluso funciones matemáticas que sirven para calcular datos sobre estos cuerpos. En el caso de este proyecto, en estas tablas se guardarán polinomios de Chebyshev con la información de las posiciones y las velocidades de los cuerpos celestes para el intervalo de tiempo que nos interese.

Estos polinomios se pueden obtener mediante interpolación polinómica [8][10] en los nodos de Chebyshev [22]. Para ello se necesita acceder a las posiciones y velocidades de los cuerpos de interés en los nodos de Chebyshev. Esa información se obtendrá de las bases de datos externas, y con la información externa en los nodos se generarán los polinomios interpoladores para cada intervalo. El objetivo final es poder utilizar esos polinomios para calcular posiciones y velocidades en cualquier punto interno del intervalo que abarca el polinomio. En función del intervalo temporal de interés, harán falta más o menos polinomios. Ya que la longitud del intervalo temporal de cada polinomio y el grado del polinomio determinará el error numérico cometido en la aproximación.

Los nodos de Chebyshev dentro de un intervalo, no están uniformemente distribuidos; cuanto más cerca están de los extremos del intervalo más juntos están. Los polinomios obtenidos mediante la interpolación polinómica en nodos uniformemente distribuidos suelen cometer un error mayor en los extremos del intervalo al que pertenecen. Los polinomios de interpolación basados en nodos de Chebyshev no cometen un error tan grande en los extremos, de hecho, con estos polinomios se puede aproximar con un error muy pequeño los valores de las coordenadas de las posiciones y de las velocidades para cualquier momento del intervalo de tiempo que abarca el polinomio. Es por esto que se guardan estos polinomios en unas tablas, para más tarde poder calcular las coordenadas de posición y velocidad de manera ágil.

Los polinomios interpoladores tienen la siguiente forma:

$$P(x) = c_0 + \sum_{i=1}^{d} c_i T_i(x)$$

donde $c_0, c_1, \ldots, c_d$ son los coeficientes que determinan el polinomio y dependen de los valores en los nodos de Chebyshev y T_i son los polinomios de Chebyshev. Se trata de unos polinomios ortogonales de grado i que se pueden calcular de forma recursiva.

Los coeficientes de estos polinomios se guardarán en las tablas, es decir, $c_0, c_1, \ldots, c_d$. Estas tablas no contendrán un único polinomio, sino que pueden contener polinomios de distintos intervalos de tiempo y cuerpos. Es por ello que estas tablas siempre tendrán el mismo formato. Cada linea contendrá ocho coeficientes de un polinomio, la tabla estará dividida en diferentes secciones, cada una de estas siendo los polinomios de un cuerpo, y cada una de estas secciones estará dividida en intervalos de tiempo, donde se guardarán los coeficientes de los polinomios que nos permitirán aproximar las coordenadas de las posiciones y de las velocidades. Adicionalmente para poder interpretar estas tablas se generará un fichero con información para cada una de las tablas. En este fichero se indicará el orden en el que están los cuerpos, el número de polinomios que contiene, el número de coeficientes que tiene cada intervalo y los intervalos de tiempo que abarcan estos polinomios.

En este trabajo se ha desarrollado un paquete para usuarios que no necesiten bases de datos de Efemérides tan grandes como las que encontramos hoy en día. Usuarios que necesitan una base de datos más específica, con menos cuerpos y para un intervalo de tiempo más reducido. El paquete creado a lo largo de este proyecto permite generar estas bases de datos y gestionarlas.

En esta memoria se pueden encontrar seis capítulos además de esta introducción que contiene una pequeña explicación de los objetivos y aclaraciones de conceptos más o menos técnicos.

- El primer capítulo está dedicado exclusivamente a la gestión del proyecto. En este capítulo se pueden ver tanto la planificación del proyecto como el seguimiento del mismo.
- El segundo capítulo trata sobre las diferentes APIs de gestión de Efemérides en Julia y cómo podemos hacer uso de ellas.
- El tercer capítulo trata sobre los ficheros de Efemérides, la información que contienen y la manera en la que guardan esta información.
- El cuarto capítulo explica los diferentes métodos utilizados en este proyecto para obtener los polinomios de Chebyshev. En este capítulo además se muestra el error relativo cometido al aproximar las posiciones y las velocidades con los polinomios que se han obtenido. Para ello se comparan los valores interpolados con los valores que nos dan las bases de datos externas.
- El quinto capítulo trata sobre el paquete de gestión de Efemérides creado en este proyecto. Habla sobre los diferentes ficheros del paquete así como sus funcionalidades.
- El sexto y último capítulo está reservado para las conclusiones y el trabajo a futuro.

Efemérides

En el estudio de los cuerpos celestes, las Efemérides [28] son una tabla de valores que da las posiciones de los objetos astronómicos en el universo en un momento o en algunos

momentos dados. Una efemérides planetaria moderna constituye un software que genera las posiciones de los planetas o satélites, o de asteroides o cometas o cuerpos celestes en general en cualquier momento virtualmente deseado por el usuario. Existen diferentes servidores que ofrecen la posibilidad de obtener la posición y la velocidad de los cuerpos celestes a lo largo del tiempo (puede ser pasado o actual o incluso futuro). Estos servidores tienen toda la información de todos los cuerpos celestes en grandes bases de datos y permiten el acceso a dicha información mediante alguna API. Las APIs que se utilizarán en este proyecto son SPICE, JPLEphemeris y Horizons.

Diferentes formatos de tiempo

A lo largo del proyecto se mencionan varias veces los formatos de tiempo, además, en el código se hacen transformaciones entre estos. Son muchos los formatos en los que se puede representar la fecha y hora, sin embargo estos son los de interés para este proyecto.

- UTC (Coordinated Universal Time) [21], [25]: El Tiempo Universal Coordinado es el principal estándar de tiempo por el cual el mundo regula los relojes y el tiempo. El formato de este es el siguiente: Año-Mes-Día T Horas:Minutos:Segundos
- ET (Ephemeris Time) [21], [26]: Esta medida de tiempo es una forma constante y uniforme de tiempo utilizada en astronomía al hacer cálculos del movimiento orbital de objetos del sistema solar. Se representa mediante un número de coma flotante de doble precisión.
- JDTDB (Julian Date relative to TDB) [21], [29]: Este es el formato en el que se guardan los intervalos de tiempo de los coeficientes de Chebyshev en los ficheros SPK. Estos ficheros son utilizados en los softwares de gestión de efemérides y contienen información sobre distintos cuerpos celestes.

Ficheros LSK y SPK

A lo largo de la memoria se mencionan estos ficheros frecuentemente. Es por ello que es importante conocer el significado de las siglas y la utilidad de estos. Ambos ficheros son ficheros binarios para que las funciones de los distintos paquetes puedan acceder rápidamente a la información de estos. Sin embargo esto hace que las personas que no conocen la manera en la que están codificados estos ficheros no puedan acceder a la información cruda que hay dentro.

- LSK (Leapseconds Kernel) [18], [19]: Estos ficheros se utilizan para realizar conversiones entre distintos formatos de tiempo. El uso más común es transformar de ET a UTC y viceversa. En este proyecto también se realizan transformaciones de JDTDB a ET.
- SPK (Spacecraft and Planet Ephemeris Kernel) [17], [18]: Es un fichero de Efemérides, es decir, contiene información del estado de diferentes planetas. En estos ficheros hay varios segmentos, estos segmentos contienen información de un cuerpo en un intervalo pequeño de tiempo. Hay segmentos de diferentes tipos, sien-

do el más útil para este proyecto el segmento tipo 2 que contiene coeficientes de Chebyshev.

Identificación de los cuerpos celestes: La ID de cada cuerpo

En el código de este proyectos se puede ver cómo cada cuerpo tiene una ID asignada. Podemos encontrar la ID correspondiante a cada cuerpo en el siguiente link: https://ssd.jpl.nasa.gov/horizons/time_spans.html. Estas IDs siguen ciertas reglas:

- La ID 0 pertenece al baricentro del sistema solar. Cuando se habla del baricentro de un sistema se refiere al centro de masa del cuerpo principal y los cuerpos que orbitan alrededor de este.
- Las IDs entre el 1 y 8 (ambos incluidos) pertenecen al baricentro de los ocho sistemas planetarios respectivamente.
- La ID 9 pertenece al baricentro de Plutón.
- Si se quiere la ID de un cuerpo y no de su baricentro, se añade un 99 al final de la ID del cuerpo, por ejemplo la ID 399 corresponde a la Tierra. En el caso del Sol el número reservado a este cuerpo es el 10.
- Si se quiere la ID de un satélite, se coge la ID del baricentro al que pertenece y se añade el número del satélite al final, este número tiene que tener 2 dígitos puesto que hay planetas como Saturno que tienen más de 10 satélites. Por ejemplo la ID de la Luna es 301.
- En cuanto a ciertos satélites artificiales y ciertos cometas también poseen IDs propias, sin embargo en este proyecto no han sido investigados.

CAPÍTULO 1

Gestión del proyecto

Este capítulo presenta la planificación del proyecto y el seguimiento del mismo. La gestión ayuda a realizar el trabajo de manera ordenada, permitiendo reaccionar frente a cualquier imprevisto que pueda surgir a lo largo del proyecto sin tener que dedicarle más tiempo del necesario.

1.1. Alcance

Actualmente se pueden encontrar diferentes APIs que permitan conseguir las coordenadas y las velocidades de diferentes cuerpos celestes. Sin embargo, estas APIs hacen uso de bases de datos gigantes. El objetivo de este proyecto es poder conseguir las coordenadas y las velocidades de cualquier cuerpo sin la necesidad de acceder a esas enormes bases de datos.

1.2. Objetivos del proyecto

1.2.1. Objetivo general

El objetivo de este proyecto es desarrollar un software que permita calcular de manera ágil las coordenadas y las velocidades de ciertos cuerpos celestes específicos. El software calculará estos valores mediante polinomios interpoladores. También se crearán ficheros para almacenar localmente estos polinomios, de esta manera no habrá que crearlos ni tampoco habrá que acceder a los servidores externos cada vez que se quiera conocer las coordenadas o la velocidad de uno de los cuerpos disponibles.

1.2.2. Objetivos específicos

- Obtener las coordenadas y la velocidad de cualquier cuerpo mediante un paquete de Gestión de Efemérides.
- Calcular de manera correcta la posición y la velocidad de cualquier cuerpo celeste.

- Comparar distintos métodos para calcular la posición y la velocidad de un cuerpo celeste.
- Crear un paquete de gestión de Efemérides.

1.3. Requisitos

- El error relativo al calcular la posición y velocidad de un cuerpo celeste respecto a otros paquetes de gestión de Efemérides será de alrededor de 10^{-14}.
- El paquete será creado usando el lenguaje de programación Julia.
- El código fuente de la implementación y el manual estarán disponible a través de un repositorio GitHub.
- El paquete es responsable de verificar los datos que se le especifican para evitar errores de ejecución.
- El paquete estará disponible dentro del repositorio de paquetes de Julia.

1.4. Estructura de descomposición del trabajo

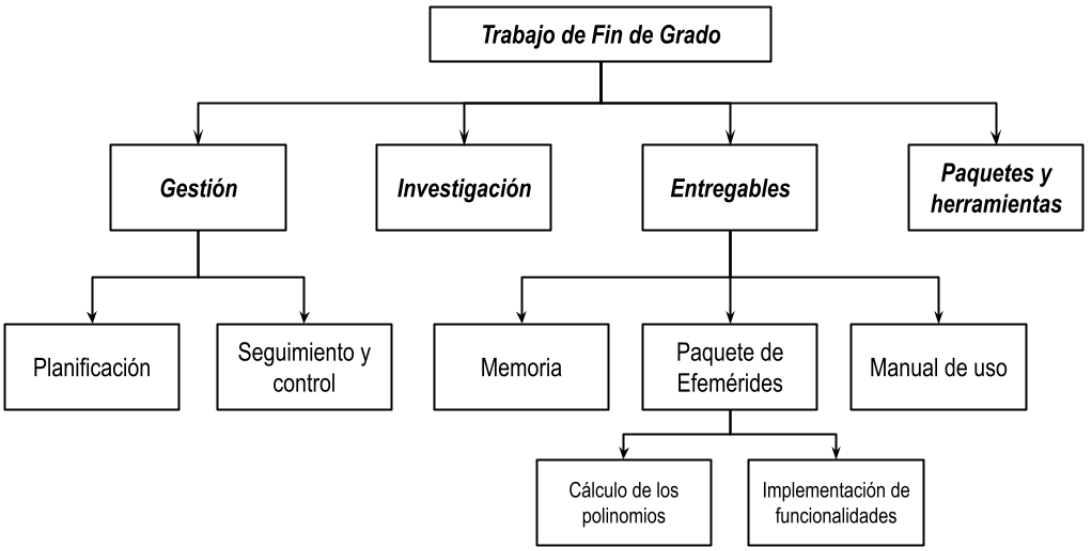

Figura 1.1: Estructura de Descomposición del Trabajo

1.4.1. Paquete de trabajo (PT1): Gestión

El paquete de trabajo **Gestión** se divide en dos partes

- **Planificación:** Este paquete de trabajo contiene las tareas relacionadas a la planificación del proyecto. Tiempo estimado: 25 horas.

- **Seguimiento y control:** Este paquete de trabajo contiene las tareas relacionadas con el seguimiento del proyecto. Tiempo estimado: 20 horas.

1.4.2. Paquete de trabajo (PT2): Investigación

El paquete de trabajo **Investigación** contiene las tareas relacionadas con el aprendizaje de herramientas necesarias para el proyecto y las tareas relacionadas con la recolección de información necesaria para el desarrollo del proyecto. Tiempo estimado: 50 horas.

1.4.3. Paquete de trabajo (PT3): Entregables

El paquete de trabajo **Entregables** contiene las tareas relacionadas al desarrollo de los entregables principales del proyecto. Este paquete se divide en tres partes:

- **Memoria:** Este paquete de trabajo contiene las tareas necesarias para el desarrollo de la memoria del proyecto. Tiempo estimado: 50 horas.
- **Paquete de Efemérides:** Este paquete de trabajo contiene las tareas relacionadas al desarrollo del paquete de gestión de Efemérides planteado en el alcance del proyecto. Podemos dividirlo en dos partes:
  - **Cálculo de los polinomios:** Este paquete de trabajo contiene las tareas relacionadas al cálculo de los polinomios interpoladores así como la comparación con los resultados obtenidos mediante a distintos paquetes de gestión de Efemérides. Tiempo estimado: 55 horas.
  - **Implementación de funcionalidades**: Este paquete de trabajo contiene las tareas relacionadas a la implementación de las funcionalidades del paquete de gestión de Efemérides planteado en el alcance del proyecto. Tiempo estimado: 75 horas.
- **Manual de uso:** Este paquete de trabajo contiene las tareas necesarias para la realización del manual de uso del paquete de gestión de Efemérides planteado en el alcance del proyecto. Tiempo estimado: 15 horas.

1.4.4. Paquete de trabajo (PT4): Paquetes y herramientas

El paquete de trabajo **Paquetes y herramientas** contiene las tareas relacionadas a la instalación de paquetes y herramientas necesarias para la realización de cualquier entregable del proyecto. También contiene la tarea de añadir el paquete de gestión de Efemérides al repositorio de paquetes de Julia. Tiempo estimado: 10 horas.

1.5. Tareas a realizar

1.5.1. Gestión

Planificación:

1. Creación de la planificación.

Seguimiento y control:

1. Reuniones con los tutores.
2. Recogida del tiempo utilizado para el desarrollo de cada tarea.

1.5.2. Investigación

1. Investigación sobre paquetes de gestión de Efemérides.
2. Profundización sobre técnicas de interpolación.
3. Investigación sobre gestión de ficheros en Julia.
4. Investigación sobre la creación de paquetes instalables en Julia.

1.5.3. Entregables

Memoria:

1. Diseño de la estructura.
2. Redacción del resumen.
3. Redacción de la introducción.
4. Redacción del contenido.
5. Redacción de las conclusiones.
6. Redacción de las referencias.

Paquete de Efemérides

- **Cálculo de los polinomios:**
    1. Búsqueda de los distintos softwares de gestión de Efemérides.
    2. Obtención de las coordenadas y velocidades de los cuerpos.
    3. Decisión entre los distintos softwares de gestión de Efemérides.
    4. Cálculo de los polinomios mediante la resolución de sistema de ecuaciones.
    5. Cálculo de los polinomios mediante la Transformada Discreta de los Cosenos.
    6. Evaluación de los polinomios.
- **Implementación de funcionalidades:**
    1. Decisión de formato de los ficheros de coeficientes.
    2. Creación de software para almacenar los polinomios de varios cuerpos.
    3. Creación de software para evaluar polinomios específicos de los ficheros.

4. Creación de software para crear un fichero de polinomios usando como base otro fichero de polinomios.
5. Creación de estructura para almacenar los coeficientes de un cuerpo específico.
6. Creación de software para evaluar polinomios específicos de la estructura.

Manual de uso:

1. Redacción de la documentación del paquete de gestión de Efemérides.
2. Redacción del manual.
3. Generación de los fichero HTML, CSS, JavaScript, etc. del manual de uso.
4. Alojamiento del manual en una página de Github.

1.5.4. Paquetes y herramientas

1. Instalación de las herramientas necesarias para el desarrollo del proyecto.
2. Instalación de los paquetes necesarios para el desarrollo del proyecto.
3. Registro del paquete de gestión de Efemérides en el repositorio de paquetes de Julia.

1.6. Diagrama de Gantt

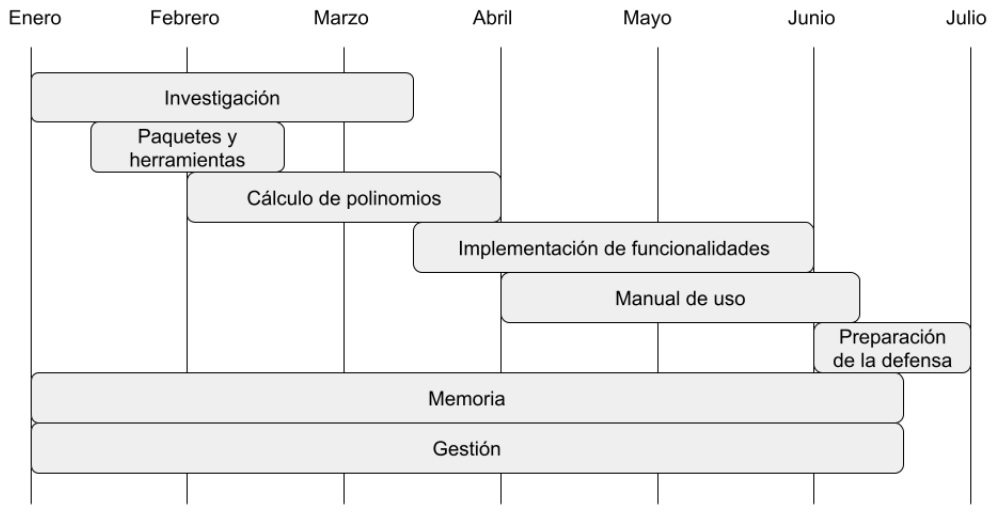

Figura 1.2: Diagrama de Gantt del proyecto simplificado

1.7. Análisis de riesgos

Descripción	Probabilidad	Impacto	Respuesta
Pérdida de información	Baja	Alto	Realizar copias de seguridad en la nube de manera constante.
Algún paquete puede no funcionar o puede haber incompatibilidad entre los paquetes	Baja	Medio	Se busca otro paquete o paquetes similares.
Ser incapaz de entender los conceptos matemáticos requeridos para el desarrollo del proyecto	Media	Alto	Solicitar una tutoría con los directores.

1.8. Desviación de tiempo

	Estimación (h)	Real (h)
GESTIÓN		
Planificación	25	22
Seguimiento y control	20	19
INVESTIGACIÓN		
Investigación	50	54
ENTREGABLES		
Memoria	50	58
Cálculo de los polinomios	55	56.5
Implementación de las funcionalidades	75	68
Manual de uso	15	17
PAQUETES Y HERRAMIENTAS		
Paquetes y herramientas	10	8.5
Total	**300**	**303**

En la tabla de desviación de tiempo se puede observar varias pequeñas desviaciones, esto es completamente normal pues en un proyecto tan grande no es posible atinar con exactitud aun así se explicarán las desviaciones de más de tres horas, las cuales son:

- Investigación: La desviación de tiempo se debe que se tuvieron ciertos problemas al inicio del proyecto con el uso de paquetes de gestión de Efemérides. En concreto el paquete de Julia JPLEphemeris.jl no funciona correctamente, sin embargo esto no está indicado en ningún lugar por lo que se perdió mucho tiempo tratando de hacer funcionar este.
- Memoria: La desviación de tiempo se debe a una mala asignación de tiempo, esto debido a la falta de experiencia para realizar una estimación más adecuada puesto que es la primera vez que realizo una memoria tan grande.

- Implementación de las funcionalidades: La desviación de tiempo se debe a que resulto ser una tarea más sencilla de lo pensado al inicio del proyecto.

1.9. Herramientas utilizadas

A lo largo del proyecto se han utilizado diferentes herramientas con diferentes fines, en esta sección se nombrarán estas herramientas además de explicar para qué han sido utilizadas en este proyecto.

- **Dropbox**: Dropbox es una herramienta de alojamiento de archivos multiplataforma en la nube. En este proyecto además de ser utilizado para compartir los archivos entre alumno y director, se ha utilizado para guardar copias de seguridad de los diferentes ficheros.
- **Julia**: Julia es un lenguaje de programación diseñado principalmente para el análisis numérico y la computación científica. En este proyecto ha sido utilizado como único lenguaje de programación. En este lenguaje están escritos los notebooks y el paquete generado en este proyecto. Además, todos los gráficos que se pueden encontrar en esta memoria han sido generados con Julia (a excepción del diagrama de Gantt).
- **GitHub**: GitHub es un servicio de alojamiento para el desarrollo de software y el control de versiones. El paquete generado en este proyecto está alojado en GitHub. Sin embargo, el repositorio en el que se encuentra este paquete no fue creado hasta la creación del paquete, es por eso que no coincide con la fecha en la que se inició el proyecto. También se ha utilizado la herramienta de GitHub, GitHub Pages para alojar la página web con la documentación del paquete.
- **Jupyter Lab**: Jupyter Lab es una herramienta web para el desarrollo de notebooks. Todos los notebooks de este proyecto han sido desarrollados con ayuda de esta herramienta.
- **Overleaf**: Overleaf es un editor colaborativo de LaTeX basado en la nube que se utiliza para escribir, editar y publicar documentos científicos. Toda la memoria ha sido escrita con esta herramienta.
- **Visual Studio Code**: Visual Studio Code es un editor de código fuente. Este editor ha sido utilizado para la creación del paquete desarrollado en este proyecto.
- **Paint3D**: Paint3D es un programa editor de imágenes y modelos 3D. En este proyecto ha sido utilizado para la creación de algunas de las figuras que se encuentran en este documento.

CAPÍTULO 2

Acceso a las bases de datos de Efemérides

Como se ha mencionado anteriormente, uno de los objetivos del proyecto es generar polinomios que sirvan para calcular las posiciones y las velocidades de diferentes cuerpos celestes. Pero para generar estos polinomios primero es necesario conocer los valores de las coordenadas y las velocidades del cuerpo en unos instantes de tiempo determinados. Para obtener estos datos se hará uso de diferentes APIs.

2.1. Búsqueda de los distintos softwares de gestión de Efemérides

Para realizar el proceso de cálculo de los polinomios interpoladores primero se necesita conocer el estado de los planetas en ciertos instantes específicos. Para ello se compararán tres paquetes de gestión de Efemérides en Julia.

2.1.1. JPLEphemeris.jl

Las Efemérides de desarrollo del JPL son el resultado de simulaciones del Sistema Solar utilizadas para la navegación de naves espaciales y con fines astronómicos. Se publican como archivos del núcleo SPK que contienen conjuntos de coeficientes polinómicos de Chebyshev con los que la posición y la velocidad de los planetas del Sistema Solar se pueden interpolar con alta precisión para todas las fechas cubiertas por las Efemérides.

2.1.2. Horizons.jl

El sistema de Efemérides en línea JPL Horizons brinda acceso a datos del sistema solar y producción personalizable de Efemérides precisas para observadores, planificadores de misiones, investigadores y el público, al caracterizar numéricamente la ubicación, el movimiento y la observabilidad de los objetos del sistema solar como una función del tiempo, visto desde lugares dentro del Sistema Solar.

2.1.3. SPICE.jl

SPICE.jl es un paquete de Julia para el conjunto de herramientas de SPICE proporcionado por la instalación de información auxiliar y de navegación (NAIF) de la NASA. Proporciona funcionalidad para leer archivos de datos SPICE y calcular la geometría de observación derivada, como la altitud, la latitud/longitud y los ángulos de iluminación.

2.2. Obtención de las coordenadas y velocidades de los cuerpos

Se utilizan los paquetes mencionados en la sección anterior para obtener el estado de los diferentes cuerpos. No hay mucho que mencionar en esta sección ya que solo hay que utilizar métodos de los respectivos paquetes. Sin embargo, el paquete JPLEphemeris.jl está descontinuado y tiene varios errores por lo que no es posible conseguir el estado de ningún cuerpo.

2.3. Decisión entre los distintos softwares de gestión de Efemérides

Sabiendo que el paquete JPLEphemeris.jl ya no es una opción solo hay que decidir entre los paquetes Horizons.jl y SPICE.jl. Hay bastantes diferencias entre estos dos paquetes, sin embargo, para este proyecto estas son las más importantes:

- Horizons permite conseguir el estado de los cuerpos en varios instantes de tiempo con una única llamada a una función, mientras que Spice solo permite obtener el estado en un instante de tiempo. No obstante, los instantes de tiempo en los que Horizons proporciona los datos de los cuerpos, están uniformemente distribuidos, esto lo convierte en una desventaja ya que para este proyecto se necesita el estado de un cuerpo en los nodos de Chebyshev.
- Spice tiene un método que permite obtener los coeficientes de un fichero SPK.

Por estos dos motivos para este proyecto se ha decidido usar el paquete SPICE.jl. Por desgracia, el método de obtención de los coeficientes únicamente está disponible en el paquete C_SPICE, que es el paquete de C en el que está basado SPICE.jl.

CAPÍTULO 3

Polinomios en los ficheros SPK

Como se ha mencionado anteriormente los ficheros SPK son ficheros de Efemérides. Estos ficheros están formados por segmentos SPK. Cada uno de estos segmentos esta formado por dos partes, la cabecera del segmento y los datos del segmento.

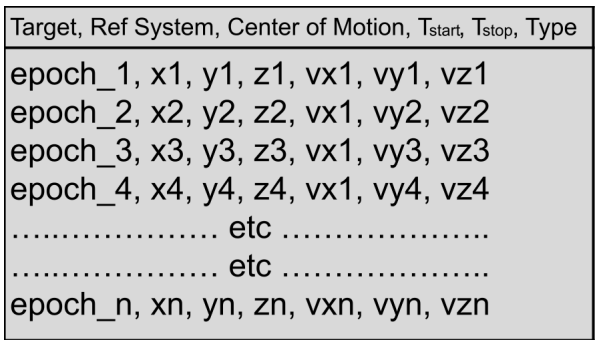

Figura 3.1: Segmentos SPK

En la cabecera se encuentra la información acerca del segmento: esta información contiene el cuerpo al que corresponde la información, el sistema de referencia, el centro del sistema de referencia el intervalo de tiempo y el tipo del segmento. El tipo del segmento indica los datos que contiene el segmento, estos datos pueden ser las coordenadas, o pueden ser los coeficientes de los polinomios de Chebyshev entre otros.

Los datos de los segmentos varían según el tipo de segmento. El segmento de interés para este proyecto es el segmento tipo 2 que contiene los coeficientes de los polinomios de Chebyshev. Cada epoch dentro de este segmento no se corresponde a un polinomio sino un conjunto de polinomios, el tamaño del conjunto y el grado de los polinomios varía dependiendo del cuerpo. Uno de los objetivos de este proyecto es calcular estos polinomios ya sea resolviendo un sistema lineal o utilizando la Transformada Discreta de los Cosenos [2][7]. Sin embargo, para esto es necesario conocer el intervalo al que pertenece cada uno de los polinomios y el grado de estos.

Ningún paquete de gestión de Efemérides mencionado anteriormente permite conocer los polinomios ni tampoco permite conocer a qué intervalo pertenecen estos. Tampoco se puede sacar la información de los propios ficheros SPK puesto que son ficheros binarios y no sabemos cómo están codificados. Afortunadamente, en el servidor FTP de la NASA `ftp://ssd.jpl.nasa.gov/pub/`, se puede encontrar un fichero de texto con la información correspondiente al fichero SPK usado en este proyecto.

De este fichero se puede obtener la siguiente información de interés:

- El intervalo de tiempo que abarca el fichero SPK en formato JDTDB o UTC.
- El número de días que abarca cada epoch.
- El número de polinomios en cada conjunto de cada cuerpo.
- El grado de los polinomios de cada cuerpo.

Con esa información se pueden calcular los polinomios mediante el proceso de interpolación polinómica en los nodos de Chebyshev, utilizando el siguiente sistema de ecuaciones:

$$P = T \cdot C \tag{3.1}$$

donde:

- P es un vector con la coordenada deseada en distintos instantes de tiempo. En nuestro caso, estos instantes de tiempo serán los nodos de Chebyshev del intervalo de tiempo que se desea interpolar.
- C es un vector con los coeficientes que deseamos guardar en las tablas.
- T es una matriz con los polinomios de Chebyshev evaluados en los mismos instantes de tiempo utilizados para obtener P.

Los polinomios interpoladores tienen la siguiente forma:

$$P(t) = c_0 + \sum_{i=1}^{d} c_i T_i(x) \quad donde: \quad x = 2 \cdot \frac{t - \frac{t_{j-1}+t_j}{2}}{t_j - t_{j-1}} \tag{3.2}$$

$c_0, c_1, \ldots, c_d$ son los coeficientes que determinan el polinomio y T_i son los polinomios. Se trata de unos polinomios ortogonales de grado i que se pueden calcular de forma recursiva.

Sabiendo esto, se definen los polinomios interpoladores de Chebyshev de la siguiente manera:

$$P_j(t_k) = c_0 + \sum_{i=1}^{d} c_i T_i(x_k) \tag{3.3}$$

done j indica la ecuación a que coordenada pertenece, las tres primeras pertenecen a las coordenadas de posición y las tres siguientes a las coordenadas de velocidad. $P_j(t_k)$ es el

valor de la coordenada en el instante t_k. d es el grado del polinomio que vamos a guardar y $c_0, c_1, \ldots, c_i$ son los coeficientes del mismo. $T_i(x_k)$ es el polinomio de Chebyshev evaluado en el instante x_k. Los polinomios de Chebyshev se evalúan para valores de x entre -1 y 1, pero el instante t_k en el que se tiene la información del cuerpo celeste corresponde a un momento del espacio temporal que abarca el polinomio interpolador, por lo que para cada momento temporal hay que calcular el valor de la variable donde se definen los polinomios de Chebyshev, dicha relación se puede ver en la ecuación (4.3).

El polinomio de Chebyshev es definido mediante la siguiente expresión de recurrencia:

$$\begin{aligned} T_0(x) &= 1 \\ T_1(x) &= x \\ T_{n+1}(x) &= 2xT_n(x) - T_{n-1}(x) \end{aligned} \tag{3.4}$$

Este polinomio es evaluado en el instante x_k en vez de en el instante t_k ya que t_k pertenece al intervalo que interpola el polinomio mientras que x_k pertenece al intervalo $[-1, 1]$.

Una vez se obtengan los polinomios estos se guardarán en un formato similar a los ficheros SPK. En vez de guardar los polinomios junto a la información de estos en un único fichero, se generarán dos ficheros: Un JSON con la información del cuerpo o de los cuerpos y un CSV con los polinomios del cuerpo o de los cuerpos.

CAPÍTULO 4

Cálculo de los polinomios interpoladores

Es posible que el usuario del paquete de gestión de Efemérides este interesado en obtener los polinomios de diferentes cuerpos que abarcan un gran intervalo de tiempo, por ello, es importante poder generarlos de manera eficiente. En este capítulo se explican diferentes métodos para generar estos coeficientes. El primero es más lento que el segundo, pero este nos permite comprender de manera más simple el proceso. Mientras que el segundo es más rápido debido a que utiliza técnicas matemáticas más avanzadas. Ambos métodos están también explicados en el nootebok Chebyshev_poly.ipynb.

4.1. Cálculo de los polinomios mediante la resolución de sistema de ecuaciones

El primer método consiste en generar los polinomios mediante la resolución de un sistema de ecuaciones. Sin embargo, antes de realizar el proceso para generar el polinomio es necesario entender cómo es el polinomio y conocer los datos que lo definen.

4.1.1. Polinomios de Chebyshev

Los polinomios de Chebyshev son definidos mediante a la siguiente relación de recurrencia:

$$\begin{aligned} T_0(x) &= 1 \\ T_1(x) &= x \\ T_{n+1}(x) &= 2xT_n(x) - T_{n-1}(x) \end{aligned} \tag{4.1}$$

La siguiente gráfica muestra los cinco primeros polinomios de Chebyshev en el intervalo $[-1,1]$.

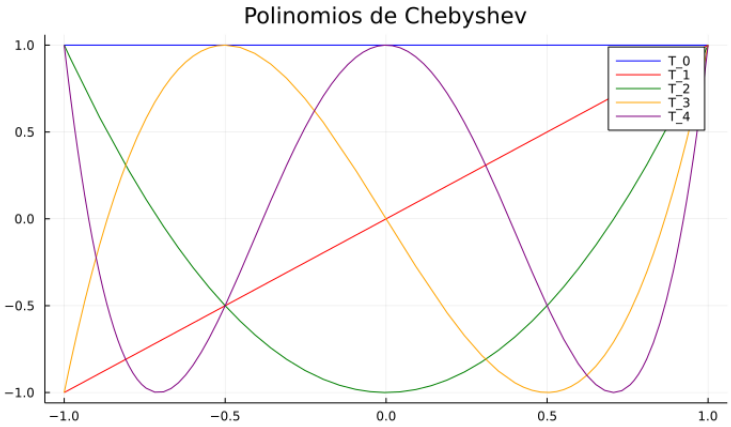

Figura 4.1: Polinomios de Chebyshev en el intervalo $[-1,1]$

Utilizaremos estos polinomios para la interpolación polinómica, además la interpolación no se realiza en nodos uniformemente distribuidos sino que se utilizan nodos de Chebyshev.

4.1.2. Nodos de Chebyshev

Los nodos de Chebyshev en el intervalo $[-1,1]$ se pueden definir mediante la siguiente función:

$$x_k = \cos\left(\frac{2k-1}{2n}\pi\right), k = 1,2,\ldots,n \tag{4.2}$$

Gráficamente se vería así:

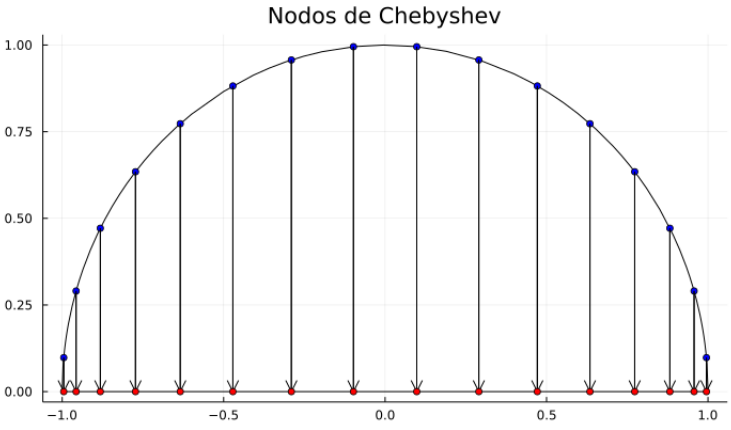

Figura 4.2: Raíces del polinomio de Chebyshev

Utilizando está distribución se minimiza el error que se suele cometer en la interpolación polinómica en los extremos.

No obstante, estas raíces pertenecen al intervalo $[-1,1]$ y para este proyecto es necesario obtener las raíces entre diferentes intervalos de tiempo, para la explicación matemática se usará el intervalo $[t_{j-1},t_j]$. Puesto que los nodos de Chebyshev pertenecen al intervalo $[-1,1]$ es necesario hacer un cambio de variable, para transformar cualquier valor entre -1 y 1 al valor equivalente entre t_{j-1} y t_j. De esta manera obtendremos los nodos de Chebyshev en el intervalo $[t_{j-1},t_j]$. Para ello tenemos la siguiente formula, se trata de un escalado y una traslación, es decir, el intervalo que $[t_j,t_{j-1}]$ hay que escalarlo a un intervalo de longitud 2 (intervalo que va de -1 a 1, es decir intervalo de longitud $1-(-1)=2$) y el valor en vez de ir de 0 a 2 va de -1 a 1 (traslación de -1)

$$x = 2 \cdot \frac{t - \frac{t_{j-1}+t_j}{2}}{t_j - t_{j-1}} \tag{4.3}$$

Esta formula te permite obtener el valor de x que le corresponde al instante t donde el usuario del paquete pretende obtener las posiciones y/o velocidades (haciendo uso de los polinomios interpoladores). También es necesaria la relación inversa, ya que para generar los polinomios se necesitan los valores en los nodos, por lo que para cada nodo x_k se puede obtener el instante t_k que le corresponde al nodo y después haciendo uso de las APIs de gestión de Efemérides se obtendrán los valores en los nuevos nodos. Para conseguir la relación inversa se despeja la t de la formula anterior y se obtiene la siguiente expresión.

$$t = \frac{x(t_j - t_{j-1}) + t_{j-1} + t_j}{2} \tag{4.4}$$

4.1.3. Sistema de ecuaciones

Se realizará interpolación polinómica para conseguir un polinomio de grado d, como este polinomio va a ser obtenido mediante a la resolución de un sistema de ecuaciones son necesarios $d+1$ nodos de Chebyshev. Estos nodos son fácilmente obtenibles con la formula mencionada anteriormente (4.2). Después, se realiza un cambio de variable para que los nodos pertenezcan al intervalo temporal deseado (el que va de t_{j-1} a t_j y a partir de estos nodos se obtiene el vector con los valores de la posición deseada, esto se puede conseguir mediante una API de gestión de Efemérides.

El siguiente paso es evaluar $d+1$ polinomios de Chebyshev en los nodos originales. Esto, más adelante se hará con el algoritmo de Clenshaw, sin embargo, ya que este es el método manual, la evaluación se hará con la definición del polinomio (4.1). Tras la evaluación se obtendrá una matriz en la que cada fila son los primeros $d+1$ polinomios de Chebyshev evaluados en el nodo correspondiente a dicha fila.

Una vez conseguidos estos datos, se genera el siguiente sistema de ecuaciones:

$$\begin{pmatrix} p_0 \\ p_1 \\ \vdots \\ p_d \end{pmatrix} = \begin{pmatrix} t_{0,0} & t_{0,1} & \cdots & t_{0,d} \\ t_{1,0} & t_{1,1} & \cdots & t_{1,d} \\ \vdots & \vdots & \ddots & \vdots \\ t_{d,0} & t_{d,1} & \cdots & t_{d,d} \end{pmatrix} \begin{pmatrix} c_0 \\ c_1 \\ \vdots \\ c_d \end{pmatrix} \tag{4.5}$$

Donde

- $(p_0, p_1, \ldots p_d)^T$ es el vector de los valores que pretendemos interpolar en el intervalo que abarca el polinomio, y que corresponden a los valores en los nodos de Chebyshev.
- $(c_0, c_1 \ldots c_d)^T$ es el vector de los coeficientes que deseamos conocer y que determina el polinomio interpolador.
- La matriz (t_{ij}) es la matriz de los polinomios de Chebyshev evaluados donde $t_{ij} = T_j(x_i)$, es decir, el polinomio T_j evaluado en el nodo x_i. Se podría decir que cada fila i de la matriz corresponde a los polinomios de Chebyshev evaluados en el nodo i. De la misma forma se puede decir que cada columna j corresponde al polinomio j de Chebyshev evaluado en los diferentes nodos.

Se podría definir cada una de las ecuaciones del sistema de la siguiente manera:

$$P_j(t_k) = c_0 + \sum_{i=1}^{d} c_i T_i(x_k) \tag{4.6}$$

Donde:

- Tenemos tres polinomios para las coordenadas de la posición y otros tres polinomios para las coordenadas de la velocidad, además tenemos el intervalo de tiempo que nos interesa dividido en subintervalos que nos permiten aproximar los valores con la precisión requerida.
- $P_j(t_k)$ es la función que nos da la coordenada solicitada en el ínstate t_k siendo k el índice del nodo de Chebyshev.
- d es el grado del polinomio que queremos conseguir y $c_0, c_1, \ldots, c_d$ sus coeficientes.
- T_i es el i-esimo polinomio de Chebyshev.
- x_k es el k-esimo nodo de Chebyshev.

Una vez planteado el sistema de ecuaciones únicamente hace falta resolverlo para la obtención de los coeficientes del polinomio interpolador.

4.2. Cálculo de los polinomios mediante a la Transformada Discreta de los Cosenos

El segundo método consiste en generar los polinomios mediante la Transformada Discreta de los Cosenos (DCT) [2][7]. Para ello se hará uso del paquete de Julia FFTW.jl, donde

se puede calcular de manera optimizada la DCT[3]. Es por ello que este método es más rápido que el primero.

4.2.1. Transformada Discreta de los Cosenos

La transformada discreta del coseno $F(k)$ es una variación de la transformada discreta de Fourier donde la imagen se descompone en sumas de cosenos.

Se puede definir la Transformada Discreta de los Cosenos como:

$$F(k) = c(k) \sum_{j=0}^{N-1} f(j) \cos\left(\frac{(2j+1)k\pi}{2N}\right) \tag{4.7}$$

donde:

$$c(k) = \begin{cases} \sqrt{\frac{1}{N}} & si \quad k = 0 \\ \sqrt{\frac{2}{N}} & si \quad k = 1,2,\ldots,N-1 \end{cases} \tag{4.8}$$

4.2.2. Proceso de obtención de los polinomios

El proceso es bastante similar al anterior con unos pocos cambios. Los nodos serán generados de la misma manera, sin embargo la parte de transformar los nodos para que pasen de estar en el intervalo $[-1,1]$ al intervalo deseado es incorporada a la función que genera los nodos. En este método también se necesitan los valores en los nodos, el proceso es el mismo.

Para calcular los coeficientes se usa la función DCT (Transformada Discreta de Cosenos) que dados los valores de los nodos devuelve los coeficientes del polinomio interpolador.

Para finalizar se utiliza el algoritmo de Clenshaw [1] para evaluar el polinomio. El algoritmo de Clenshaw es una generalización del algoritmo de Horner y es usado para evaluar funciones que puedan definirse mediante una recurrencia de tres términos de manera eficiente.

El algoritmo de Clenshaw realiza la suma ponderada de una serie finita de funciones $\phi_k(x)$:

$$S(x) = \sum_{k=0}^{n} a_k \phi_k(x) \tag{4.9}$$

donde $\phi_k(x)$ para $k = 0,1,\ldots$ es una secuencia de funciones que satisfacen la relación de recurrencia lineal

$$\phi_{k+1}(x) = \alpha_k(x)\phi_k(x) + \beta_k(x)\phi_{k-1}(x), \tag{4.10}$$

Se puede realizar la suma de una serie de n coeficientes mediante a la formula de recurrencia inversa:

$$\begin{aligned} b_{n+1} &= b_{n+2} = 0 \\ b_k(x) &= a_k + \alpha_k(x)b_{k+1}(x) + \beta_{k+1}(x)b_{k+2}(x) \end{aligned} \tag{4.11}$$

Combinando las formulas (4.9) y (4.11) se obtiene la siguiente expresión para evaluar funciones:

$$S(x) = \phi_0(x)a_0 + \phi_1(x)b_1(x) + \beta_1(x)\phi_0(x)b_2(x) \tag{4.12}$$

Utilizando la ecuación que satisface la relación de recurrencia lineal (4.10) y la definición de polinomios de Chebyshev (4.1) se puede deducir:

$$\begin{aligned} \alpha_k(x) &= 2x \\ \beta_k(x) &= 1 \\ \phi &= T \end{aligned} \tag{4.13}$$

Si se sustituyen estos nuevos valores se obtiene la siguiente función de evaluación:

$$\begin{aligned} S(x) &= T_0(x)a_0 + T_1(x)b_1(x) + 1 \cdot T_0(x)b_2(x) \\ S(x) &= a_0 + xb_1(x) + b_2(x) \end{aligned} \tag{4.14}$$

donde:

$$\begin{aligned} b_{n+1} &= b_{n+2} = 0 \\ b_k(x) &= a_k + 2x \cdot b_{k+1}(x) + b_{k+2}(x) \end{aligned} \tag{4.15}$$

4.3. Comprobación de los polinomios

Para comprobar que los polinomios han sido calculados correctamente se podría comparar directamente los coeficientes generados con los coeficientes que usa SPICE, pero se ha decidido calcular el error relativo cometido al calcular el estado del planeta. Para

considerar que se ha calculado de manera correcta el estado de un planeta el error relativo debería ser aproximadamente de 10^{-14}.

La siguientes gráficas muestran el error relativo a la hora de interpolar el estado de la Tierra en 1000 instantes de tiempo uniformemente distribuidos entre el 1 de enero del 2022 y el 1 de enero del 2023. En el eje y vemos el error cometido mientras que en el eje x vemos el tiempo en formato ET.

4.3.1. Método manual

Las primeras tres gráficas muestran el error cometido para las tres coordenadas de la posición mientras que las tres siguientes muestran el error cometido para la velocidad en los tres ejes. Para poder apreciar las gráficas estas serán mostradas en escala logarítmica.

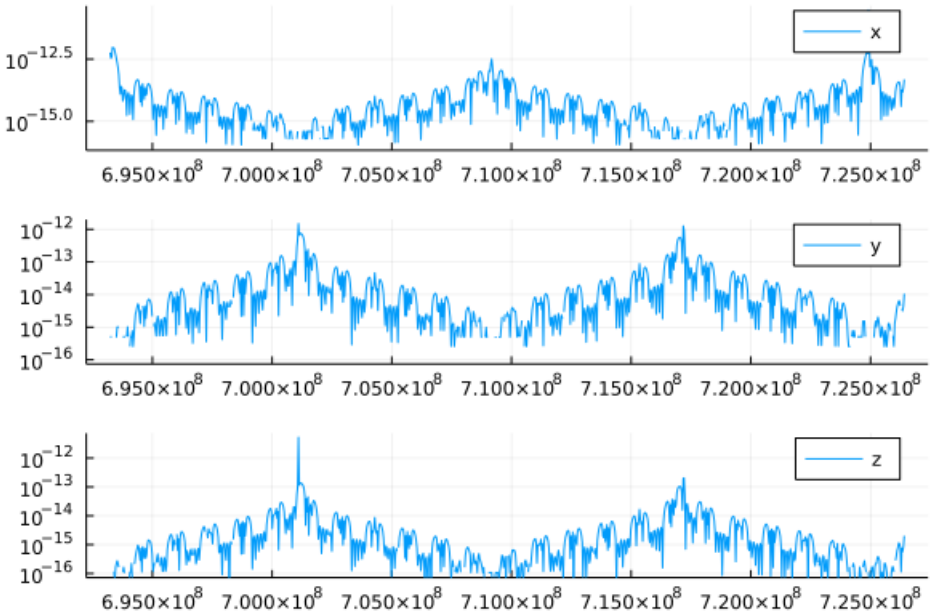

Figura 4.3: Error relativo de las coordenadas de la posición con el método manual

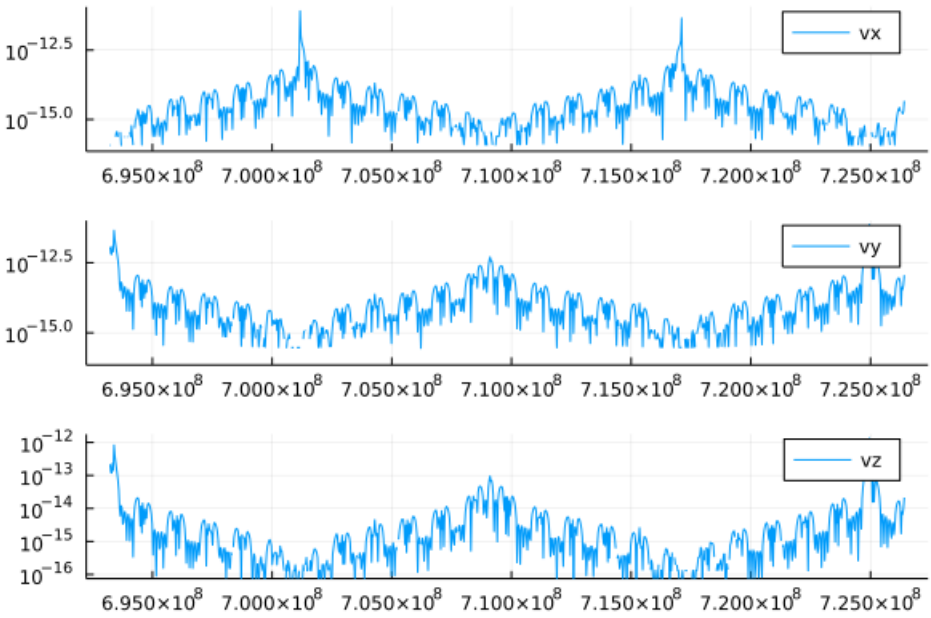

Figura 4.4: Error relativo de las coordenadas de la velocidad con el método manual

4.3.2. Método de la Transformada Discreta de los Cosenos

Las primeras tres gráficas muestran el error cometido para las tres coordenadas de la posición mientras que las tres siguientes muestran el error cometido para la velocidad en los tres ejes. Para poder apreciar las gráficas estas serán mostradas en escala logarítmica.

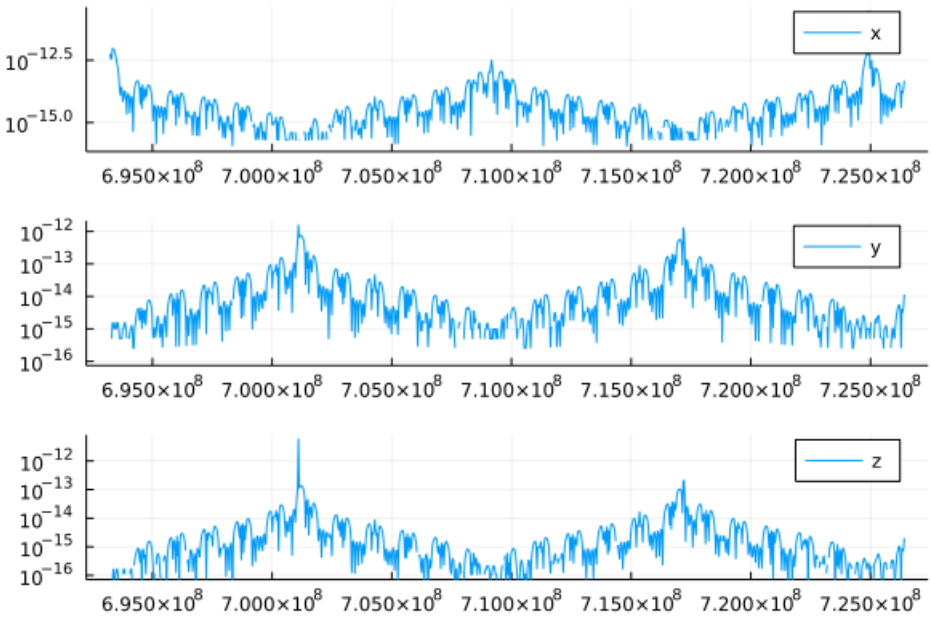

Figura 4.5: Error relativo de las coordenadas de la posición con el método de la Transformada Discreta de los Cosenos

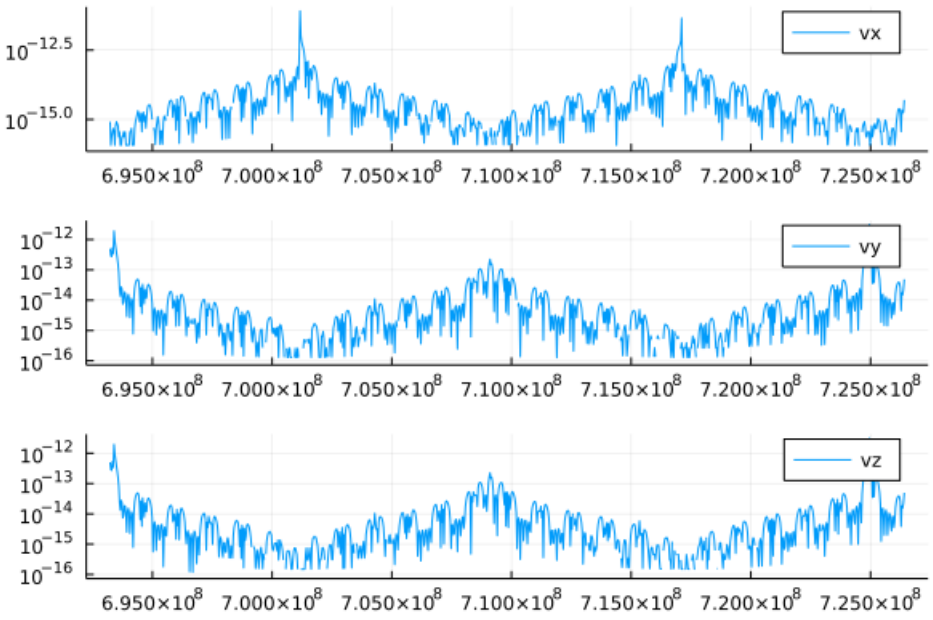

Figura 4.6: Error relativo de las coordenadas de la velocidad con el método de la Transformada Discreta de los Cosenos

4.3.3. Conclusiones de las gráficas

Se puede observar que se han obtenido resultados parecidos a los esperados, aun así en las gráficas se puede ver que hay picos en momentos determinados. En una primera instancia se podría pensar que estos picos ocurren en los instantes de tiempo que coinciden con el punto medio de un polinomio de Chebyshev, puesto que estos polinomios cometen un mayor error en el centro que en los extremos. También se podría pensar que estos errores pertenecen a los extremos de los polinomios y que por un error de código se está utilizando un polinomio incorrecto.

Sin embargo, se ha comprobado en qué instantes de tiempo ocurren estos picos de error y no coinciden con ninguna de las dos hipótesis. Al final se ha llegado a la conclusión de que estos picos ocurren debido a la acumulación de errores de redondeo. Y por lo tanto se dan por buenos los polinomios de interpolación obtenidos.

CAPÍTULO 5

Gestión de ficheros de coeficientes

Uno de los objetivos de este proyecto es conseguir almacenar los coeficientes en ficheros para poder evaluar los polinomios sin tener que realizar todo el proceso de obtención de los polinomios. Para esto se ha creado un paquete llamado LittleEphemeris con diferentes funcionalidades interesantes para este proyecto "(el manual se puede encontrar en el Anexo B)". Este paquete ha sido añadido al repositorio de paquetes de Julia "(ver Anexo A)". Además se puede encontrar en GitHub en el siguiente link: https://github.com/AitorIglesias/LittleEphemeris.jl, además con ayuda de la herramienta Documenter.jl se ha creado la siguiente pagina: https://aitoriglesias.github.io/LittleEphemeris.jl/index.html con la documentación del paquete

Este paquete está compuesto por cuatro ficheros.

5.1. generate_files.jl

Este fichero implementa una única funcionalidad a la que se puede acceder mediante a la función *generate_files*. Esta función almacena en la ruta indicada los ficheros necesarios para realizar cualquier operación del módulo. Estos ficheros son:

- naif0012.tls: Un fichero LSK que será necesario cargar antes de realizar cualquier operación que necesite realizar transformaciones de tiempo o que necesite obtener el estado de un cuerpo mediante al paquete SPICE.jl.
- de440.bsp: Es un fichero SPK que será necesario cargar antes de realizar cualquier operación que implique obtener el estado de un cuerpo mediante a SPICE.jl
- header_data.json: Es un fichero JSON con la información sobre la manera en la que están almacenados los polinomios de Chebyshev en SPICE. Esta información es necesaria para generar coeficientes desde cero. Los datos que hay dentro de este fichero son:
    - La fecha inicial del primer polinomio interpolador.
    - La fecha final del último polinomio interpolador.

    - El número de días que abarca un conjunto de polinomios.
    - Una lista con información de cada cuerpo. Esta información es la ID del planeta, el nombre del planeta, el número de polinomios que hay en cada conjunto y el número de coeficientes que tiene cada polinomio.
- time.csv: Es un fichero tipo CSV que contiene un vector con todos los intervalos de tiempo que se utiliza en SPICE.

5.2. generate_coefficients.jl

Este fichero implementa una única funcionalidad. Esta funcionalidad permite generar polinomios interpoladores para un cuerpo indicado y un intervalo de tiempo dado. El proceso que sigue es el explicado en el capítulo 4 de este documento.

5.3. manage_coefficients.jl

Este es probablemente el fichero más importante del módulo puesto que implementa las funcionalidades para gestionar los ficheros de coeficientes. Las funcionalidades que implementa este fichero son las siguientes:

- La posibilidad de crear un fichero de coeficientes dado un vector de IDs (o de nombres de planetas) y un vector de intervalos de tiempo.
- La posibilidad de crear un fichero de coeficientes partiendo de otro fichero de coeficientes dado un vector de IDs (o de nombres de planetas) y un vector de intervalos de tiempo.
- La posibilidad de añadir más polinomios a un fichero de coeficientes.
- La posibilidad de evaluar los polinomios que nos interesen a partir de un fichero de coeficientes.

Es necesario saber de que manera se van a almacenar los ficheros.

5.3.1. Formato de los ficheros

Los ficheros de coeficientes están acompañados por un fichero de información, los cuales son necesarios para poder interpretar de manera adecuada los ficheros de coeficientes. Estos ficheros son del tipo JSON y contienen una lista en la que cada elemento contiene los siguientes datos:

- La ID del cuerpo.
- El nombre del cuerpo.
- El número de polinomios que interpolan el estado de ese cuerpo.
- El número de coeficientes que tiene cada polinomio.
- Un vector con los intervalos de tiempo correspondientes a los polinomios.

En los ficheros de coeficientes se guardan los polinomios de Chebyshev vistos anteriormente:

$$P_j(t_k) = c_0 + \sum_{i=1}^{d} c_i T_i(x_k) \tag{5.1}$$

Estos ficheros están en formato CSV, es decir son ficheros que almacenan tablas. La tabla encontrada en estos ficheros siempre tendrá 8 columnas, debido a que todos los polinomios tiene un número de coeficientes igual a una potencia de dos y como poco tienen 8 coeficientes. Es por esto que un polinomio de grado 15 ocupará 2 filas. De está manera se aprovechan todas las celdas de la tabla.

Abajo se puede ver un ejemplo de un fragmento de una tabla. Este fragmento contiene dos polinomios, el primero con 8 coeficientes y el segundo con 16. Los coeficientes del primer polinomio se representan con la letra c y los coeficientes del segundo polinomio se representan con la letra u.

c_0	c_1	c_2	c_3	c_4	c_5	c_6	c_7
u_0	u_1	u_2	u_3	u_4	u_5	u_6	u_7
u_8	u_9	u_{10}	u_{11}	u_{12}	u_{13}	u_{14}	u_{15}

Los ficheros están divididos en tantas secciones como cuerpos hay en el fichero de información, es decir, cada sección contiene los polinomios interpoladores de su respectivo cuerpo. Dentro de cada sección hay tantas subsecciones como intervalos de tiempo. En cada una de estas subsecciones hay seis polinomios de Chebyshev. Si evaluamos los tres primeros polinomios en un instante t perteneciente a la subsección en la que se encuentra el polinomio, obtendremos las coordenadas de la posición del cuerpo en el instante t. Mientras que si evaluamos los otros tres polinomios en un instante t perteneciente a la subsección en la que se encuentra el polinomio, obtendremos las coordenadas de la velocidad del cuerpo en el instante t. Sabiendo la manera en la que se guarda la información y con el fichero de información es simple interpretar los ficheros de coeficientes.

Abajo se puede ver un ejemplo de un fichero de coeficientes donde hay l cuerpos y el primer cuerpo tiene polinomios de k coeficientes y n intervalos de tiempo. No es necesario que todos los cuerpos tengan el mismo número de intervalos o el mismo grado para los polinomios.

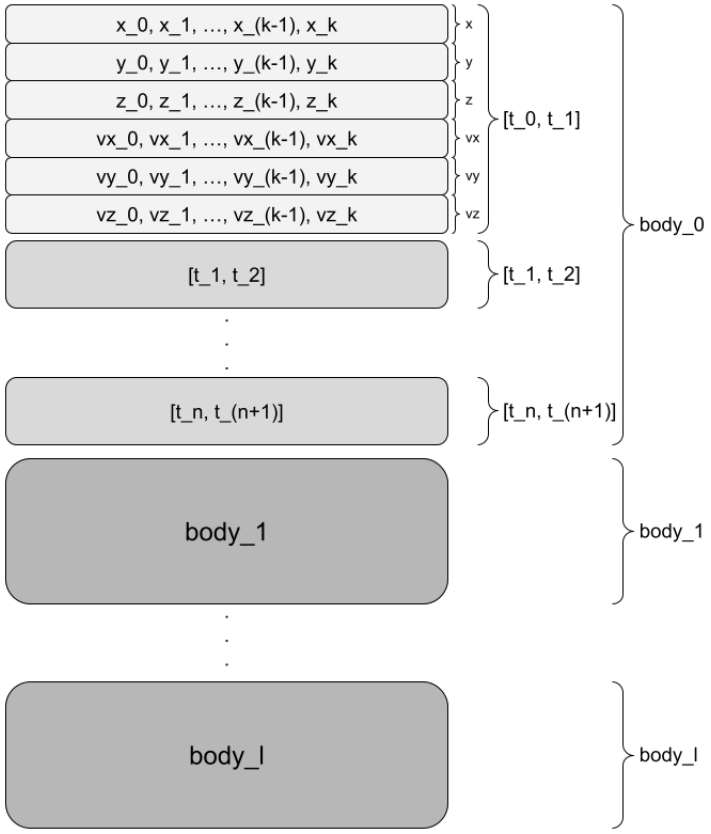

Figura 5.1: Ejemplo de fichero de coeficientes

Es importante mencionar que se podrían haber guardado únicamente los polinomios que interpolan las coordenadas de los cuerpos, y calcular las velocidades mediante a la derivada de estas funciones. Puesto que la velocidad es la variación de la posición respecto al tiempo, es decir la derivada de la función que obtiene la posición respecto al tiempo. Si se tiene la función que calcula la posición respecto al tiempo (que es exactamente la función interpoladora), entonces si se deriva esa función respecto al tiempo, se obtiene la velocidad. Sin embargo si se calcula la velocidad de esa manera el error de redondeo cometido sería aun mayor. Es por eso que se ha decidido utilizar más espacio para cometer un error más pequeño.

5.4. BodyCoeffs

Sin embargo, es posible que se tenga un fichero enorme de coeficientes pero que solo sea necesario un cuerpo especifico en un intervalo específico. En ese caso no se desea tener todo el fichero cargado en memoria. De esa idea surge la estructura BodyCoeffs. En esta estructura está guardada tanto la información como los coeficientes del cuerpo en un intervalo de tiempo. Además se pueden evaluar los polinomios en los instantes de tiempo deseados con un método propio de la estructura.

CAPÍTULO 6

Conclusiones y trabajo a futuro

Una vez explicado la base teórica detrás de la implementación del paquete LittleEphemeris, así como sus funcionalidades, es hora de hablar sobre las conclusiones de este proyecto. En este capítulo también se habla sobre el trabajo que se puede realizar para complementar o mejorar el paquete.

6.1. Conclusiones

Se han cumplido los objetivos iniciales del proyecto. Se ha diseñado e implementado una herramienta para poder obtener las coordenadas y las velocidades de cualquier cuerpo celeste del que se tengan un mínimo de datos. Este paquete permite a investigadores trabajar con los polinomios de Chebyshev o con las coordenadas de los cuerpos celestes que deseen accediendo a las bases de datos solo para generar los polinomios interpoladores. Una vez que se hayan generado se puede calcular la posición (y la velocidad) localmente sin tener que acceder a las bases de datos externas.

Además este paquete no está únicamente en el registro de paquetes de Julia, sino que también está disponible en un repositorio público de GitHub. De esta manera cualquiera puede descargar este paquete y utilizarlo para sus propios proyectos.

6.2. Trabajo a futuro

En el cuarto capítulo se puede ver que hay picos en el error relativo. Una posible mejora de este paquete sería implementar diferentes técnicas para disminuir el error de redondeo acumulado. Se pude ampliar la funcionalidad de la generación de los polinomios interpoladores de forma que se calculen a partir de datos obtenidos mediante cálculos muy precisos que requieren mucho tiempo de computación. A partir de esos cálculos se pueden genera los polinomios de interpolación que aproximen esos costosos datos y utilizarlos para compararlos con otros resultados.

En cuanto al paquete LittleEphemeris, este tiene muchas funcionalidades que se le podrían

añadir para que este sea un paquete de gestión de Efemérides más completo. Por ejemplo se podría añadir una funcionalidad que permita realizar transformaciones entre formatos de tiempo.

Bibliografía

[1] D. B. Hunter, «Clenshaw's Method for Evaluating Certain Finite Series,» *The Computer Journal*, vol. 13, n.º 4, págs. 378-381, nov. de 1970, ISSN: 0010-4620. DOI: 10.1093/comjnl/13.4.378. eprint: https://academic.oup.com/comjnl/article-pdf/13/4/378/1438443/13-4-378.pdf. dirección: https://doi.org/10.1093/comjnl/13.4.378.

[2] N. Ahmed, T. Natarajan y K. Rao, «Discrete Cosine Transform,» *IEEE Transactions on Computers*, vol. C-23, n.º 1, págs. 90-93, 1974. DOI: 10.1109/T-C.1974.223784.

[3] W.-H. Chen, C. Smith y S. Fralick, «A Fast Computational Algorithm for the Discrete Cosine Transform,» *IEEE Transactions on Communications*, vol. 25, n.º 9, págs. 1004-1009, 1977. DOI: 10.1109/TCOM.1977.1093941.

[4] J. OLIVER, «An Error Analysis of the Modified Clenshaw Method for Evaluating Chebyshev and Fourier Series,» *IMA Journal of Applied Mathematics*, vol. 20, n.º 3, págs. 379-391, nov. de 1977, ISSN: 0272-4960. DOI: 10.1093/imamat/20.3.379. eprint: https://academic.oup.com/imamat/article-pdf/20/3/379/1851253/20-3-379.pdf. dirección: https://doi.org/10.1093/imamat/20.3.379.

[5] ——, «An Error Analysis of the Modified Clenshaw Method for Evaluating Chebyshev and Fourier Series,» *IMA Journal of Applied Mathematics*, vol. 20, n.º 3, págs. 379-391, nov. de 1977, ISSN: 0272-4960. DOI: 10.1093/imamat/20.3.379. eprint: https://academic.oup.com/imamat/article-pdf/20/3/379/1851253/20-3-379.pdf. dirección: https://doi.org/10.1093/imamat/20.3.379.

[6] C. Acton, «Ancillary Data Services of NASA's Navigation and Ancillary Information Facility,» *Planetary and Space Science*, vol. 44, n.º 1, págs. 65-70, 1996.

[7] G. Strang, «The Discrete Cosine Transform,» 1999.

[8] J. Mason y D. C. Handscomb, *Chebyshev Polynomials*, 1.ª ed. CRC, 2002.

[9] A. Smoktunowicz, «Backward Stability of Clenshaw's Algorithm,» *BIT Numerical Mathematics*, vol. 42, n.º 3, págs. 600-610, sep. de 2002, ISSN: 1572-9125. DOI: 10.1023/A:1022001931526. dirección: https://doi.org/10.1023/A:1022001931526.

[10] S. A. Sarra, «Chebyshev Interpolation: an interactive tour,» *Journal of Online Mathematics and Its Applications*, 2005.

[11] V. Britanak, P. C. Yip y K. Rao, «CHAPTER 1 - Discrete Cosine and Sine Transforms,» en *Discrete Cosine and Sine Transforms*, V. Britanak, P. C. Yip y K. Rao, eds., Oxford: Academic Press, 2007, págs. 1-15, ISBN: 978-0-12-373624-6. DOI: https://doi.org/10.1016/B978-012373624-6/50003-5. dirección: https://www.sciencedirect.com/science/article/pii/B9780123736246500035.

[12] ——, «CHAPTER 4 - Fast DCT/DST Algorithms,» en *Discrete Cosine and Sine Transforms*, V. Britanak, P. C. Yip y K. Rao, eds., Oxford: Academic Press, 2007, págs. 73-140, ISBN: 978-0-12-373624-6. DOI: https://doi.org/10.1016/B978-012373624-6/50006-0. dirección: https://www.sciencedirect.com/science/article/pii/B9780123736246500060.

[13] A. Bostan y É. Schost, «Polynomial evaluation and interpolation on special sets of points,» *Journal of Complexity*, 2010.

[14] N. Brisebarre y M. M. Joldes, «Chebyshev Interpolation Polynomial-based Tools for Rigorous Computing,» *International Symposium on Symbolic and Algebraic Computation*, 2010.

[15] J. D. Cook, *Chebyshev interpolation*, 2017. dirección: https://www.johndcook.com/blog/2017/11/06/chebyshev-interpolation/.

[16] V. Ledoux y G. Moroz, «Evaluation of Chebyshev polynomials on intervals and application to root finding,» *Mathematical Aspects of Computer and Information Sciences*, 2019. dirección: https://doi.org/10.1093/imamat/20.3.379.

[17] *Ephemeris Subsystem SPK*, ene. de 2020. dirección: https://naif.jpl.nasa.gov/pub/naif/toolkit_docs/Tutorials/pdf/individual_docs/18_spk.pdf.

[18] *Introduction to Kernels*, ene. de 2020. dirección: https://naif.jpl.nasa.gov/pub/naif/toolkit_docs/Tutorials/pdf/individual_docs/12_intro_to_kernels.pdf.

[19] *Leapseconds and Spacecraft Clock Kernels*, ene. de 2020. dirección: https://naif.jpl.nasa.gov/pub/naif/toolkit_docs/Tutorials/pdf/individual_docs/16_lsk_and_sclk.pdf.

[20] T. J. Rivlin, *Chebyshev Polynomials*. Courier Dover Publications, 2020.

[21] *Time Conversion and Time Formats*, ene. de 2020. dirección: https://naif.jpl.nasa.gov/pub/naif/toolkit_docs/Tutorials/pdf/individual_docs/15_time.pdf.

[22] *Chebyshev nodes — Wikipedia, The Free Encyclopedia*, 2022. dirección: https://en.wikipedia.org/wiki/Chebyshev_nodes.

[23] *Chebyshev polynomials — Wikipedia, The Free Encyclopedia*, 2022. dirección: https://en.wikipedia.org/wiki/Chebyshev_polynomials.

[24] *Clenshaw algorithm — Wikipedia, The Free Encyclopedia*, 2022. dirección: https://en.wikipedia.org/wiki/Clenshaw_algorithm.

[25] *Coordinated Universal Time — Wikipedia, The Free Encyclopedia*, 2022. dirección: https://en.wikipedia.org/wiki/Coordinated_Universal_Time.

[26] *Coordinated Universal Time — Wikipedia, The Free Encyclopedia*, 2022. dirección: https://en.wikipedia.org/wiki/Ephemeris_time.

[27] *Discrete cosine transform — Wikipedia, The Free Encyclopedia*, 2022. dirección: https://en.wikipedia.org/wiki/Discrete_cosine_transform.
[28] *Efemérides — Wikipedia, The Free Encyclopedia*, 2022. dirección: https://en.wikipedia.org/wiki/Ephemeris.
[29] *Julian Date — Wikipedia, The Free Encyclopedia*, 2022. dirección: https://en.wikipedia.org/wiki/Julian_day.
[30] *Chebyshev Polynomial of the First Kind – from Wolfram MathWorld*. dirección: https://mathworld.wolfram.com/ChebyshevPolynomialoftheFirstKind.html.
[31] *Chebyshev Polynomial of the Second Kind – from Wolfram MathWorld*. dirección: https://mathworld.wolfram.com/ChebyshevPolynomialoftheSecondKind.html.
[32] *Clenshaw Recurrence Formula."From MathWorld–A Wolfram Web Resource*. dirección: https://mathworld.wolfram.com/ClenshawRecurrenceFormula.html.
[33] N. F. Marshall, «Chebyshev Interpolation,»
[34] T. Sun, «Chebyshev Interpolation for Function in 1D,»

Anexos

ANEXO A

Creación de paquetes en el entorno Julia

En este breve anexo se explica el proceso para generar un paquete de Julia desde cero y añadir este al repositorio de paquetes de Julia. Para ello se usará como ejemplo el paquete LittleEphemeris.

A.1. Crear un paquete

Un paquete de Julia tiene dos elementos obligatorios el primero es una carpeta src con el modulo del paquete y el segundo es un fichero Project.toml, este fichero contiene información sobre el paquete. Si bien la carpeta y el modulo del paquete se pueden crear manualmente, para crear el fichero Project.toml es necesario hacer uso del paquete de Julia Pkg. Esto debido a que Julia tiene que asignarle un identificador único a tu paquete. Para crear el paquete vacío hay que seguir los siguientes pasos:

Entrar al REPL del paquete Pkg de Julia, para eso en la terminal de Julia se debe escribir `]`. Una vez dentro se podrá crear un paquete vacío con el siguiente comando:

```
(@v1.6) pkg> generate LittleEphemeris
```

En vez de LittleEphemeris se debe poner el nombre del paquete que se desea crear. Una vez ejecutado el comando se habrá creado el paquete con los elementos obligatorios mencionados anteriormente.

El modulo del paquete tendrá una única función, no es necesario conservar esta función. Este fichero será el que se ejecutará al importar el paquete.

El fichero Project.toml contendrá el nombre del fichero el identificador único del paquete, una lista con los nombres y los correos de los autores y la versión del paquete. Así es como se ve en el caso de LittleEphemeris:

```
name = "LittleEphemeris"
uuid = "216eeeb1-80f3-427b-9618-77bd73e1755d"
authors = ["AitorIglesias <aitoriglesiashernandez@gmail.com>"]
version = "0.1.0"
```

Únicamente con estos datos en el Project.toml sería suficiente para que el paquete funcione correctamente, sin embargo en caso de tener imports en el código habría que indicarlo en el fichero Project.toml. Indicarlo no es necesario para que el paquete funcione, pero si es recomendable, además en caso de querer registrar el paquete si será obligatorio.

Para indicar las dependencias en el fichero Project.toml hay que añadir dos apartados, el primero [deps] se deben indicar los nombres de los paquetes utilizados junto con los identificadores únicos de estos. En el segundo apartado [compat] se deben indicar las versiones de Julia y de los paquetes utilizados.

Para comprobar la versión de Julia que se está utilizando hay que escribir el siguiente comando:

```
julia> VERSION
v"1.6.1"
```

Para comprobar la versión de un paquete de Julia hay que escribir el siguiente comando:

```
(@v1.6) pkg> satatus SPICE
    Status `~/.julia/environments/v1.6/Project.toml`
  [5bab7191] SPICE v0.2.2
```

Para comprobar el identificador único no hay ningún comando especifico, sin embargo, este se puede encontrar en el fichero Project.toml del paquete instalado en el ordenador. La ruta del fichero es la misma que ha indicado el comando anterior.

El paquete LittleEphemeris hace uso de los paquetes: CSV, DataFrames, FFTW, JSON y SPICE. Por lo que el Project.toml de este paquete es el siguiente:

```
name = "LittleEphemeris"
uuid = "216eeeb1-80f3-427b-9618-77bd73e1755d"
authors = ["AitorIglesias <aitoriglesiashernandez@gmail.com>"]
version = "0.1.0"

[deps]
```

```
CSV = "336ed68f-0bac-5ca0-87d4-7b16caf5d00b"
DataFrames = "a93c6f00-e57d-5684-b7b6-d8193f3e46c0"
FFTW = "7a1cc6ca-52ef-59f5-83cd-3a7055c09341"
JSON = "682c06a0-de6a-54ab-a142-c8b1cf79cde6"
SPICE = "5bab7191-041a-5c2e-a744-024b9c3a5062"

[compat]
julia = "1.6.1"
CSV = "0.10.2"
DataFrames = "1"
FFTW = "1"
JSON = "0.21.3"
SPICE = "0.2.2"
```

Con esto el paquete ya es funcional, sin embargo solo las personas que tengan el paquete podrán usarlo, ya que el paquete no pertenece al repositorio de paquetes de Julia. Para usar un paquete que no pertenece al repositorio de paquetes de Julia es necesario activarlo cada vez que este va a ser usado. En julia un paquete se activa de la siguiente manera:

```
julia> cd("LittleEphemeris")
(@v1.6) pkg> activate .
```

A.2. Registrar un paquete

Para registrar un paquete en el repositorio de paquetes de Julia lo primero es crear un repositorio público de GitHub donde se debe guardar el paquete creado anteriormente. Este repositorio debe cumplir ciertos requisitos, se pueden encontrar todas en la documentación del paquete RegistryCI.jl. He aquí los requisitos más importantes:

- El repositorio debe tener una licencia Open Source.
- El nombre del paquete debe ser distinto al resto de los paquetes pertenecientes a la lista de paquetes de Julia.
- El nombre del paquete debe estar compuesto por caracteres alfanuméricos, (por lo menos 5) y debe comenzar con una letra mayúscula, además debe tener al menos una letra minúscula.
- El nombre del repositorio debe ser el nombre del paquete seguido de .jl.

Una vez se hayan cumplido estos requisitos ya se puede registrar el paquete, para ello en el último commit del repositorio del paquete que se quiere registrar se debe añadir el siguiente comentario: @JuliaRegistrator register. En menos de un minuto un bot llamado JuliaRegistrator te responderá al comentario indicando que ha hecho un pull request en el

repositorio JuliaRegistries/General. Si el paquete cumple todos requisitos, un colaborador del paquete JuliaRegistries/General aceptará el pull request (este proceso puede llegar ha tardar días). De esta manera el paquete quedará registrado y cualquiera podrá utilizarlo.

A.3. Página de documentación del paquete

En este apartado se explica cómo crear una página con la documentación del paquete, con ayuda del paquete de Julia Documenter.jl. La creación de la página no es nada necesaria, sin embargo esto será de gran ayuda para los usuarios del paquete ya que de esta manera serán capaces de utilizar el paquete de manera apropiada. Es necesario mencionar que se puede crear la página de documentación de muchas maneras diferentes y con muchos estilos, pero en este articulo se explica la manera en la que se creó la página de documentación del paquete LittleEphemeris.

El primer paso es instalar el paquete de Julia Documenter.jl para ello hay que escribir el siguiente comando:

```
(@v1.6) pkg> add Documenter
```

Esta herramienta permite al usuario crear una página de documentación con el mismo estilo que utilizan la mayoría de páginas de documentación de paquetes de Julia. Para ello hace uso de ficheros de Markdown que le indican al software lo que este debe añadir en cada página.

El siguiente paso es crear una carpeta llamada src y un fichero llamado make.jl en una carpeta docs dentro del paquete que queremos documentar. En el fichero make.jl estará el código para la creación de la página. Por ejemplo:

```
using LittleEphemeris
using Documenter

makedocs(
    format = Documenter.HTML(
        prettyurls = get(ENV, "CI", nothing) == "true",
    ),
    sitename = "LittleEphemeris.jl",
    authors = "Aitor Iglesias",
    pages = [
        "Home" => "index.md",
        "API" => "api.md",
    ],
    doctest = false,
)
```

```
deploydocs(
    repo = "github.com/AitorIglesias/LittleEphemeris.git",
    target = "build",
)
```

Este código está dividido en dos partes en la primera se llama a la función makedocs. El parámetro pages de la llamada, indica las páginas que la API debe crear y donde puede encontrar el contenido de lo que debe añadir a dichas páginas, este contenido debe ser un fichero Markdown. En la segunda parte del código se le indica en qué repositorio se encuentra el código, de esta manera, la API añadirá links al código de cada una de las funciones en la documentación de cada una de dichas funciones. De forma que al hacer click al botón source en la explicación de cada función, la página te llevará al código de dicha función en el repositorio Git.

En la carpeta src se debe añadir el contenido que tendrá la página web. En el caso de LittleEphemeris, este contiene dos páginas: Home que está basada en el fichero index.md y API está basado en el fichero api.md. A su vez el fichero api.md contiene indicaciones para que la API genere la documentación de las funciones públicas del paquete, basandose en la documentación del código. El aspecto del fichero es el siguiente:

````
# API

```@meta
DocTestSetup = quote
    using LittleEphemeris
end
```

```@autodocs
Modules = [LittleEphemeris]
Private = false
```
````

A pesar de no ser necesario, en caso de querer añadir un logo a la página, se debe crear una carpeta llamada assets en la carpeta src y en esta añadir la imagen con el nombre logo. En caso de tener un logo diferente para la versión oscura de la página se debe añadir esta imagen en la misma carpeta con el nombre logo-dark. En caso de tener más imágenes para la página es recomendable guardar estas en esta carpeta.

El último paso para la creación de la página es ejecutar el fichero make.jl, al hacerlo se creará una carpeta llamada build dentro de la carpeta docs con la página.

A.4. Alojamiento de la página

La página ya ha sido creada, sin embargo esta no está alojada en ningún lado, una página se puede alojar el la web de muchas maneras diferentes. En este último apartado se explicará como se puede alojar la página de documentación del paquete de manera gratuita. Para ello se hará uso de la funcionalidad de GitHub, GitHub Pages.

En el repositorio del paquete correspondiente se debe crear una branch llamada gh-pages. En esta branch se deben añadir los ficheros HTML, JavaScript, CSS, etc. de la página de documentación. Al darle el nombre gh-pages a la branch, GitHub alojará automáticamente los ficheros HTML en el dominio: https://Usuario.github.io/NombreDelPaquete.

ANEXO B

Manual de usuario de LittleEphemeris

En este anexo se puede encontrar un manual de usuario para el paquete de Julia LittleEphemeris. Esta manual está dividido en dos partes. La primera parte, explica paso a paso cómo utilizar todas las funcionalidades del paquete. La segunda parte son las descripciones de las funciones del paquete.

B.1. Introducción

LittleEphemeris es un paquete de Julia que permite realizar una gestión de Efemérides de manera ágil. El uso de este paquete es para usuarios que no necesiten bases de datos de Efemérides tan grandes como las que encontramos hoy en día, es decir, usuarios que necesitan una base de datos más específica, con menos cuerpos y para un intervalo de tiempo más reducido.

Este paquete, entre otras cosas, permite crear un fichero de polinomios que interpolan las coordenadas de posición y de velocidad de diferentes cuerpos en diferentes intervalos de tiempo para después poder evaluar estos polinomios en los instantes de tiempo deseados, ya sea directamente desde el fichero de polinomios o construyendo un objeto que almacene los polinomios y llamando a un método que evalúa los polinomios en el instante de tiempo deseado.

B.2. Instalación del paquete

El paquete LittleEphemeris puede ser fácilmente instalado con los siguientes comandos de Julia:

```
julia> using Pkg
julia> Pkg.add("LittleEphemeris")
```

B.3. Descarga de ficheros necesarios

Para que el paquete LittleEphemeris sea capaz de generar un fichero de polinomios necesita algunos ficheros, los cuales son:

- Un fichero de gestión de Efemérides también conocido como fichero SPK (Spacecraft and Planet Ephemeris Kernel). De este fichero se sacan los valores en los nodos para realizar interpolación polinómica y generar los polinomios.
- Un fichero LSK (Leapseconds Kernel), este fichero puede ser utilizado para realizar transformaciones entre formatos de tiempo.
- Un fichero CSV con una columna con los intervalos de tiempo que utiliza el fichero SPK descargado. Este fichero se utiliza para generar polinomios en los mismos intervalos de tiempo que utiliza el fichero SPK descargado.
- Un fichero JSON con los parámetros de los polinomios interpoladores contenidos en los ficheros SPK. Este fichero sirve para conocer el grado de los polinomios y el número de polinomios contenidos en cada intervalo.

Si bien estos ficheros pueden ser generados manualmente, el paquete LittleEphemeris, tiene implementada una funcionalidad para generar estos ficheros. Es necesario mencionar que esta funcionalidad puede tardar algunos minutos y el tiempo que tarde dependerá de la conexión a internet que posea el equipo que esté ejecutando el programa. Para ejecutar la funcionalidad se deben escribir los siguientes comandos de Julia:

```
julia> using LittleEphemeris
julia> generate_files("./data/")
```

Mediante estos comandos se generarán los ficheros mencionados anteriormente en una carpeta llamada data. El fichero JSON contiene únicamente datos de los baricentros de los planetas del sistema solar, junto con datos del baricentro de Plutón, datos del Sol y de la Luna. Si se quisieran generar polinomios de otros cuerpos o polinomios de los mismo cuerpos con un grado diferente, habría que modificar este fichero manualmente.

Este es un ejemplo de un pequeño fichero de datos de polinomios (no es el fichero que genera la función generate_files):

```
{
    "firstDate": -1.42007472e10,
    "lastDate": 2.05140816e10,
    "numberOfDays": 32,
    "bodyData": [
        {
            "bodyID": 3,
            "bodyName": "EARTH BARYCENTER",
```

```
            "numberOfSets": 2,
            "numberOfCoeffs": 13
        },
        {
            "bodyID": 4,
            "bodyName": "MARS BARYCENTER",
            "numberOfSets": 1,
            "numberOfCoeffs": 11
        }
    ]
}
```

Si se quisiera añadir por ejemplo dos cuerpos, el primero el baricentro de la Tierra con 32 coeficientes y el segundo la Tierra con 16 coeficientes, el fichero quedaría así:

```
{
    "firstDate": -1.42007472e10,
    "lastDate": 2.05140816e10,
    "numberOfDays": 32,
    "bodyData": [
        {
            "bodyID": 3,
            "bodyName": "EARTH BARYCENTER",
            "numberOfSets": 2,
            "numberOfCoeffs": 13
        },
        {
            "bodyID": 4,
            "bodyName": "MARS BARYCENTER",
            "numberOfSets": 1,
            "numberOfCoeffs": 11
        },
        {
            "bodyID": 3,
            "bodyName": "EARTH BARYCENTER 2",
            "numberOfSets": 2,
            "numberOfCoeffs": 32
        },
        {
            "bodyID": 399,
            "bodyName": "EARTH",
            "numberOfSets": 2,
            "numberOfCoeffs": 16
        }
    ]
```

```
}
```

No es recomendable modificar los parámetros firstDate, lastDate y numberOfDays, ya que estos están para evitar errores. Además, se pueden tener dos cuerpos con la misma ID ya que esta es utilizada para llamar a una función del paquete SPICE.jl. Sin embargo, para que el software del paquete LittleEphemeris funcione de la manera deseada en esta situación, se deberán usar las funciones para la generación del fichero de coeficientes refiriéndose al nombre del cuerpo en vez de a la ID. Por ello se le deben dar distintos nombres a ambos cuerpos.

B.4. Generación de una tabla de coeficientes

Una vez se tienen los ficheros necesarios, se pueden generar los polinomios deseados. Para ello existe la función generate_coeffs, que devuelve una tabla de polinomios. Sin embargo, para utilizar esta función, primero es necesario cargar los Kernels descargados anteriormente, si bien esto podría hacerlo la función que genera los coeficientes. Esto tiene algunos inconvenientes, los cuales son:

- Los Kernels tardan bastante en cargarse por lo que el programa para generar los coeficientes tardaría más de lo necesario.
- No hay manera de comprobar si un Kernel específico está cargado, por lo que en caso de llamar al programa generate_files varias veces, este cargaría varias veces los Kernels.

Para evitar estos inconvenientes se ha decidido que los Kernels sean cargados con anterioridad. Para ello se puede hacer uso del paquete SPICE.jl puesto que el paquete LittleEphemeris no contiene esta funcionalidad. Para ello se deben ejecutar los siguientes comandos:

```
julia> using SPICE
julia> furnsh("data/naif0012.tls", "data/de440.bsp")
```

El método furnsh cargará los Kernels indicados. Si se desean cargar varios Kernels es recomendable indicarlos todos en la misma llamada, por cuestiones de eficiencia.

Una vez cargados los Kernels se puede llamar a la función generate_coeffs. Esta función no depende del fichero JSON, pues el usuario indica cómo son los polinomios que desea. He aquí un ejemplo de cómo obtener los polinomios del baricentro de la Tierra para el año 2022:

```
# Intervalo de tiempo al que pertenecen los polinomios
et_0 = utc2et("2022-01-01T12:00:00")
et_end = utc2et("2023-01-01T12:00:00")
```

```
# Parámetros de los polinomios
ID = 3 # ID del baricentro de la Tierra
n_coeffs = 16 # Número de coeficientes del polinomio que se desea
n_sets = 2 # Número de polinomios en cada intervalo de tiempo contenido
    ↪ en el fichero de intervalos de tiempo

time_vec, x, y, z, vx, vy, vz = generate_coeffs(et_0, et_end, (ID,
    ↪ n_coeffs, n_sets), "data/time.csv")
```

La primera función utilizada, utc2et, es una función propia del paquete SPICE.jl, que permite transformar del formato de tiempo UTC al formato de tiempo ET. El paquete LittleEphemris trabaja exclusivamente con el formato de tiempo ET, sin embargo, no tiene un método que realice transformaciones entre formatos de tiempo, es por ello que si se tienen los instantes de tiempo en un formato distinto al formato ET, será necesario hacer uso de otras aplicaciones para realizar esta transformación.

generate_coeffs devolverá un vector con los intervalos de tiempos de los polinomios generados junto a seis matrices. Cada una de estas matrices corresponderán a una coordenada, siendo las tres primeras las coordenadas de posición y las tres últimas las coordenadas de velocidad. Cada fila de estas matrices es un polinomio, más específicamente el polinomio interpolador para el intervalo de tiempo correspondiente en el vector de intervalos de tiempo. La matriz tendrá tantas columnas como coeficientes, por lo que cada uno de los elementos de la matriz corresponde a un coeficiente.

Sin embargo, puesto que los datos están almacenados en variables, es necesario generarlos cada vez que van a ser usados. El proceso es bastante rápido por lo que no supone un gran inconveniente. Aun así es más cómodo tener estos datos en un fichero. Por ello, LittleEphemeris proporciona las funcionalidades explicadas en las siguientes secciones de este anexo.

B.5. Generación de un fichero de coeficientes

Para generar ficheros de coeficientes al igual que para generar una tabla de coeficientes es necesario tener cargados los Kernels. Una vez cargados, se puede generar un fichero de coeficientes de manera bastante sencilla con la función create_coeffs_file. Un ejemplo del uso de la función sería el siguiente:

```
ID_list = [1, 3] # ID de los cuerpos (también funciona con los nombres de
    ↪  los cuerpos)
time_interval = (utc2et("2022-01-01T12:00:00"), utc2et("2022-02-01T00
    ↪ :00:00"))
time_interval_list = fill(time_interval, 2)
```

```
create_coeffs_file("data/coeffs.json", "data/coeffs.csv", ID_list,
    ↪ time_interval_list, "data/header_data.json", "data/time.csv")
```

Para llamar a esta función es necesario que los cuerpos indicados estén ordenados de manera ascendente respecto a las IDs de dichos cuerpos. Esto debido a que el programa que genera la tabla no los ordenará y el algoritmo de búsqueda del fichero que utiliza el paquete LittleEphemeris da por hecho que están ordenados, debido a cuestiones de eficiencia. Esta función genera un fichero de coeficientes y un fichero de información correspondiente al fichero de coeficientes.

El fichero de coeficientes está en formato CSV, es decir, es un fichero que almacena tablas. La tabla encontrada en estos ficheros siempre tendrá 8 columnas, debido a que todos los polinomios tiene un número de coeficientes igual a una potencia de dos y como poco tienen 8 coeficientes, esto debido a cuestiones de eficiencia.

El fichero de coeficientes está dividido en tantas secciones como cuerpos se le hayan indicado a la función create_coeffs_file, es decir, los cuerpos de secciones son los mismos que los indicados en el vector. Dentro de cada sección hay tantas subsecciones como intervalos de tiempo. En cada una de estas subsecciones hay seis polinomios de Chebyshev, los tres primeros sirven para calcular las coordenadas de la posición del cuerpo en el instante específico, mientras que los tres últimos sirven para calcular las coordenadas de la velocidad del cuerpo en el instante específico.

El fichero de información generado por la función, es necesario para poder interpretar de manera adecuada el fichero de coeficientes. Este fichero es del tipo JSON y contienen una lista en la que cada elemento contiene los siguientes datos:

- La ID del cuerpo.
- El nombre del cuerpo.
- El número de polinomios que interpolan el estado de ese cuerpo.
- El número de coeficientes que tiene cada polinomio.
- Un vector con los intervalos de tiempo correspondientes a los polinomios.

Este sería el fichero JSON generado por el código anterior:

```
[
    {
        "bodyID": 1,
        "bodyName": "MERCURY BARYCENTER",
        "numberOfPolynomials": 8,
        "numberOfCoeffs": 16,
        "timeIntervals": [
            6.932304e8,
```

```
            6.939216e8,
            6.946128e8,
            6.95304e8,
            6.959952e8,
            6.966864e8,
            6.973776e8,
            6.980688e8,
            6.9876e8
        ]
    },
    {
        "bodyID": 3,
        "bodyName": "EARTH BARYCENTER",
        "numberOfPolynomials": 4,
        "numberOfCoeffs": 16,
        "timeIntervals": [
            6.932304e8,
            6.946128e8,
            6.959952e8,
            6.973776e8,
            6.9876e8
        ]
    }
]
```

B.6. Generar fichero de coeficientes a partir de un fichero de coeficientes

Las siguientes funcionalidades no necesitarán los ficheros generados por la función generate_files, por lo que si ya ha sido creado un fichero de coeficientes con la información necesaria, estos ficheros pueden ser eliminados. Aún así se recomienda mantener el fichero LSK para realizar transformaciones entre los distintos formatos de tiempo.

Una funcionalidad que proporciona LittleEphemeris es la de crear un fichero de coeficientes más pequeño a partir de un fichero de coeficientes. Esto puede resultar útil si se quieren evaluar únicamente ciertos cuerpos, pero se quiere tener información sobre más cuerpos para trabajar con esta información en el futuro. Para utilizar esta funcionalidad se debe usar la función generate_subfile de la siguiente manera:

```
ID_list = [3] # ID de los cuerpos (también funciona con los nombres de
    ↪ los cuerpos)
time_interval = (utc2et("2022-01-01T12:00:00"), utc2et("2022-02-01T00
    ↪ :00:00"))
time_interval_list = fill(time_interval, 1)
```

```
generate_subfile("data/coeffs_subfile.json", "data/coeffs_subfile.csv",
    ↪ ID_list, time_interval_list, "data/coeffs.json", "data/coeffs.csv"
    ↪ )
```

Esta función es igual a la función create_coeffs_file solo que no necesita ficheros adicionales a los ficheros de coeficientes generados por el paquete LittleEphemeris. Además, únicamente copia datos de otro fichero, es decir, no tiene que generar polinomios, por ello, es mucho más rápido que la función create_coeffs_file.

B.7. Evaluar un fichero de coeficientes

Para poder darle uso a un fichero de coeficientes, el paquete LittleEphemeris proporciona la funcionalidad de evaluar los polinomios de este fichero para los instantes de tiempo deseados. Para ello hay que hacer uso de la función eval_coeffs_file, esta función permite evaluar los polinomios de un cuerpo en varios instantes de tiempo. Este es un ejemplo de como usar la función eval_coeffs_file:

```
ID = 3
et_0 = utc2et("2022-01-01T12:00:00")
et_end = utc2et("2022-02-01T12:00:00")

x, y, z, vx, vy, vz = eval_coeffs_file("data/coeffs.json", "data/coeffs.
    ↪ csv", ID, [et_0:10000:et_end])
```

B.8. Objeto de coeficientes de un cuerpo

Para finalizar es posible que se desee trabajar únicamente con un cuerpo. Por ello LittleEphemeris tiene la funcionalidad de generar un objeto de coeficientes de un cuerpo a partir de un fichero de coeficientes. De esta manera se puede tener la información cargada en memoria y así evaluar los polinomios de una manera más ágil. Este objeto se puede crear mediante su constructora:

```
ID = 3
time_interval = (utc2et("2022-01-01T12:00:00"), utc2et("2022-02-01T00
    ↪ :00:00"))

Earth = BodyCoeffs("data/coeffs.json", "data/coeffs.csv", ID,
    ↪ time_interval)
```

Evaluar estos objetos es de lo más sencillo puesto que solo es necesario indicarle el instante de tiempo en el que se quiere evaluar, de la siguiente manera:

```
t = utc2et("2022-01-107T00:00:00")
Earth(t)
```

B.9. Descripciones de las estructuras y las funciones y del paquete LittleEphemeris

---

BodyCoeffs — Tipo: Estructura

Descripción:

Estructura de coeficientes de un cuerpo.

Campos de la estructura:

Campo	Tipo	Descripción
bodyID	Int	ID del cuerpo.
bodyName	String	Nombre del cuerpo.
numberOfPolynomials	Int	Numero de polinomios.
numberOfCoeffs	Int	Número de coeficientes en cada polinomio.
timeIntervals	VectorFloatType	Vector de intervalos de tiempo.
x_coeffs	MatrixFloatType	Matriz de coeficientes de las posiciones en el eje x.
y_coeffs	MatrixFloatType	Matriz de coeficientes de las posiciones en el eje y.
z_coeffs	MatrixFloatType	Matriz de coeficientes de las posiciones en el eje z.
vx_coeffs	MatrixFloatType	Matriz de coeficientes de la velocidad en el eje x.
vy_coeffs	MatrixFloatType	Matriz de coeficientes de la velocidad en el eje y.
vz_coeffs	MatrixFloatType	Matriz de coeficientes de la velocidad en el eje z.

BodyCoeffs Tipo: Función

Definición de la función:

BodyCoeffs(
info_file_path::String, file_path::String, body::Union{Int, String},
tspan::Tuple{Float64, Float64})

Descripción:

Constructora del objeto, BodyCoeffs.

Argumentos de entrada:

info_file_path	Ruta del fichero de información.
file_path	Ruta del fichero de coeficientes.
body	ID o Nombre del cuerpo.
tspan	Intervalo de tiempo al que pertenecen los polinomios.

Argumentos de salida:

bc	Objeto de coeficientes del cuerpo especificado

BodyCoeffs Tipo: Función

Definición de la función:

body_coeffs::BodyCoeffs(t::Float64, code::Int)

Descripción:

Método del objeto que permite calcular las coordenadas y/o las velocidades del cuerpo especificado en el instante de tiempo especificado.

Argumentos de entrada:

t	Instante de cuerpo en el que se quieren conocer las coordenadas de posición y/o velocidad del cuerpo.
code	Integer que indica los parámetros de salida. Por defecto code = 3. - code = 1: Devuelve el vector de las coordenadas de la posición. - code = 2: Devuelve el vector de las coordenadas de la velocidad. - code = 3: Devuelve el vector de las coordenadas de la posición y la velocidad.

Argumentos de salida:

res	Vector con las coordenadas y/o las velocidades del cuerpo especificado en el instante de tiempo especificado.

---

create_coeffs_file Tipo: Función

Definición de la función:

```
create_coeffs_file(
    coeffs_info_file_path::String, coeffs_file_path::String,
    body_vec::Union{Vector{Int}, Vector{String}},
    time_interval_vec::Vector{Tuple{Float64, Float64}}, header_file_path::String,
    time_file_path::String)
```

Descripción:

Genera dos ficheros, un fichero CSV con los coeficientes y uno JSON con los datos de los coeficientes

Argumentos de entrada:

coeffs_info_file_path	Ruta del fichero de información que se quiere crear.
coeffs_file_path	Ruta del fichero de coeficientes que se quiere crear.
body_vec	Vector de IDs o de nombres de los cuerpos de los que se quieren obtener los coeficientes.
time_interval_vec	Vector de intervalos de tiempo.
header_file_path	Ruta del fichero de información existente.
time_file_path:	Ruta del fichero de intervalos de tiempo.

Precondiciones:

La longitud del vector de cuerpos debe ser igual a la longitud del vector de intervalos de tiempos.
El fichero de información (header) debe ser un fichero JSON con un formato específico.
El fichero de intervalos de tiempo debe ser un fichero CSV de una única columna.

---

eval_coeffs_file Tipo: Función

Definición de la función:

```
eval_coeffs_file(
    coeffs_info_file_path::String, coeffs_file_path::String, body::Union{Int, String},
    time_vector::Vector{Float64})
```

Descripción:

Evalúa los coeficientes que se encuentran en un fichero especificado en los intervalos de tiempo especificados.

Argumentos de entrada:

coeffs_info_file_path	Ruta del fichero de información.
coeffs_file_path	Fichero de coeficientes.
body	Cuerpo del que se quieren evaluar los coeficientes.
time_vector	Vector de instantes en los que se quiere evaluar los coeficientes.

Argumentos de salida:

x	Vector de posiciones en el eje x.
y	Vector de posiciones en el eje y.
z	Vector de posiciones en el eje z.
vx	Vector de velocidad en el eje x.
vy	Vector de velocidad en el eje y.
vz	Vector de velocidad en el eje z.

Precondiciones:

El fichero de información debe ser un fichero JSON y tiene que estar en un formato específico.

---

generate_coeffs Tipo: Función

Definición de la función:

```
generate_coeffs(
    initial_date::Float64, final_date::Float64, coeffs_info::Tuple{Int, Int, Int},
    time_file_path::String)
```

Descripción:

Genera los coeficientes de Chebyshev entre dos fechas dadas para un cuerpo especificado.

Argumentos de entrada:

initial_date	Fecha inicial, perteneciente al primer intervalo (en formato ET).
final_date	Fecha final, perteneciente al último intervalo (en formato ET).
coeffs_info	Una tupla con la ID del cuerpo del que queremos obtener las Efemérides, número de coeficientes que queremos en nuestros polinomios y número de polinomios que queremos por intervalo.
time_file_path	Ruta del fichero de fechas.

Argumentos de salida:

time_vec	Vector de fechas (en formato ET).
x	Matriz de polinomios interpoladores de las posiciones del eje x. Cada fila es un polinomio, hay tantas columnas como coeficientes.
y	Matriz de polinomios interpoladores de las posiciones del eje y. Cada fila es un polinomio, hay tantas columnas como coeficientes.
z	Matriz de polinomios interpoladores de las posiciones del eje z. Cada fila es un polinomio, hay tantas columnas como coeficientes.
vx	Matriz de polinomios interpoladores de la velocidad en el eje x. Cada fila es un polinomio, hay tantas columnas como coeficientes.
vy	Matriz de polinomios interpoladores de la velocidad en el eje x. Cada fila es un polinomio, hay tantas columnas como coeficientes.
vz	Matriz de polinomios interpoladores de la velocidad en el eje x. Cada fila es un polinomio, hay tantas columnas como coeficientes.

Precondiciones:

initial_date $<$ final_date
El fichero de intervalos de tiempo debe ser un fichero CSV de una única columna.

---

generate_files Tipo: Función

Definición de la función:

generate_files(path::String)

Descripción:

Genera los ficheros necesarios para el uso de algunas funciones del paquete.

Argumentos de entrada:

path	Ruta en la que se quieren guardar los ficheros.

---

generate_subfile Tipo: Función

Definición de la función:

```
generate_subfile(
  info_subfile_path::String, subfile_path::String, body_vec::Union{Vector{Int},
  Vector{String}}, time_interval_vec::Vector{Tuple{Float64, Float64}},
  info_main_file_path::String, main_file_path::String)
```

Descripción:

Partiendo de un fichero genera otro con los cuerpos indicados en los intervalos de tiempo indicados.

Argumentos de entrada:

info_subfile_path	Ruta del nuevo fichero de información.
subfile_path	Ruta del nuevo fichero de coeficientes.
body_vec	Vector de IDs de los cuerpos o de los nombres de los cuerpo.
time_interval_vec	Vector de intervalos de tiempo.
info_main_file_path	Ruta del fichero de información original.
main_file_path	Ruta del fichero de coeficientes original.