



AGNI B3V: SISTEMA DE SIMULACIÓN TÉRMICA SATELITAL PARALELIZADO EN GPU

TRABAJO PROFESIONAL DE INGENIERÍA EN INFORMÁTICA

FACULTAD DE INGENIERÍA
UNIVERSIDAD DE BUENOS AIRES

Nombre	Padrón	Correo electrónico
Barreneche, Franco	102205	fbarreneche@fi.uba.ar
Belinche, Gianluca Ernesto	102674	gbelinche@fi.uba.ar
Botta, Guido Tomás	102103	gbotta@fi.uba.ar
Ventura, Julián	102391	jventura@fi.uba.ar

Febrero 2024

Índice

1. Introducción	3
2. Objetivos	4
3. Análisis	5
3.1. Modelo teórico	5
3.1.1. Transferencia de calor	5
3.1.1.1. Transferencia de calor debido a conducción	5
3.1.1.2. Transferencia de calor debido a radiación	6
3.1.2. Fuentes de calor en sistemas espaciales	7
3.1.2.1. Radiación solar	7
3.1.2.2. Albedo	7
3.1.2.3. Radiación infrarroja terrestre	8
3.1.2.4. Transferencia de calor en el satélite	8
3.2. Modelo numérico	8
3.2.1. Mallado FEM	9
3.2.2. Factores de vista	10
3.3. Herramientas y bibliotecas disponibles	11
3.3.1. Modelado 3D	11
3.3.2. Mallado para análisis	11
3.3.3. Cálculos orbitales	11
3.3.4. Cómputo en GPU	12
3.3.5. Trazado de rayos	12
3.3.6. Postprocesado	12
4. Desarrollo	13
4.1. Supuestos	13
4.2. Workflow general	13
4.2.1. Interfaz gráfica de usuario	14
4.2.2. Instalador	14
4.2.3. Documentación	15
4.3. Modelado	15
4.3.1. Modelo tridimensional	15
4.3.2. Addon de FreeCAD	16
4.3.3. Cálculos orbitales	17
4.4. Preprocesado	17
4.4.1. Modelo numérico	19
4.4.2. Interacción elemento-elemento	19
4.4.3. Interacción Sol-elemento	20
4.4.4. Interacción Tierra-elemento	21
4.5. Solver	25
4.5.1. Modelo FEM	25
4.5.1.1. Ecuación de calor	25
4.5.1.2. Construcción de matrices locales	26

4.5.1.3. Construcción de matrices globales	28
4.5.1.4. Expansión a tres dimensiones	31
4.5.1.5. Incorporación de radiación	32
4.5.1.6. Integración de las partes	34
4.5.2. Resolución del sistema de ecuaciones	35
4.5.3. Resolución en CPU	37
4.5.4. Resolución en GPU	37
4.6. Postprocesado	41
5. Resultados	42
5.1. Comparaciones	42
5.1.1. Conducción	42
5.1.1.1. Prueba 1: Conducción simple en cubo	42
5.1.1.2. Prueba 2: Conducción en cubo con flujo constante	45
5.1.1.3. Prueba 3: Conducción simple en cubo de dos materiales	47
5.1.2. Radiación	50
5.1.2.1. Prueba 1: Placas Paralelas	50
5.1.3. Conclusiones	52
5.2. Conservación de la Energía	52
5.3. Convergencia	54
5.3.1. Convergencia en Tiempo	56
5.3.2. Convergencia en Malla	57
5.4. Tiempo de Ejecución	59
5.5. Profiling	63
5.6. Consumo de Memoria	66
5.7. Validación con Expertos	70
6. Conclusión	71
7. Bibliografía	72
8. Anexos	74
8.1. Resultados Extendidos	74
8.1.1. Conducción	74
8.1.1.1. Prueba 1: Conducción simple en cubo	74
8.1.1.2. Prueba 3: Conducción simple en cubo de dos materiales	77
8.1.1.3. Prueba 4: Conducción simple en cubo de aluminio y cobre	80
8.1.1.4. Prueba 5: Conducción en cubo con flujo constante y dos materiales .	86
8.1.2. Radiación	91
8.1.2.1. Prueba 2: Placas paralelas distinta emisividad	91
8.1.2.2. Prueba 3: Placas paralelas distinta emisividad	94
8.1.3. Convergencia	97
8.1.3.1. Convergencia en Tiempo	98
8.1.3.2. Convergencia en Malla	101

1. Introducción

En el ámbito de la industria aeroespacial, entender los fenómenos que afectan a un satélite en órbita es tan valioso como comprender las condiciones a las que se ve sujeto durante su lanzamiento. Una importante herramienta es el análisis térmico, para el que debe contemplarse la posición en el tiempo de los cuerpos celestes, la propagación de los fotones que transmiten la energía y el modo en que la geometría y materiales del satélite interactúan con los mismos.

Dada la complejidad del modelo, investigadores del entorno académico y la industria suelen optar entre dos alternativas:

- Realizar modelos físicos y extrapolar los resultados, lo que requiere equipo especializado para aproximar las condiciones extremas del espacio (cuasi-vacío, temperaturas cercanas a 0K, etc.) y la construcción de un modelo satelital con materiales comparables.
- Acudir a simulaciones por software, que necesitan de una capacitación adicional en el manejo de herramientas digitales y pueden presentar márgenes de incertidumbre afectados por el error numérico, pero admiten configurar y modificar las condiciones experimentales con mayor velocidad y simpleza.

Grandes empresas o agencias gubernamentales pueden permitirse asumir los costos de explorar ambas vías y compararlas. No obstante, otros actores se enfrentarán a obstáculos al abordar una versión acotada de la segunda. En la actualidad, contados productos ofrecen una simulación completa de los fenómenos descritos. Todos ellos de código cerrado, escasa documentación y licencias privativas para pequeñas empresas y ámbitos académicos. Existen herramientas abiertas y gratuitas que resuelven problemas de conducción, radiación y trayectorias orbitales por separado, aunque acoplar dichas simulaciones no es una tarea sencilla, ni desde la interacción de las piezas de software, ni del control del error numérico.

Al indagar en el área fue posible contactar con un experto que aún trabaja en la industria y que, además de confirmar los hallazgos anteriores, ha comentado su experiencia con el software comercial. Si bien estos cubren las necesidades funcionales del problema, existen indicios de que la experiencia de usuario y el aprovechamiento de los recursos computacionales pueden mejorarse. Puntualmente, no parece que se estén explotando las capacidades de las tarjetas gráficas, que han beneficiado distintas áreas, como la de simulación de fluidos.

En síntesis, se ha detectado la necesidad de un software de simulación numérica térmica satelital gratuito que cubra los requerimientos de pequeños grupos de investigación de la industria y el ámbito académico, manteniendo un buen nivel de usabilidad. Así como la necesidad de explorar la posibilidad de distribuir en GPU partes de la simulación numérica.

2. Objetivos

El presente trabajo tiene como propósito brindar una solución de software de licencia gratuita, cómoda y capaz de explotar las ventajas de la computación heterogénea para ingenieros aeroespaciales, científicos y demás profesionales que trabajan en la planificación de misiones satelitales terrestres. Puede resumirse en los siguientes objetivos:

- Desarrollar un sistema que permita a un usuario experto en la materia realizar simulaciones térmicas satelitales básicas, desde el modelado hasta la visualización de resultados.
- Ofrecer un conjunto de características que respondan a las necesidades concretas de los usuarios esperados. Esto es, ajustar el sistema de forma tal que su uso sea accesible para un investigador, para lo que se tomará como referencia las indicaciones del experto contactado.
- Plantear un modelo simple, numéricamente estable, físicamente fundado y capaz de alcanzar resultados con un margen de incertidumbre aceptable para el dominio del problema.
- Diseñar un sistema que escale verticalmente, junto con la capacidad de los recursos computacionales, explorando especialmente la posibilidad de distribuir cómputo en GPU. Particularmente, lograr un speed up con respecto al sistema que se ejecuta exclusivamente en CPU.

3. Análisis

3.1. Modelo teórico

Las interacciones físicas que afectan a un satélite en órbita, en lo que respecta a los cambios de temperatura, se manifiestan a través de procesos como la transferencia de calor por conducción y radiación. Estos fenómenos se complejizan al tener en cuenta las fuentes de calor intrínsecas a los sistemas espaciales, que inciden de manera significativa en la estabilidad térmica del satélite.

3.1.1. Transferencia de calor

En líneas generales, la transferencia de calor puede explicarse a través de tres fenómenos distintos: la conducción, la convección y la radiación.

Dado que la densidad del aire a altitudes muy elevadas es extremadamente baja, a partir de órbitas terrestres bajas (LEO) en adelante, la transferencia de calor es, prácticamente en su totalidad, debida a la conducción y radiación. Es por esto que es posible omitir el fenómeno de convección en el modelo.

3.1.1.1 Transferencia de calor debido a conducción

La transferencia de calor puede ocurrir dentro de un material o entre dos o más cuerpos en contacto. Está dada por la Ley de Fourier, en una dimensión:

$$f_x'' = -k \frac{dT}{dx}. \quad (1)$$

Donde:

- q es la tasa de flujo de calor [$\frac{W}{m^2}$],
- k es la conductividad térmica [$\frac{W}{m K}$] del material,
- $\frac{dT}{dx}$ es la diferencia de temperatura a lo largo de la longitud.

En el caso más general, la ecuación se expresa en forma diferencial:

$$c\rho \frac{\partial T}{\partial t} = k \left(\frac{\partial^2 T}{\partial x^2} + \frac{\partial^2 T}{\partial y^2} + \frac{\partial^2 T}{\partial z^2} \right) + q_v. \quad (2)$$

Donde:

- c es el calor específico [$\frac{J}{kg K}$],
- ρ la densidad [$\frac{kg}{m^3}$],
- q_v el calor generado [$\frac{W}{m^3}$].

3.1.1.2 Transferencia de calor debido a radiación

La transferencia de calor por radiación ocurre entre dos o más superficies a través de ondas electromagnéticas. Depende de la temperatura y del revestimiento de la superficie radiante. El tipo de emisor más eficiente es conocido como cuerpo negro. La radiación que éste emite por unidad de área a una temperatura dada sigue la Ley de Stefan-Boltzmann:

$$E = \sigma T^4 . \quad (3)$$

Donde:

- σ es la constante de Stefan-Boltzmann ($5,63 \times 10^{-8} \frac{\text{W}}{\text{m}^2 \text{K}^4}$),
- T es la temperatura del cuerpo negro [K].

Multiplicando por el área radiante A^r , se obtendrá el calor absoluto emitido por el cuerpo negro $E_r[\text{W}]$:

$$E_r = A^r \sigma T^4 . \quad (4)$$

A un cuerpo que no irradia con la misma eficiencia que un cuerpo negro se lo conoce como cuerpo gris. La radiación emitida por un cuerpo gris hace uso de la ecuación de cuerpo negro, corregida por un factor de emisividad:

$$E_{\text{Gris}} = \epsilon A^r \sigma T^4 . \quad (5)$$

Donde ϵ es la emisividad del cuerpo gris.

El calor transferido desde un cuerpo gris 1, hacia otro cuerpo gris 2 por radiación puede expresarse de la siguiente forma:

$$f_{1,2} = \sigma \epsilon_1 \alpha_2 F_{1,2} A_1 T_1^4 . \quad (6)$$

Donde:

- ϵ_1 es la emisividad de la superficie del cuerpo 1,
- α_2 es la absorbividad de la superficie del cuerpo 2,
- $F_{1,2}$ es el factor de vista entre la superficie del cuerpo 1 y el cuerpo 2,
- A_1 es el área de la superficie del cuerpo 1,
- T_1 es la temperatura del cuerpo 1.

El factor de vista de una superficie 1 hacia una superficie 2, $F_{1,2}$, se define como la porción del campo de visión de la superficie 1 que es ocupada por la superficie 2. Dado que se trata de un porcentaje, este toma valores entre cero y uno.

Así como el factor de emisividad corrige la diferencia de emisión de un cuerpo gris frente a la emisión teórica de un cuerpo negro, el factor de absorbividad realiza la misma corrección sobre la cantidad de energía absorbida por la superficie de un cuerpo gris. Dado que ambos factores suelen ser similares (para una misma superficie), se suelen simplificar las ecuaciones reemplazando la emisividad por la absorbividad. Es importante aclarar que tanto la absorbividad como la emisividad dependen no solo de la superficie en cuestión, sino también de la frecuencia de la radiación emitida o recibida.

3.1.2. Fuentes de calor en sistemas espaciales

En el contexto de un sistema espacial, las fuentes de energía se derivan comúnmente de la radiación solar, la radiación de albedo y la radiación infrarroja terrestre, contribuyendo así al equilibrio térmico del sistema.

3.1.2.1 Radiación solar

Es la principal fuente de calor y puede ser considerada constante. Suele rondar entre $1322 \frac{\text{W}}{\text{m}^2}$ y $1414 \frac{\text{W}}{\text{m}^2}$.

La radiación proveniente desde el Sol e incidente sobre una superficie se encuentra en función del flujo solar $S [\frac{\text{W}}{\text{m}^2}]$ y de la orientación de esta con respecto al Sol. Debido a la gran distancia con el Sol, se puede tomar la suposición de que la radiación está dada por rayos paralelos.

El calor recibido por una superficie i debido a la radiación proveniente del Sol estará dado por:

$$f_{\text{Sun}} = \alpha_i^{\text{Sun}} F_{\text{Sun},i} A_i S. \quad (7)$$

α_i^{Sun} considera la absorbividad de la superficie i bajo las frecuencias de las ondas electromagnéticas emitidas por el Sol.

3.1.2.2 Albedo

El albedo refiere a la radiación solar reflejada desde la superficie terrestre. Se suele expresar como la fracción de radiación solar incidente que es reflejada hacia el espacio:

$$f = A_f S. \quad (8)$$

Donde:

- A_f es el factor de albedo,
- S es la constante solar $[\frac{\text{W}}{\text{m}^2}]$.

El albedo promedio de la Tierra es de aproximadamente 0,3^[1].

De esta forma, es posible expresar el calor recibido por una superficie i debido al albedo de la siguiente forma:

$$f_{\text{Albedo}} = \alpha_i^{\text{Sun}} F_{\text{Earth},i} A_i A_{f_i} S. \quad (9)$$

3.1.2.3 Radiación infrarroja terrestre

La radiación Infrarroja (IR) terrestre refiere a la radiación emitida por la Tierra por el simple hecho de tratarse de un cuerpo con temperatura mayor a 0 K. Por lo general se considera que esta radiación es constante a lo largo de su superficie y se la suele tomar con un valor cercano a $235 \frac{\text{W}}{\text{m}^2}$.

El calor recibido por una superficie S debido a radiación IR estará dado por:

$$f_{\text{IR}} = \alpha_i^{\text{IR}} F_{\text{Earth},i} E_{\text{IR}}. \quad (10)$$

Donde E_{IR} es la constante de IR terrestre $[\frac{\text{W}}{\text{m}^2}]$.

3.1.2.4 Transferencia de calor en el satélite

Además de recibir energía por radiación del Sol y la Tierra, un satélite emite hacia su alrededor.

Tratándose de un cuerpo gris, el flujo de calor emitido por una superficie del satélite estará dado por:

$$f_{\text{Lost}} = \alpha^{\text{IR}} A^r \sigma T^4. \quad (11)$$

Parte de esta energía será perdida en el espacio y otra parte será absorbida por otras superficies del propio satélite. El flujo recibido por una superficie j del satélite, debido a la emisión de una superficie i , estará dado por:

$$f_{i,j}^{\text{Elem}} = \sigma \alpha_i^{\text{IR}} \alpha_j^{\text{IR}} F_{i,j} A_i T_i^4. \quad (12)$$

3.2. Modelo numérico

La solución al problema teórico no siempre puede abordarse analíticamente y debe resolverse a través de métodos numéricos. Los métodos numéricos más utilizados son el método de diferencias finitas (FDM), el método de volúmenes finitos (FVM) y el método de elementos finitos (FEM). Todos tienen el objetivo de discretizar el dominio y aproximar la solución reemplazando el sistema de ecuaciones diferenciales.

- FDM discretiza el dominio del problema en una malla regular, reemplazando derivadas con aproximaciones de diferencias finitas. Es comúnmente utilizado para problemas en geometrías regulares.
- FVM divide el dominio en volúmenes de control y aproxima la solución integrando las ecuaciones gobernantes sobre dichos volúmenes. Suele ser utilizado para problemas de fluidodinámica computacional.
- FEM discretiza el dominio en elementos finitos y utiliza funciones de forma para aproximar las soluciones dentro de cada elemento. Se utiliza para diversos problemas de ingeniería estructural, térmica, entre otros. FEM es ampliamente utilizado debido a la precisión física y a su flexibilidad para manejar geometrías complejas.

3.2.1. Mallado FEM

En un mallado FEM, la estructura de un cuerpo se construye a partir de elementos. En dos dimensiones pueden utilizarse triángulos o cuadriláteros, mientras que para tres dimensiones tetraedros y hexaedros son una elección habitual.

A esto se le suma la posibilidad de utilizar elementos de distinto orden (Fig. 1). Un mayor orden en los elementos permite una mayor deformación del mallado a lo largo del tiempo de la simulación. Este punto no es de particular interés en la simulación térmica satelital cuando se trabaja entre rangos de temperatura donde la estructura no se deforma significativamente. Además, un mayor orden en el mallado implica un mayor costo computacional.

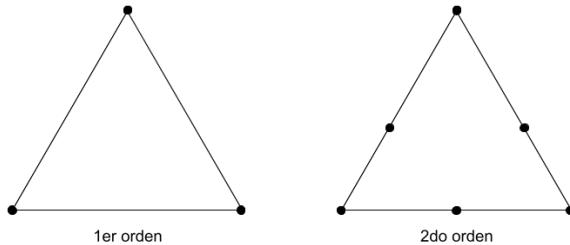


Figura 1: Órdenes en un triángulo bidimensional

Es importante que un mallado FEM sea bueno para no introducir mayor error numérico. Para ello, hay ciertos puntos a tener en cuenta:

- Las transiciones de densidades de elementos en la malla no deben ser abruptas.
- Las transiciones de densidad de elementos deben estar localizadas lejos de las regiones de mayor interés.
- Es recomendable no tener grandes diferencias entre los elementos de menor y mayor tamaño. Sobre todo si son regiones cercanas.

3.2.2. Factores de vista

El modelo tridimensional del satélite, junto con los materiales asignados a cada polígono que lo compone, son información suficiente para el cálculo de la transmisión de calor por conducción. Estos datos se encuentran estructurados de forma tal que es preciso y eficiente considerar solo el aporte de la vecindad para cualquier punto en un tiempo determinado.

Para la transmisión de calor por radiación deben contemplarse además la posición de la Tierra y el Sol, así como el modo en que las distintas partes del satélite eclipsan a otras. La energía recibida en un punto debido a radiación es el resultado de integrar los rayos que fueron emitidos/reflejados por otros cuerpos u elementos del mismo cuerpo.

El factor de vista es un concepto central en muchos de los modelos de cálculo de radiación y puede entenderse coloquialmente como la proporción del campo de vista que cubre una superficie respecto a otra.

El método Montecarlo permite definir y aproximar factores de vista con mayor precisión^[2]. En él se seleccionan puntos del elemento emisor y direcciones al azar en la que se emiten rayos, se toma registro de los elementos impactados y luego se computa el factor de vista como la razón entre los rayos impactados y el total de rayos emitidos. Dependiendo del propósito de la simulación, se pueden emitir rayos desde un elemento en todas las direcciones, como si fuese una placa, o solo en dirección de la normal, suponiendo que es parte de un cuerpo sólido. También pueden introducirse reflexiones reemitiendo rayos desde las posiciones de impacto.

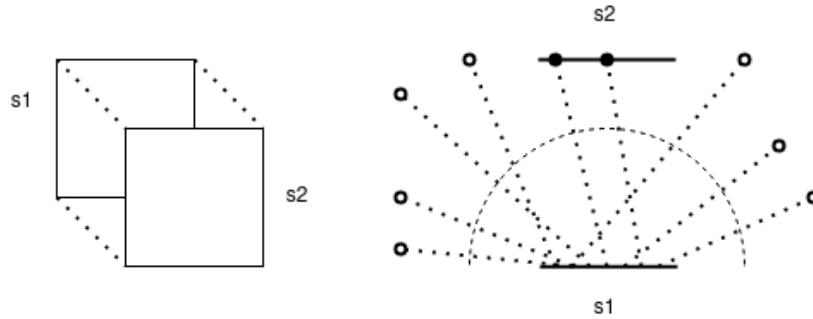


Figura 2: Ejemplo de Factor de Vista

En el ejemplo (Fig. 2) se disparan diez rayos desde s_1 , de los cuales solo dos impactan s_2 , por lo que el factor de vista entre s_1 y s_2 se aproximaría como $\frac{2}{10}$. Notar que si s_1 y s_2 no tuvieran la misma área, el factor de vista s_1s_2 diferiría del factor de vista s_2s_1 . Si se tratases de dos cuerpos negros, la energía que s_1 aporta a s_2 podría calcularse como el producto entre el total de energía emitida por s_1 y el factor de vista s_1s_2 .

El modelo puede extenderse para considerar el medio de transmisión y características de los materiales con los que se construyen los elementos. Sin embargo, el raycasting es una operación costosa, por lo que el cómputo es más eficiente cuando se expresan tales fenómenos en función de los factores de vista y no al contrario.

3.3. Herramientas y bibliotecas disponibles

3.3.1. Modelado 3D

La construcción de mallas tridimensionales es un campo que lleva décadas en desarrollo y en el que confluyen dos tendencias: la representación visual y el diseño industrial. En la primera de ellas priman las propiedades vinculadas a la interacción con la luz y la estética, se cuida la interfaz de usuario para ser accesible a artistas y se soportan formatos que incluyen materiales (texturas, mapa de normales, color, etc) y compresión, como ser obj, fbx o gltf. Aquí se encuentran muchas opciones comerciales, como ser ZBrush^[3], Cinema4D^[4] o Maya^[5] y dentro de las alternativas gratuitas y abiertas, Blender^[6] se destaca por su grado de madurez, documentación y comunidad. En la segunda tendencia el objetivo es el control preciso sobre las dimensiones y topología del modelo para su construcción y posterior uso en simulaciones, con interfaces paramétricas destinadas a un público familiarizado con la matemática. FreeCAD^[7] y OpenSCAD^[8] permiten construir mallas y combinarlas. Sketchup^[9] y la suite de herramientas AutoCAD^[10], que cubren la mayor parte del mercado, no se diferencian tanto en las características, sino por los formatos de archivo privativos que otros softwares de simulación requieren para funcionar dentro de un workflow razonable.

3.3.2. Mallado para análisis

Como se explicó anteriormente, un mallado irregular puede introducir significativos errores numéricos en simulaciones. Lo habitual en la industria es perfeccionar una malla preeexistente empleando herramientas como Gmsh^[11] o Netgen^[12]. Ambas soluciones abiertas, documentadas y ampliamente soportadas.

El principal obstáculo para su aplicación es el mapeo de propiedades adicionales a partes de los modelos, porque trabajan con formatos de archivos que se limitan a describir la topología de los modelos (stl en caso de Netgen, msh dentro de Gmsh). Es posible exportar una malla, transformarla en una malla FEM y reimporrtarla, pero al modificarse la cantidad de polígonos y su indexado no puede retenerse con facilidad los datos adicionales incorporados en etapas tempranas, como ser el material de una cara.

3.3.3. Cálculos orbitales

Dependiendo de la precisión requerida, los datos orbitales pueden ser estimados simplemente aplicando las ecuaciones de Newton a los cuerpos celestes más próximos y/o masivos. De todas formas, al ser este análisis un requerimiento común a todo estudio espacial, existen herramientas como 42^[13] o GMAT^[14]. Estas son dos herramientas gratuitas, ambas publicadas por la NASA y ampliamente utilizadas para la planificación general de misiones, ofreciendo información sobre la trayectoria y orientación de los cuerpos en el tiempo, desde diferentes sistemas de referencia y bajo distintos escenarios base.

La principal diferencia entre estas herramientas es la complejidad y profundidad que ofrecen. 42 es una herramienta muy completa, sin embargo, cuenta con una documentación limitada y su curva de aprendizaje es muy compleja. Por otro lado, GMAT está más orientada a un usuario final, ofreciendo una interfaz gráfica fácil de usar y muy intuitiva. Además, cuenta con una documentación completa y distintos ejemplos orientativos.

3.3.4. Cómputo en GPU

En el propio nombre de las unidades de procesamiento gráfico (graphical processing units, GPU) se representa su propósito original de acelerar operaciones matriciales asociadas a la representación en pantallas. Recientemente, se ha profundizado en tecnologías como blockchain o machine learning la tendencia de reformular problemas no visuales en naturaleza con tal de poder aprovechar la paralelización en GPU para conseguir disminuir el tiempo de cómputo. Buena parte de la bibliografía al respecto gravita hacia CUDA^[15], una plataforma de computación en paralelo y herramientas auxiliares desarrollado por Nvidia y exclusivamente soportado por sus productos. También es posible hacer uso de OpenCL^[16], un estándar para computación heterogénea, en el que se enmarca la paralelización en GPU, que fue desarrollado por el grupo Kronos, responsable de especificar otros estándares libres para aplicaciones 3D, realidad virtual y machine learning como Vulkan^[17], OpenGL^[18] y Collada^[19].

3.3.5. Trazado de rayos

El trazado de rayos ha evolucionado conjuntamente con el modelado tridimensional y, por tanto, es realizada por las mismas herramientas, aunque usualmente restringida a la representación visual de la escena modelada. En motores de videojuegos orientados a tres dimensiones, el trazado de rayos para obtener información de la distancia y visibilidad de los objetos es una característica básica común. El problema con ambos grupos es que si ofrecen APIs, no suelen exponerse por fuera del entorno de la propia plataforma. Existen numerosas bibliotecas que solucionan el trazado de rayos, pero es importante medir la eficiencia con la que lo hacen y el modo en que se integra el modelo tridimensional. Entre ellas destacan Open3D^[20], con funciones dedicadas al análisis de las mallas; three.js^[21], wrapper minimalista de WebGL que exhibe benchmarks prometedoras; y trimesh^[22], que puede delegar el raycast a Intel Embree, biblioteca que se ejecuta en GPU bajo arquitecturas de esa empresa y que aun en CPU tuvo un buen rendimiento.

3.3.6. Postprocesado

Casi todo software de simulación ofrece algún tipo de resumen, plot o visualización interactiva para interpretar los datos generados. Para las representaciones bidimensionales suelen emplearse directamente hojas de cálculo como Microsoft Excel^[23] o LibreOffice Calc^[24], mientras que para visualizaciones tridimensionales existen herramientas especializadas, como ser Paraview^[25] o Mayavi^[26]. A ellas se suman incontables bibliotecas que permiten procesar datos y generar representaciones ajustadas a las necesidades del usuario. Algunas de ellas son Matplotlib^[27], Pandas^[28] u open3D.

4. Desarrollo

4.1. Supuestos

Se tomaron los siguientes supuestos acerca de las características del problema a resolver:

- **Orientación del Satélite:** Los tipos de orientación satelital más usuales en la industria son Sun Pointing, Earth Pointing, Inertial Pointing y Barbecue Spin. Se soportó exclusivamente la aptitud Sun Pointing, en donde el satélite siempre se encuentra orientado de la misma forma en dirección al Sol.
- **Modelo de Caparazón:** Se utilizó un modelo 2.5D. Esto quiere decir que se modeló a los satélites como objetos huecos, en donde la propagación del calor se realiza únicamente a través de la frontera. Esta simplificación es comúnmente utilizada en la industria y cubre la gran mayoría de los casos de estudio.
- **Convección:** Debido a que en el ambiente sobre el cual se lleva a cabo la simulación no existe la presencia de fluidos, se descarta por completo la modelación del fenómeno de transmisión de calor por convección, incluyendo el interior del propio satélite.
- **Órbitas:** En órbitas de gran excentricidad, las fluctuaciones en la distancia entre el satélite y la Tierra complejizan el modelo numérico. Se decidió restringir el análisis a órbitas circulares con el objetivo de simplificar el modelo.
- **Penumbra:** Se despreciaron los fenómenos asociados a la penumbra, el tiempo de transición entre que el satélite se encuentra completamente expuesto al Sol y se ve totalmente eclipsado por la Tierra. La conducción térmica es un fenómeno muy lento y, para una órbita ecuatorial de ensayo a 7000 km de altura, 50 grados de inclinación (respecto a la recta de incidencia de los rayos del Sol) y tiempo de eclipse de 2062 segundos, menos de 20 segundos transcurren en penumbra.
- **Propiedades de Materiales:** Se supusieron materiales isotrópicos con propiedades físicas constantes a lo largo del tiempo.

4.2. Workflow general

Una de las decisiones más importantes que se tomó tras la investigación y relevamiento inicial fue la de dividir el sistema en partes separadas con el fin de poder emplear libremente las tecnologías idóneas para cada tarea y permitir que estos módulos evolucionen independientemente o bien sean reemplazados por software de terceros con facilidad. La performance se vería favorecida por dicha táctica, a costa de la mayor complejidad en la comunicación entre las etapas y de la potencial disminución de usabilidad frente al usuario si no se ocultaba o al menos homogeneizaba su interacción con los mismos. Estas ventajas fueron evidentes en la separación entre el preprocessor y el solver, en donde se asignó al primero las características más inestables y al segundo los cálculos del modelo bien definidos, sensibles a ser optimizados. Se mantuvo el siguiente workflow a lo largo de todo el proyecto:

- **Modelado:** Construcción del modelo tridimensional; mallado FEM; asignación de materiales y condiciones de contorno; asignación de propiedades globales; y estimación de órbita.

- **Preprocesado:** Cálculo de factores de vista. Adaptador entre la salida (o salidas) de la etapa previa y la siguiente.
- **Solver:** Cómputo eficaz de la simulación térmica y escritura de resultados.
- **Postprocesado:** Presentación ordenada de resultados.

4.2.1. Interfaz gráfica de usuario

Para facilitar la realización de los distintos pasos del workflow, se desarrolló una interfaz gráfica que permite la utilización de las distintas herramientas de manera visual y desde un único punto (Fig. 3).

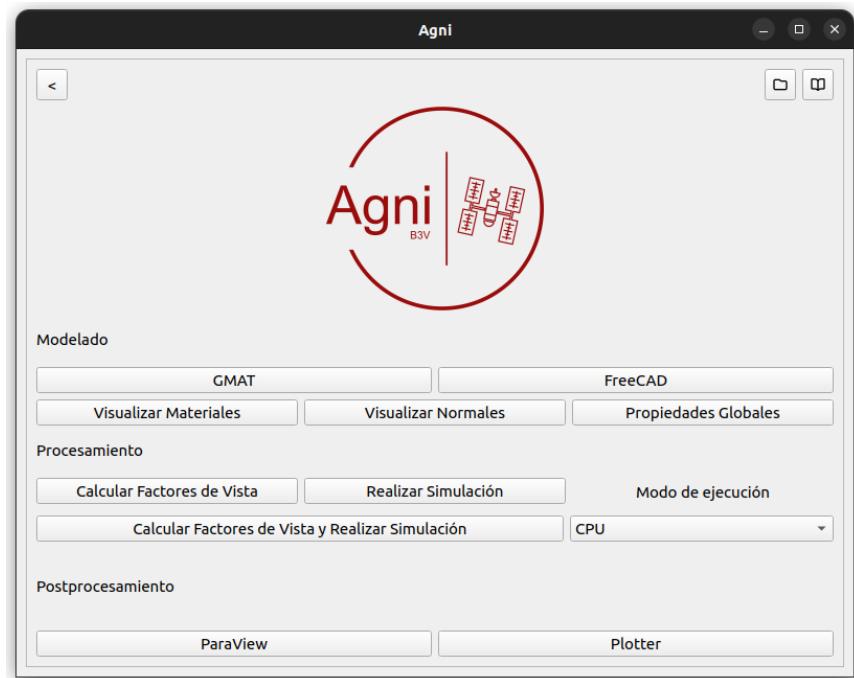


Figura 3: Interfaz Gráfica de Usuario

4.2.2. Instalador

Se implementó un instalador para distribuciones Ubuntu 22.04, encargado de descargar requerimientos y programas de terceros, organizarlos en la carpeta del proyecto y configurar el addon y la interfaz de usuario para que Agni B3V pueda ser obtenido y ejecutado en pocos pasos por un usuario conocimientos técnicos básicos.

4.2.3. Documentación

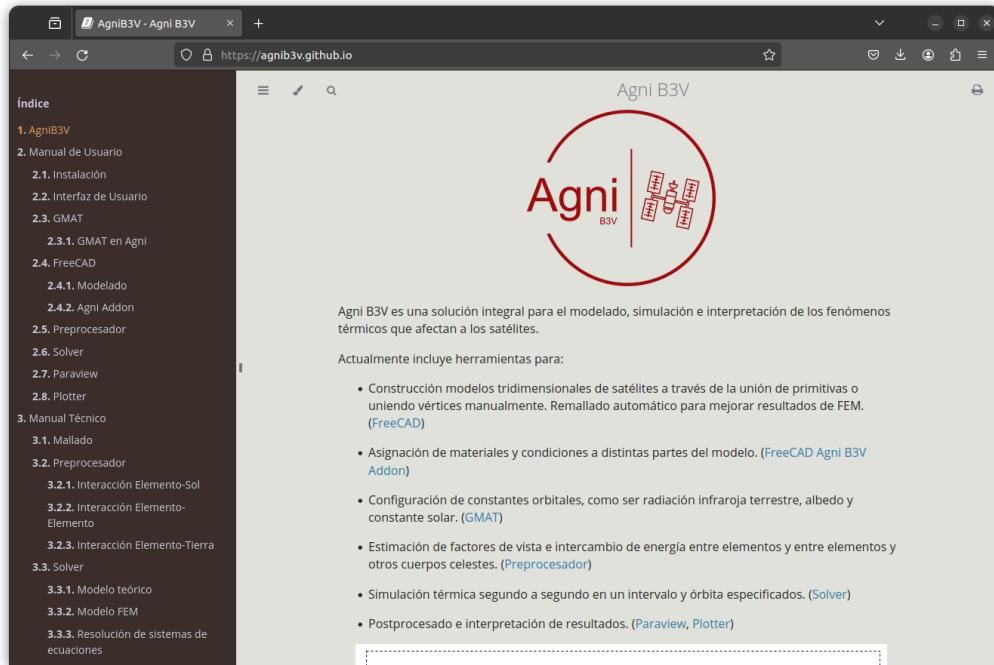


Figura 4: Página Web de Documentación

Se alojó en Github una página web en la dirección <https://agnib3v.github.io>. En la misma se escribió un manual de usuario para la correcta instalación y uso del programa, un manual técnico con los detalles de implementación.

4.3. Modelado

4.3.1. Modelo tridimensional

Al comienzo Blender se consideró la mejor opción para las primeras etapas. Como se explicó con anterioridad, es una herramienta de modelado muy completa, que además provee una API de trazado de rayos optimizada mediante BHTrees y con expectativas de soportar GPU en el futuro. También permite definir con facilidad interfaces, con lo que podría actuar como la cara visible del sistema, invocando al resto de las partes internamente.

Se alcanzó a desarrollar un prototipo de plugin que dado un modelo y un vector solar calculaba la visibilidad de los nodos hacia el Sol. No obstante, conforme progresaba el trabajo se hacían evidentes algunas falencias: asignar propiedades a los elementos no era sencillo, no se tenía un control preciso de las dimensiones del modelo 3D y no era claro como mantener el vínculo entre propiedades asignadas y la malla procesada por Gmsh o Netgen.

Estos obstáculos suscitaron la evaluación del uso de FreeCAD, que contaba con diversas ventajas:

- Es un modelador paramétrico 3D orientado al diseño de elementos mecánicos. Por lo que resulta una herramienta más adecuada para modelos de ingeniería que requieren de precisión en las dimensiones.
- Tiene integrado generadores de mallas FEM como Gmsh y Netgen.
- Presenta un sistema de creación y asignación de materiales, el cual se extendió para la asignación de condiciones de contorno.
- Dispone de una API para extender funcionalidades. Si bien al principio resultó muy compleja de utilizar, fue clave en el desarrollo del workbench que facilita la integración de FreeCAD con el resto de herramientas.

FreeCAD no ofrecía una solución interna para el trazado de rayos, por lo que debió resolverse en etapas posteriores.

4.3.2. Addon de FreeCAD

Se extendió FreeCAD a través de un addon, agregando un nuevo workbench (Fig. 5). Dentro de este, se le facilita al usuario distintas herramientas para agilizar el trabajo y permitir la integración con las siguientes herramientas. Este ofrece comandos que permiten:

- Crear un mallado FEM con elementos triangulares bidimensionales de primer orden, haciendo uso de la integración de FreeCAD con Gmsh.
- Asignar mayor o menor detalle a las regiones deseadas en el mallado FEM.
- Crear y asignar tanto materiales como condiciones de contorno a las distintas superficies del modelo.
- Modificar las propiedades globales del sistema. Que abarcan tanto propiedades físicas (factor de albedo, constante solar, etc), como propiedades propias de la simulación (paso del tiempo, cantidad de rayos para el procesamiento de factores de vista).
- Exportar el modelo en formato VTK y las propiedades globales, las propiedades de los materiales y las condiciones de contorno en formato json.

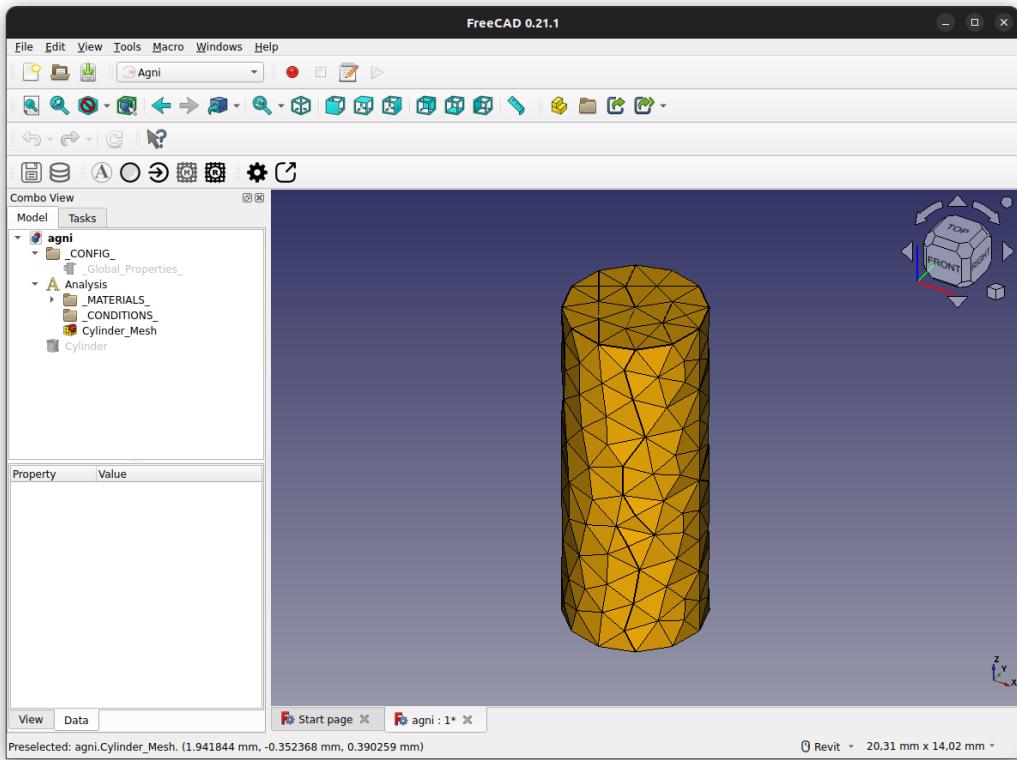


Figura 5: Addon de FreeCAD

4.3.3. Cálculos orbitales

Para los cálculos orbitales se optó por GMAT por sobre 42 debido a que el primero cuenta con mejor documentación, es más amigable con el usuario final, los archivos de salida son sencillos de interpretar y cuenta con un alcance suficiente para completar la simulación.

Se utilizaron dos archivos de salida:

- **ReportFile:** Incluye los datos de la órbita a lo largo del tiempo, como la posición del satélite y del Sol con respecto a la Tierra.
- **EclipseLocator:** Contiene la información del momento en el que transcurre un eclipse y su duración.

4.4. Preprocesado

El preprocesador fue ideado como un breve script de Python para adaptar archivos disímiles a un formato legible por el solver y acabó cubriendo todo lo referido al cálculo de factores de vista.

El primer prototipo se programó en javascript, utilizando WebGL a través de three.js. Se llegó a construir una herramienta simple que podía ejecutarse desde cualquier navegador web moderno, donde se cargaba un modelo, se definían propiedades y se las asignaba a un elemento pintando

con el cursor. El siguiente paso era implementar las funciones de raycast y exportar los resultados, pero en este momento se detectaron las restricciones a las operaciones de I/O que imponen los navegadores web. Éstas podían evitarse, por ejemplo, embebiendo la aplicación en Electron, pero esto devengaría en contra de la simpleza y portabilidad que motivó su elección. Cuando finalmente fue posible aprovechar el sistema de materiales de FreeCAD, se acabó por descartar esta alternativa.

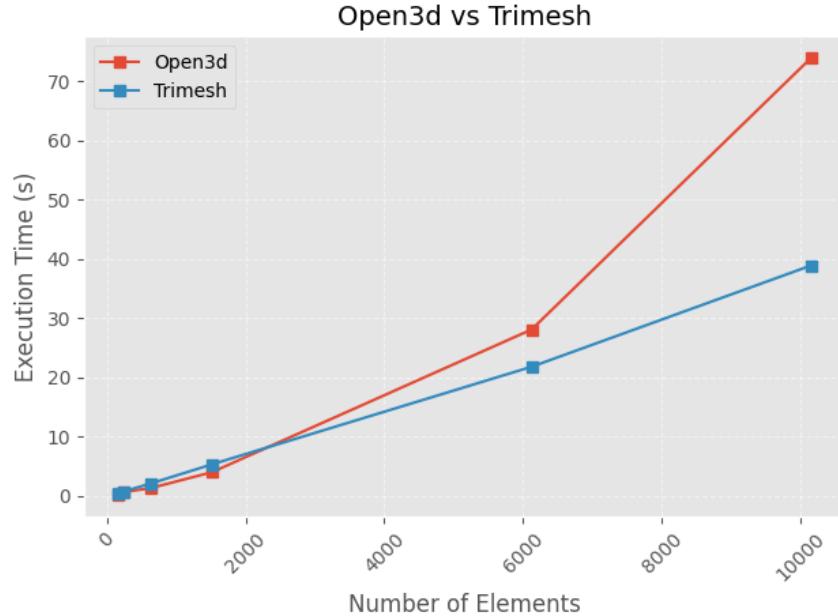


Figura 6: Benchmarks Open3D vs Trimesh

Para resolver el raycasting se evaluaron múltiples bibliotecas. Open3D y trimesh (usando Intel Embree) superaron con creces al resto de las opciones en tiempos de ejecución. Se realizaron benchmarks más finos para terminar de decidir cuál emplear.

Se enseña un escenario en donde las caras de un cubo se subdividen progresivamente y desde cada elemento se lanzan 5000 rayos (Fig. 6). Open3D es más veloz para una baja cantidad de elementos, pero al aumentar se aprecia una tendencia cuadrática, mientras que trimesh exhibe un comportamiento lineal y para 10000 elementos, que es el orden de un mallado típico, resuelve el raycasting en la mitad de tiempo. Se continuó trabajando con trimesh bajo el supuesto de que, como en el solver se realizarían operaciones de orden cuadrático y hasta cúbico, mientras se programase delegando las operaciones computacionalmente intensivas a numpy (que las resuelve desde un backend en C), el porcentaje del tiempo total que representaría el preprocesador sería pequeño.

Para la lectura de los datos orbitales de GMAT se implementó un parser propio. En GMAT no se tenía un control fino del espaciado de los puntos, por lo que era posible que se concentraran en una sección pequeña de la órbita. Cada punto en la órbita implicaba recalcular los factores de vista satélite-Tierra, lo cual resultaba costoso y no producía beneficios cuando los puntos se hallaban tan próximos. Por ello se agregó además la posibilidad de especificar la cantidad de puntos deseada y

se implementó un algoritmo de filtrado que intenta retener esa cantidad de puntos equiespaciados.

4.4.1. Modelo numérico

Antes de brindar detalles de implementación, es fundamental comprender cuál es la formulación que permite resolver todo el cómputo de radiación en el preprocesador y aprovechar esos resultados en el solver sin pérdida de precisión.

La malla tridimensional del satélite, junto con los materiales asignados a cada polígono que la compone, son información suficiente para el cálculo de la transmisión de calor por conducción. Estos datos ya se encuentran estructurados de forma tal que es preciso y eficiente considerar solo el aporte de calor de la vecindad para cualquier punto en un tiempo determinado.

En cambio, para la transmisión de calor por radiación, la energía recibida en un punto es el resultado de integrar los rayos que fueron emitidos/reflejados por otros cuerpos (Tierra, Sol) o elementos del mismo satélite. Pese a ello, en el dominio del problema, ni la geometría del satélite, ni las características de los materiales se modifican. Más aún, los rayos viajan por el vacío, con lo que su energía solo disminuye con los rebotes. Estas simplificaciones permiten transformar la integración del efecto de rayos individuales en flujos constantes de energía en función de factores de vista entre elementos y los astros. De forma tal que conducción y radiación puedan resolverse homogéneamente con operaciones matriciales en el solver. Como ventaja adicional, la magnitud del aporte energético se separa de la proporción que se esperaría que un cuerpo reciba de otro, permitiendo reutilizar parámetros geométricos en distintos escenarios físicos.

Ahora bien, aún fue preciso realizar raycasting para calcular los factores de vista y extender el concepto a la Tierra y el Sol.

Para introducir el efecto de los materiales sobre la reflexión se evaluaron dos enfoques:

Enfoque A: Considerar la pérdida de energía por rayo en cada rebote de acuerdo a las características del material y detener la propagación al no sobrepasar un nivel mínimo de energía.

Enfoque B: Utilizar números pseudoaleatorios para decidir si un elemento absorbe o no a un rayo y solo contar al elemento que finalmente lo absorbió.

Se optó por el enfoque B, dado que si bien el enfoque A producía resultados más precisos para pocos rayos, acababa por disparar mayor cantidad de rayos al ser más frecuentes las reflexiones. Además, fue más sencillo razonar sobre el efecto de los materiales en rayos individuales.

En las siguientes secciones se especifica la implementación de cada interacción en la que se descompone el cálculo de transmisión de calor por radiación.

4.4.2. Interacción elemento-elemento

Para llevar cuenta de los factores de vista elemento a elemento se inicializa una matriz cuadrada con la misma cantidad de filas y columnas que de elementos. Los factores de vista no son simétricos, por lo que no se puede ahorrar cómputo o espacio en memoria con matrices triangulares.

Se emite la misma cantidad de rayos desde cada elemento y para cada rayo que colisionó se genera un número pseudoaleatorio entre cero y uno y se lo compara con la absorbividad al espectro infrarrojo del material del elemento impactado. Si es menor, se suma uno a la fila del elemento emisor y la columna del elemento impactado. De lo contrario, se reemite el rayo desde la posición de la colisión en la dirección reflejada respecto a la normal del elemento. Tras considerar todos los rayos de un elemento, se divide la fila correspondiente por el total de rayos emitidos. Se introdujo un

límite de reflexiones consideradas para asegurar que el algoritmo finalice en tiempo finito, aunque gracias a esto es posible que una fracción de rayos se pierdan cuando en la realidad no sería así.

Notar que los factores de vista elemento a elemento solo dependen de la geometría del modelo del satélite y, por tanto, alcanza con calcularlos una única vez.

4.4.3. Interacción Sol-elemento

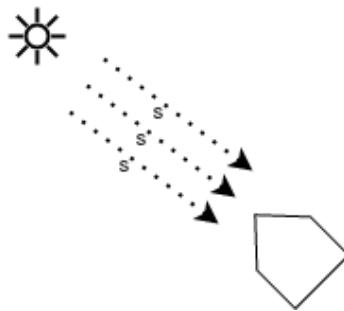


Figura 7: Orientación de Rayos Solares

Dada la distancia promedio entre el satélite y el Sol, los rayos que éste emite e impactan al satélite lo hacen en un ángulo comparable, es decir, el satélite interactúa con el Sol como si se tratase de un conjunto de rayos paralelos (Fig. 7). Además, en una órbita de tipo Sun pointing la aptitud del satélite no se modificará de forma apreciable. Aun así, fue necesario rotar el modelo del satélite para alineararlo con el Sol al inicio de la simulación (Fig. 8). Para ello, se definió que se modele el mismo tomando al eje Z en sentido positivo como la dirección hacia el Sol.

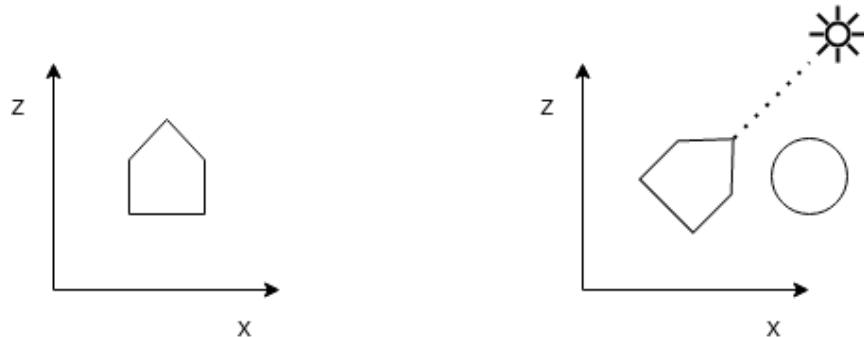


Figura 8: Rotación de Satélite

Fue posible simplificar aún más la estimación de las zonas iluminadas. En vez de emitir rayos desde el plano definido por la dirección del Sol, esperando que se alcancen todos los elementos del satélite, pueden emitirse rayos desde los elementos hacia el Sol. Si un rayo colisiona con el propio satélite, el rayo inverso Sol-satélite también colisionará (aunque no necesariamente con el mismo

punto), mientras que de perderse el rayo en el espacio, necesariamente el rayo inverso podría llegar al elemento sin problemas.



Figura 9: Corrección por Área Aparente

Por último, se debió aplicar una corrección de intensidad de las áreas iluminadas por el área aparente, lo que está implícito en el método Montecarlo al promediar la colisión de múltiples rayos. Se observa el caso bidimensional (Fig. 9), en donde la longitud del elemento es L , pero debido al ángulo de su normal respecto a la dirección de incidencia de los rayos solares, la proporción de radiación que recibe es menor:

$$\begin{aligned} P &= L \sin(\theta_l) , \\ \theta_n &= \theta_l + \frac{\pi}{2} , \\ \theta_{sn} &= \pi - \theta_n = \frac{\pi}{2} - \theta_l . \end{aligned} \tag{13}$$

Entonces:

$$P = L \sin\left(\frac{\pi}{2} - \theta_{sn}\right) = L \cos(\theta_{sn}) . \tag{14}$$

Al utilizar elementos con vértices coplanares para el mallado (en nuestro caso triángulos), este razonamiento puede extenderse directamente al caso tridimensional.

4.4.4. Interacción Tierra-elemento

Debido a la poca distancia que los separa, el satélite interactúa con la Tierra como si de un plano infinito se tratase. Los rayos provenientes de la Tierra por radiación infrarroja o albedo (reflejo de la luz solar) son emitidos desde puntos dentro del plano y direcciones aleatorias. El inconveniente con ceñirse a este modelo, tal y como se plantea, es que el área de vista de la Tierra es infinita (y por ende los factores de vista prácticamente nulos) y que al intentar calcular los factores de vista, de todos modos la probabilidad de que un rayo impacte al satélite sería muy baja. Por ello se invierten nuevamente emisor y receptor: se disparan rayos desde los elementos del satélite y se consideran

aquellos que no colisionan con otro elemento y cuyo ángulo con la dirección satélite-Tierra es menor a 90 grados o alternativamente, cuyo producto interno es mayor a cero (Fig. 10).

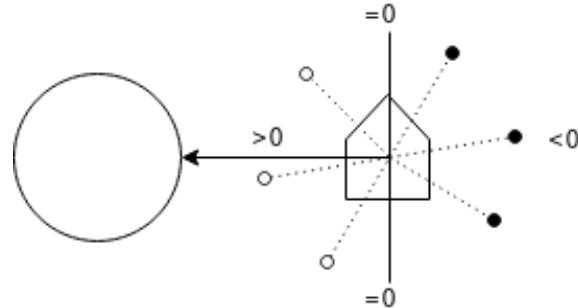


Figura 10: Producto Escalar con Dirección Terrestre

El cociente entre rayos que alcanzaron la Tierra y rayos totales fue una buena primera aproximación del factor de vista elemento-Tierra para radiación, pero no considera la curvatura de la Tierra, ni el ángulo de incidencia del rayo en el elemento. Para incorporar ambos factores se acabó realizando el promedio ponderado de los rayos, tomando como peso la multiplicación del producto interno entre vector satélite-Tierra y rayo (que brinda información del punto de procedencia del rayo) y el producto interno entre la normal del elemento y el rayo (análogo a la corrección por área aparente solar):

$$F_j^{\text{IR}} = \frac{1}{R_T} \sum_{i=0}^{R_I} (r_i \cdot v_{se}) (r_i \cdot n_j). \quad (15)$$

Donde:

- F_j^{IR} es el factor de vista del elemento j a la Tierra para la radiación infrarroja,
- R_T es la cantidad de rayos disparados por el método,
- R_I es la cantidad de rayos disparados que impactaron a la Tierra,
- r_i Es el iésimo rayo impactado,
- v_{se} Es el vector unitario en dirección a la Tierra, tomando la posición del satélite como origen de coordenadas,
- n_j Es la normal del elemento j .

En órbitas sun pointing el satélite rota respecto a la Tierra, por lo que estos factores de vista sí deberán ser calculados para distintos puntos de la órbita.

Con el procedimiento descrito pudo estimarse la energía intercambiada por radiación infrarroja, pero para el albedo debió aplicarse una corrección adicional por cada rayo que impactó al satélite, dependiendo de si había sido emitido desde un punto de la Tierra que el Sol iluminaba o no.

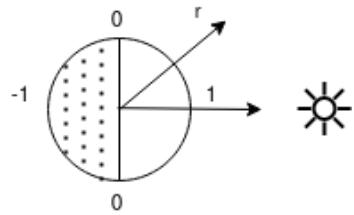


Figura 11: Producto Escalar con Dirección Solar desde Tierra

En principio, la Tierra puede considerarse como una esfera partida en dos, una semiesfera iluminada y otra en umbra. Los puntos de la semiesfera iluminada tienen un producto interno por el vector de la dirección solar positivo y los de la semiesfera en umbra un producto interno negativo (Fig. 11).

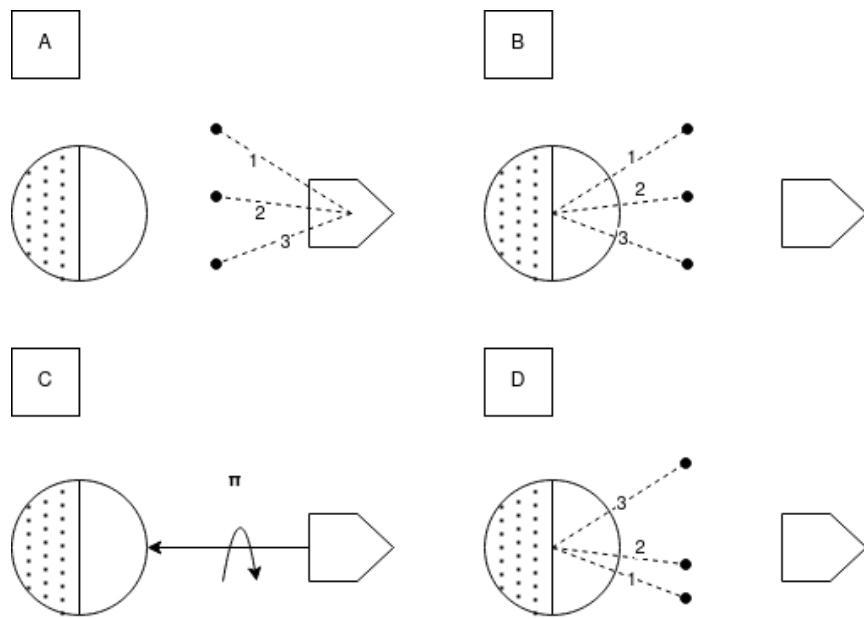


Figura 12: Estimación de Albedo - Caso 1

Como los rayos son emitidos desde el satélite hacia la Tierra en el cálculo, el producto interno debe invertirse (Fig. 12). Además, es preciso rotar en 180 grados sobre el vector Tierra a satélite, ya que como puede observarse en el segundo caso (Fig. 13), los rayos que provienen de zonas iluminadas o en umbra se intercambian si solo se invierte el sentido de los rayos.

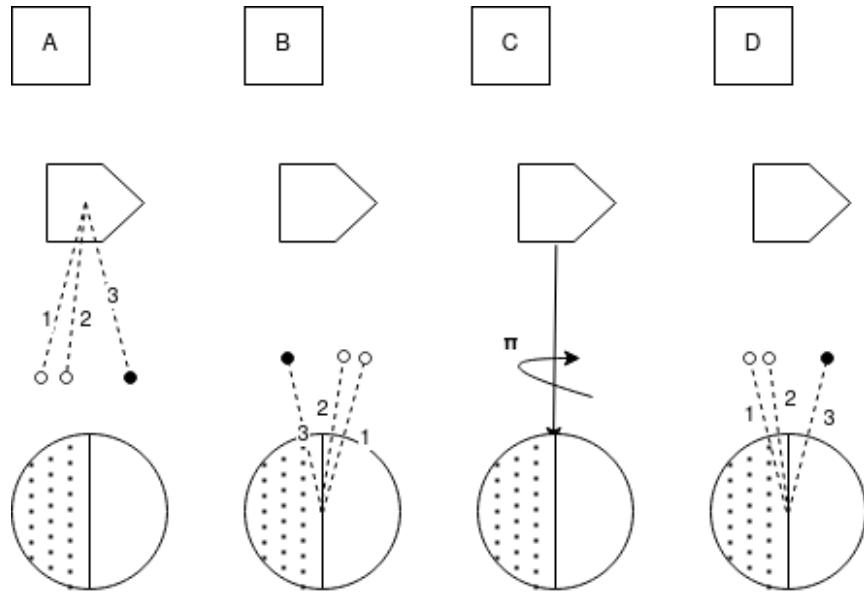


Figura 13: Estimación de Albedo - Caso 2

Es importante notar que la abrupta transición entre zonas iluminadas y en umbra es consecuencia del mapeo elegido entre producto interno e intensidad de luz (aquí la función de Heaviside). Puede suavizarse la transición entre zona iluminada y umbra e introducir así el efecto de scattering atmosférico^[29], pero no se observó que impactase en los resultados significativamente. Considerando los puntos anteriores, la ecuación del factor de vista del albedo resulta ser:

$$F_j^{\text{Albedo}} = \frac{1}{R_T} \sum_{i=0}^{R_I} u(r_i \cdot v_{es})(r_i \cdot v_{es})(r_i \cdot n_j). \quad (16)$$

Donde:

- F_j^{Albedo} es el factor de vista del elemento j a la Tierra para la radiación por albedo,
- R_T es la cantidad de rayos disparados por el método,
- R_I es la cantidad de rayos disparados que impactaron a la Tierra,
- r_i Es el iésimo rayo impactado con la rotación mencionada ya aplicada,
- v_{es} Es el vector unitario en dirección al Sol, tomando la posición de la Tierra como origen de coordenadas.
- n_j Es la normal del elemento j .
- u Es la función de Heaviside.

4.5. Solver

El solver fue construido con el objetivo de realizar la simulación de la transferencia de calor del satélite en órbita, frente a la interacción con las fuentes de energía anteriormente detalladas. Dado que el rendimiento es un objetivo relevante en esta etapa, se ha elegido a Rust como lenguaje de programación para el código a ser ejecutado en CPU, ya que brinda las facilidades de un lenguaje de programación moderno y, adicionalmente, ha demostrado tener tiempos de ejecución comparables a los de C en pruebas realizadas durante la fase de estudio de la problemática. Por otro lado, se eligió OpenCL como lenguaje de programación del código a ser ejecutado en GPU debido a que permite lograr una abstracción completa del hardware subyacente. Esta plataforma es soportada no solo por distintos proveedores de tarjetas gráficas, sino también por distintos tipos de hardware, entre los que se pueden encontrar CPUs y tensores. Como alternativa se estudió el uso de la plataforma CUDA. Sin embargo, la restricción que esta impone a su ejecución únicamente en dispositivos Nvidia fue lo que la llevó a ser descartada.

Al comienzo del proyecto, mientras aún se estudiaba la física del dominio del problema y se investigaban las herramientas tecnológicas y modelos numéricos existentes, se tomó la decisión de comenzar el desarrollo haciendo uso del método numérico FDM debido a su sencillez y a la previa experiencia del equipo en el uso de esta metodología. Sin embargo, rápidamente se detectó que la propia simplicidad de éste método podría implicar inestabilidades, las cuales a su vez podrían materializarse en grandes errores en simulaciones de geometrías complejas o incluso en la imposibilidad de realizar este tipo de simulaciones. Ante la detección de este riesgo, se decidió intensificar investigaciones sobre métodos alternativos, entre los cuales destacaban FEM y FVM por su gran utilización en la industria. Esta investigación culminó a los pocos días con la elección del método FEM, debido a su conocida estabilidad numérica frente a geometrías diversas, su abundante bibliografía y por ser el método que usualmente se emplea para el cálculo de este tipo de problemas.

4.5.1. Modelo FEM

A continuación se presentará un resumen del modelo FEM que fue empleado para la resolución de la simulación térmica. Se tomó como base el modelo presentado en el artículo Finite Element Method in Steady-State and Transient Heat Conduction^[30], el cual fue modificado para adecuarse al dominio de la problemática.

4.5.1.1 Ecuación de calor

Se comienza planteando la ecuación que gobierna la transferencia de calor en dos dimensiones:

$$c\rho \frac{\partial T}{\partial t} = \frac{\partial}{\partial x} \left(k_x \frac{\partial T}{\partial x} \right) + \frac{\partial}{\partial y} \left(k_y \frac{\partial T}{\partial y} \right) + q_v . \quad (17)$$

Donde:

- c es el calor específico del material,
- ρ la densidad del material,
- k_x y k_y las conductividades térmicas del material en cada componente.

Esta ecuación se encuentra sujeta a tres tipos de condiciones de contorno:

- Dirichlet: Temperatura conocida en una región,
- Neumann: Flujo de calor conocido en una región,
- Combinación Dirichlet y Neumann: Calor recibido debido a flujo de convección y temperatura ambiente conocidos en una región.

Debido a que el fenómeno de transmisión de calor por convección no se encuentra presente en el dominio del problema, la tercera condición fue descartada. Por otro lado, la condición de Dirichlet también fue descartada en pos de simplificar el modelo, ya que no suele ser utilizada por parte de los expertos, salvo simulaciones puntuales. Adicionalmente, la interacción de los elementos con el entorno debido al fenómeno de radiación fue resuelta por otros medios que no implican el uso de la condición de Neumann, siendo ésta anulada.

Por último, la condición inicial necesaria para resolver es:

$$T(x, y, t)|_{t=0} = T_0(x, y) . \quad (18)$$

Tras desarrollar la ecuación presentada, aplicando las condiciones y discretización en elementos, se obtiene la siguiente ecuación:

$$[M^e] \left\{ \frac{dT^e}{dt} \right\} + [K^e]\{T^e\} = \{f^e\} . \quad (19)$$

Donde:

- $[M^e]$ es la matriz de capacitancias,
- $[K^e]$ es la matriz de conductividad,
- $\{f^e\}$ es el vector de flujos.

Esta ecuación matricial es conocida como la representación de Galerkin de la ecuación de transmisión de calor transitoria. Si las propiedades físicas k, ρ, c son independientes de la temperatura, entonces la misma puede ser abordada utilizando métodos de resolución de sistemas de ecuaciones lineales. Las matrices que participan en esta ecuación hacen referencia a un único elemento de todo el dominio. A éstas se las conoce como matrices locales.

4.5.1.2 Construcción de matrices locales

Las matrices locales son construidas para elementos triangulares de primer orden (Fig. 14), es decir, tres nodos por elemento, sobre un plano bidimensional de dos coordenadas.

Dado un elemento genérico

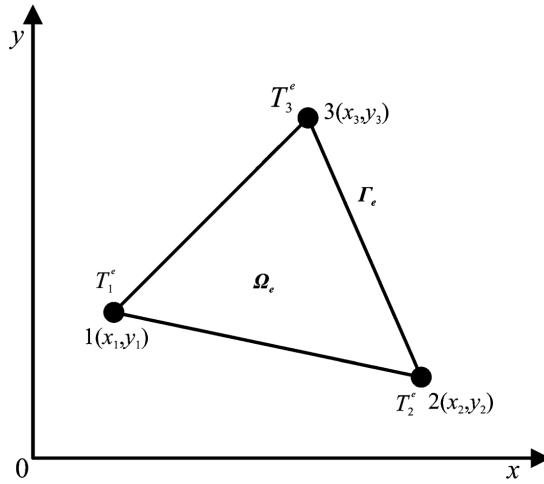


Figura 14: Elemento triangular bidimensional de primer orden^[30]

Tomando como hipótesis que la conductividad térmica del material es isotrópica ($k_x = k_y = k$). Es decir, los materiales poseen la misma conductividad en ambas direcciones. Considerando que A^e es el área del elemento, la matriz de conductividad del elemento, $[K_c^e]$, tiene la forma:

$$[K_c^e] = \begin{bmatrix} K_{11}^e & K_{12}^e & K_{13}^e \\ K_{21}^e & K_{22}^e & K_{23}^e \\ K_{31}^e & K_{32}^e & K_{33}^e \end{bmatrix}. \quad (20)$$

Donde:

$$\begin{aligned} K_{11}^e &= \frac{k}{4A^e}(b_1^2 + c_1^2) & K_{21}^e &= K_{12}^e & K_{31}^e &= K_{13}^e \\ K_{12}^e &= \frac{k}{4A^e}(b_1b_2 + c_1c_2) & K_{22}^e &= \frac{k}{4A^e}(b_2^2 + c_2^2) & K_{32}^e &= K_{23}^e \\ K_{13}^e &= \frac{k}{4A^e}(b_1b_3 + c_1c_3) & K_{23}^e &= \frac{k}{4A^e}(b_2b_3 + c_2c_3) & K_{33}^e &= \frac{k}{4A^e}(b_3^2 + c_3^2) \end{aligned} \quad (21)$$

$$\begin{aligned} b_1 &= y_2 - y_3 & c_1 &= x_3 - x_2 \\ b_2 &= y_3 - y_1 & c_2 &= x_1 - x_3 \\ b_3 &= y_1 - y_2 & c_3 &= x_2 - x_1 \end{aligned} \quad (22)$$

La matriz de capacitancia del elemento, $[M^e]$, tiene la forma:

$$[M^e] = \frac{A^e}{12} cp \begin{bmatrix} 2 & 1 & 1 \\ 1 & 2 & 1 \\ 1 & 1 & 2 \end{bmatrix}. \quad (23)$$

El vector de flujo debido a generación de calor interna, f^e , tiene la forma

$$\{f^e\} = q \frac{A^e}{3} \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix}. \quad (24)$$

Donde q es el calor generado dentro del propio elemento.

4.5.1.3 Construcción de matrices globales

Hasta el momento se han detallado las ecuaciones y matrices empleadas a nivel local, dentro de un único elemento. A continuación se explicará cómo es posible expandir este modelo al resto de los elementos.

Se parte de un ejemplo de un modelo con 2 elementos y 4 nodos. A cada nodo se le asignará una numeración local (Fig. 15) , que lo identifica dentro de un mismo elemento, y una numeración global (Fig. 16) , que lo identifica frente al resto de nodos del modelo.

La numeración global es la siguiente:

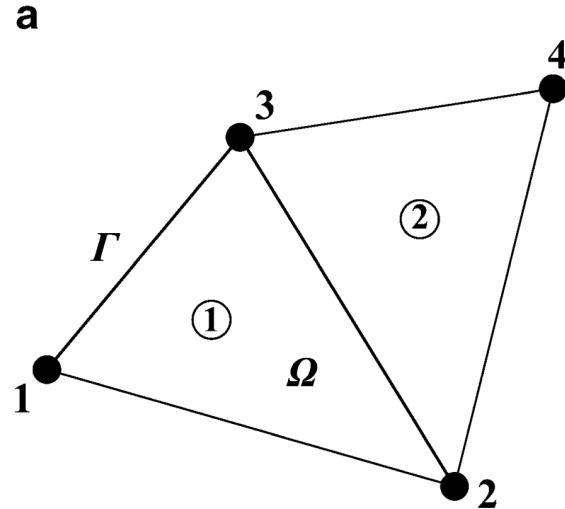


Figura 15: Numeración global de los nodos.^[30]

Y la numeración local:

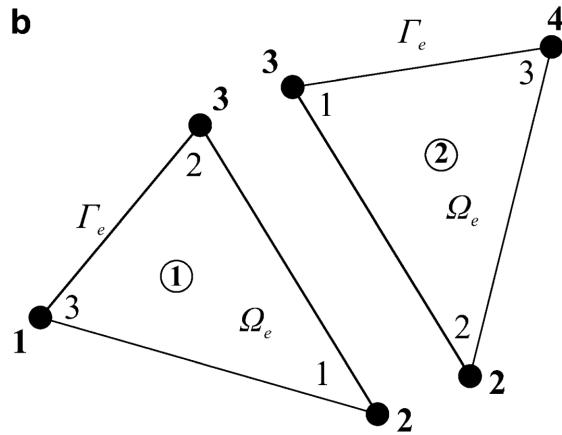


Figura 16: Numeración local de los nodos.^[30]

El proceso de construcción de una matriz global consiste en utilizar los identificadores globales de los nodos de cada matriz local para agregar los resultados en la matriz global. De esta forma, por cada coordenada (x, y) de identificadores locales dentro de una matriz local existirá un mapeo a una coordenada (u, v) de identificadores globales en la matriz global (Fig. 17).

Por tomar un ejemplo, en el elemento número 1, el nodo de identificador local 1 tiene asignado un identificador global 2. Por ende, su mapeo será $(1, 1) \rightarrow (2, 2)$.

$$\begin{aligned}
 [\mathbf{K}^{e1}] &= \begin{matrix} \textcircled{2} \\ \textcircled{3} \\ \textcircled{1} \end{matrix} \begin{bmatrix} K_{11}^{(1)} & K_{12}^{(1)} & K_{13}^{(1)} \\ K_{21}^{(1)} & K_{22}^{(1)} & K_{23}^{(1)} \\ K_{31}^{(1)} & K_{32}^{(1)} & K_{33}^{(1)} \end{bmatrix} \quad [\mathbf{K}^{e2}] = \begin{matrix} \textcircled{3} \\ \textcircled{2} \\ \textcircled{4} \end{matrix} \begin{bmatrix} K_{11}^{(2)} & K_{12}^{(2)} & K_{13}^{(2)} \\ K_{21}^{(2)} & K_{22}^{(2)} & K_{23}^{(2)} \\ K_{31}^{(2)} & K_{32}^{(2)} & K_{33}^{(2)} \end{bmatrix} \\
 [\mathbf{K}] &= \begin{matrix} \textcircled{1} \\ \textcircled{2} \\ \textcircled{3} \\ \textcircled{4} \end{matrix} \begin{bmatrix} K_{33}^{(1)} & K_{31}^{(1)} & K_{11}^{(1)} + K_{22}^{(1)} & K_{12}^{(1)} + K_{21}^{(1)} & K_{13}^{(1)} \\ K_{13}^{(1)} & K_{23}^{(1)} & K_{21}^{(1)} + K_{12}^{(1)} & K_{22}^{(1)} + K_{11}^{(1)} & K_{13}^{(1)} \\ K_{23}^{(1)} & 0 & K_{32}^{(2)} & K_{31}^{(2)} & K_{33}^{(2)} \\ 0 & K_{32}^{(2)} & 0 & K_{31}^{(2)} & K_{33}^{(2)} \end{bmatrix}
 \end{aligned}$$

Figura 17: Construcción de matriz global a partir de matrices locales.^[30]

De forma análoga es posible construir el vector de flujos global (Fig. 18).

$$\begin{aligned}
\{\mathbf{f}^{e1}\} &= \left\{ \begin{array}{c} f_1^{(1)} \\ f_2^{(1)} \\ f_3^{(1)} \end{array} \right\} \begin{array}{l} \textcircled{2} \\ \textcircled{3} \\ \textcircled{1} \end{array} \\
&\quad \left\{ \mathbf{f}^{e2} \right\} = \left\{ \begin{array}{c} f_1^{(2)} \\ f_2^{(2)} \\ f_3^{(2)} \end{array} \right\} \begin{array}{l} \textcircled{3} \\ \textcircled{2} \\ \textcircled{4} \end{array} \\
&\quad \left\{ \mathbf{f} \right\} = \left\{ \begin{array}{c} f_3^{(1)} \\ f_1^{(1)} + f_2^{(2)} \\ f_2^{(1)} + f_1^{(2)} \\ f_3^{(2)} \end{array} \right\} \begin{array}{l} \textcircled{1} \\ \textcircled{2} \\ \textcircled{3} \\ \textcircled{4} \end{array}
\end{aligned}$$

Figura 18: Construcción de vector global a partir de vectores locales.^[30]

Una vez obtenidas las matrices globales, es posible expandir la ecuación de Galerkin a su forma global:

$$[M] \left\{ \frac{dT}{dt} \right\} + [K]\{T\} = \{f\}. \quad (25)$$

Luego, si esta ecuación es discretizada en función del tiempo haciendo uso del método de Crank-Nicolson generalizado, se obtiene:

$$\left(\frac{1}{\Delta t} [M] + \theta [K] \right) \{T\}^{n+1} = \left(\frac{1}{\Delta t} [M] - (1-\theta) [K] \right) \{T\}^n + (1-\theta)\{f\}^n + \theta\{f\}^{n+1}. \quad (26)$$

Eligiendo $\theta = 0,5$ se obtiene el conocido esquema semi-implícito por método de Crank-Nicolson. Para que al añadir a ésta los términos elevados a la cuarta potencia referentes a radiación no se pierda la linealidad de la ecuación, se decidió hacer uso de una corrección del método tal que se eliminase su característica implícita en el vector de flujos. Esta modificación no ha demostrado tener impacto en la precisión del modelo al ser comparada con ejecuciones sobre ecuaciones de carácter explícito. Esto se debe posiblemente al hecho de que el vector de flujos entre iteraciones consecutivas no se ve alterado de manera significativa, salvo saltos discretos puntuales (por ejemplo, la transición desde eclipse) que rápidamente son corregidos en iteraciones posteriores.

En esta modificación se asume para el instante n , que el flujo permanecerá constante en el instante siguiente. Esto es:

$$\{f\}^n = \{f\}^{n+1}. \quad (27)$$

Por lo que la ecuación puede reescribirse de la siguiente forma:

$$\left(\frac{1}{\Delta t} [M] + \theta [K] \right) \{T\}^{n+1} = \left(\frac{1}{\Delta t} [M] - (1 - \theta) [K] \right) \{T\}^n + \{f\}^n. \quad (28)$$

4.5.1.4 Expansión a tres dimensiones

El modelo FEM que ha sido desarrollado hasta este punto es únicamente válido para modelos bidimensionales. Es decir, para transmisiones de calor en geometrías planas.

Dada la naturaleza del dominio del problema, es necesario aplicar pequeñas correcciones para extender el modelo a uno 2,5D, que no considerará la transmisión de calor de manera volumétrica, sino sobre la propia superficie de los objetos tridimensionales modelados.

Los cambios a realizar implican incorporar un parámetro de grosor G_e para cada elemento (asociado a una superficie) e incrementar la dimensionalidad de los puntos del modelo de 2 a 3 coordenadas (x, y, z) .

Tras aplicar estos cambios será necesario modificar la forma en la que algunas matrices locales son construidas.

La matriz de capacitancia será multiplicada por el grosor del elemento:

$$[M^e] = \frac{A^e}{12} G_e c \rho \begin{bmatrix} 2 & 1 & 1 \\ 1 & 2 & 1 \\ 1 & 1 & 2 \end{bmatrix}. \quad (29)$$

Por otro lado, la matriz conductividad también será multiplicada por el grosor del elemento, pero además verán modificadas las constantes que dan lugar a su creación.

Obsérvese que los cálculos de las constantes b_i y c_i refieren a propiedades espaciales del elemento. La operación $b_1^2 + c_1^2$ no es más que la distancia al cuadrado entre el vértice 2 y el vértice 3 del elemento. Por otro lado, la operación $b_1 b_2 + c_1 c_2$ se puede interpretar como el producto escalar entre dos vectores, el primero es el que une el vértice 2 con el vértice 3, y el segundo el que une el vértice 3 con el vértice 1.

Por lo tanto, todos estos cálculos pueden realizarse a partir de las coordenadas de los vértices en 3D. Es así que, la matriz de conductividad puede ser construida de la siguiente forma:

$$[K_c^e] = \begin{bmatrix} K_{11}^e & K_{12}^e & K_{13}^e \\ K_{21}^e & K_{22}^e & K_{23}^e \\ K_{31}^e & K_{32}^e & K_{33}^e \end{bmatrix}. \quad (30)$$

Donde:

$$\begin{aligned}
K_{11}^e &= G_e \frac{k}{4A_e} \|v_3 - v_2\|^2 & K_{21}^e &= K_{12}^e & K_{31}^e &= K_{13}^e \\
K_{12}^e &= G_e \frac{k}{4A_e} (v_3 - v_2) \cdot (v_1 - v_3) & K_{22}^e &= G_e \frac{k}{4A_e} \|v_3 - v_1\|^2 & K_{32}^e &= K_{23}^e \\
K_{13}^e &= G_e \frac{k}{4A_e} (v_3 - v_2) \cdot (v_2 - v_1) & K_{23}^e &= G_e \frac{k}{4A_e} (v_1 - v_3) \cdot (v_2 - v_1) & K_{33}^e &= G_e \frac{k}{4A_e} \|v_2 - v_1\|^2
\end{aligned} \tag{31}$$

Siendo v_1, v_2, v_3 los vértices del elemento.

4.5.1.5 Incorporación de radiación

Ya habiendo conseguido un modelo que permite simular la transferencia de calor debido a conducción, queda añadir a este la capacidad de transmitir energía debido a radiación.

Como se ha comentado, el modelo FEM realizado no hace uso de las condiciones de borde de Dirichlet, Neumann y Neumann-Dirichlet combinado. En su lugar, la transmisión de calor debido a radiación se modela como vectores de flujos de calor, los cuales agregados resultan en un vector de flujo de calor neto a nivel elemento. Para lograr esto, se recuerdan las ecuaciones descritas en la sección 3.1 que modelan la radiación para un sistema espacial:

$$\begin{aligned}
Q_{\text{Sun}} &= \alpha_i^{\text{Sun}} F_{\text{Sun},i} A_i S, \\
Q_{\text{Albedo}} &= \alpha_i^{\text{Sun}} F_{\text{Earth},i} A_i A_{f_i} S, \\
Q_{\text{IR}} &= \alpha_i^{\text{IR}} F_{\text{Earth},i} E_{\text{IR}}, \\
Q_{\text{Lost}} &= \alpha^{\text{IR}} A^r \sigma T^4, \\
Q_{i,j}^{\text{elem}} &= \sigma \alpha_i^{\text{IR}} \alpha_j^{\text{IR}} F_{i,j} A_i T_i^4.
\end{aligned} \tag{32}$$

Estas ecuaciones pueden ser reescritas en función de los elementos y nodos del método FEM, haciendo uso de representaciones matriciales locales y/o globales. De esta forma, para un elemento e las representaciones locales son:

$$\begin{aligned}
\{f_{\text{Sun}}^e\} &= \alpha_e^{\text{Sun}} F_{\text{Earth},e} \frac{A_e}{3} S \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix}, \\
\{f_{\text{Albedo}}^e\} &= \alpha_e^{\text{Sun}} F_{\text{Sun},e} A_{f_e} \frac{A_e}{3} S \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix}, \\
\{f_{\text{IR}}^e\} &= \alpha_e^{\text{IR}} F_{\text{Earth},e} \frac{A_e}{3} E_{\text{IR}} \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix}, \\
\{f_{\text{Lost}}^e\} &= [E^e] \begin{bmatrix} T_1^{e4} \\ T_2^{e4} \\ T_3^{e4} \end{bmatrix}.
\end{aligned} \tag{33}$$

Donde:

$$[E^e] = \sigma \alpha_e^{\text{IR}} \frac{A_e}{3} \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}. \tag{34}$$

Cada una de estas matrices se puede expandir a una matriz global aplicando el método explicado anteriormente.

De todas las ecuaciones presentadas, la única que no posee representación matricial local es la ecuación de transmisión de calor entre elementos. Dicha ecuación posee únicamente una representación global, que se muestra a continuación:

$$[L] = \frac{\sigma}{3} \begin{bmatrix} L_{11} & L_{12} & \dots & L_{1n} \\ L_{21} & L_{22} & \dots & L_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ L_{n1} & L_{n2} & \dots & L_{nn} \end{bmatrix}. \tag{35}$$

Tal que:

$$\begin{aligned}
L_{i,j} &= \sum_{k=\text{Elem}(i)} \sum_{w=\text{Elem}(j)} P_{w,k}, \\
i, j &\in \{1 \dots N\}. \\
P_{w,k} &= F_{w,k} \alpha_w^{IR} \alpha_k^{IR} A_w, \\
w, k &\in \{1 \dots V\}.
\end{aligned} \tag{36}$$

Donde:

- N : Número de nodos,
- V : Número de elementos,
- $\text{Elem}(i)$: Conjunto de los elementos a los que pertenece el nodo i ,
- $L_{i,j}$: Energía emitida por el nodo j , recibida por el nodo i ,
- $P_{w,k}$: Energía emitida por el elemento w , recibida por el elemento k .

De esta forma, queda definido el flujo de calor entrante por transmisión entre elementos:

$$\{f_{\text{Elem}}\} = [L] \begin{bmatrix} T_1^4 \\ T_2^4 \\ \vdots \\ T_n^4 \end{bmatrix}. \tag{37}$$

4.5.1.6 Integración de las partes

Habiendo explicado el modelo FEM y las ecuaciones de radiación en sus formas matriciales, queda ahora combinar las ecuaciones para obtener un sistema lineal a partir del cual se pueda realizar la simulación deseada.

Es posible reescribir la ecuación de Galerkin semi-implícita de la siguiente forma:

$$[A] \{T\}^{n+1} = [D] \{T\}^n + \{C\}^n. \tag{38}$$

Tal que:

$$\begin{aligned}
[A] &= \frac{1}{\Delta t} [M] + \theta [K], \\
[D] &= \frac{1}{\Delta t} [M] - (1 - \theta) [K], \\
\{C\}^n &= \{f_{\text{Gen}}\} + \{f_{\text{Sun}}\} + \{f_{\text{IR}}\} + \{f_{\text{Albedo}}\} + \{f_{\text{Elem}}\} - \{f_{\text{Lost}}\}.
\end{aligned} \tag{39}$$

Donde $\{f_{\text{Gen}}\}$ es el vector constante de generación interna de calor a nivel elemento, cuya representación local es:

$$\{f_{\text{Gen}}^e\} = q_{\text{Gen}} \frac{A^e}{3} \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix}. \tag{40}$$

El vector $\{C\}^n$ puede reescribirse de la siguiente forma:

$$\{C\}^n = \{f_{\text{Const}}\} + [H] \begin{bmatrix} T_1^4 \\ T_2^4 \\ \vdots \\ T_n^4 \end{bmatrix}. \tag{41}$$

Donde:

$$\begin{aligned}
\{f_{\text{Const}}\}^n &= \{f_{\text{Gen}}\} + \{f_{\text{Sun}}\} + \{f_{\text{IR}}\} + \{f_{\text{Albedo}}\}, \\
[H] &= [L] - [E].
\end{aligned} \tag{42}$$

De esta forma, al resolver el sistema de ecuaciones lineales (Ec. 38) para $\{T\}^{n+1}$, se obtendrán los valores de las temperaturas de los nodos del modelo para un paso de tiempo. Repetir este procedimiento de forma iterativa dará como resultado una simulación de la transferencia y propagación del calor en el satélite en función del tiempo, dadas las condiciones configuradas.

4.5.2. Resolución del sistema de ecuaciones

Existen distintas técnicas para resolver sistemas de ecuaciones lineales. La más sencilla de todas implica hacer uso de la inversa de la matriz de coeficientes.

Dado el sistema de ecuaciones lineales:

$$[A]\{x\} = \{b\}. \tag{43}$$

Se obtiene el vector incógnita multiplicando el vector de términos constantes por la inversa de la matriz de coeficientes:

$$\{x\} = [A^{-1}]\{b\}. \quad (44)$$

Esta técnica tiene un punto positivo y uno negativo. Como punto positivo, una vez obtenida la inversa, esta puede ser utilizada tantas veces como sea necesario para iterar sobre la simulación y obtener las temperaturas de los nodos en cada paso de tiempo con una simple multiplicación. Como punto negativo, el algoritmo para computar la inversa requiere realizar un aproximado de $2n^3$ operaciones para una matriz de $n \times n$.

Una alternativa a este método es hacer uso de una descomposición LU^[31], en la cual la matriz de coeficientes es descompuesta en dos matrices auxiliares L y U.

Dado el sistema de ecuaciones lineales:

$$[A]\{x\} = \{b\}. \quad (45)$$

Se realiza una descomposición de la matriz de coeficientes en una matriz L triangular inferior, y una matriz U triangular superior:

$$[A] = [L][U]. \quad (46)$$

Reemplazando en el sistema de ecuaciones:

$$[L][U]\{x\} = \{b\}. \quad (47)$$

Que puede ser reescrito de la siguiente forma:

$$\begin{aligned} [L]\{y\} &= \{b\}, \\ [U]\{x\} &= \{y\}. \end{aligned} \quad (48)$$

Por ende, resolviendo secuencialmente ambos sistemas de ecuaciones, se podrá obtener la solución al sistema original. Ahora bien, gracias a que las matrices L y U son triangular inferior y superior respectivamente, es posible resolver ambos sistemas de ecuaciones en aproximadamente $2n^2$ operaciones.

Por otro lado, el coste computacional de realizar la descomposición LU es de aproximadamente $\frac{2}{3}n^3$ operaciones.

De esta forma, el costo computacional total para resolver el sistema de ecuaciones para una matriz de $n \times n$ será:

$$\sim \frac{2}{3}n^3 + 2n^2. \quad (49)$$

Esta técnica, al igual que la anterior, tiene un punto positivo y otro negativo. Como punto positivo, obtener la factorización LU es unas 3 veces más rápido que obtener la inversa de la matriz de coeficientes. Como punto negativo, cada vez que se requiera despejar un nuevo vector incógnita será necesario resolver los dos sistemas de ecuaciones de las matrices L y U antes detallado.

Otros algoritmos de resolución por métodos iterativos, como son el método de Jacobi, Gauss-Seidel y SOR también han sido estudiados. Sin embargo, todos ellos fueron descartados rápidamente, ya que requieren que la matriz de coeficientes sea diagonalmente dominante, característica que no puede ser asegurada para el dominio de este trabajo.

4.5.3. Resolución en CPU

La ejecución del solver puede dividirse en dos etapas. Por un lado, se tiene la construcción de los datos necesarios para iniciar la simulación, esto es, las matrices y vectores locales y globales antes mencionados. Por el otro, se tiene la propia ejecución de la simulación, resolviendo la ecuación principal (Ec. 38) en cada iteración de la misma.

Con el objetivo de contrastar los métodos de inversión de matriz y descomposición LU para la resolución del sistema de ecuaciones, se realizó una prueba sencilla, haciendo uso de un cubo de aproximadamente 10000 elementos y 5000 nodos. Sobre este modelo se realizó una simulación que consistió en 6000 pasos o iteraciones, totalizando 30000 segundos de simulación.

		Método	
		Inversión	LU
Etapas	Construcción	183,09 s	79,39 s
	Simulación	110,39 s	117,10 s
Error		0,042399 %	

Cuadro 1: Descripción de los tiempos y error para cada etapa y método.

En base al tiempo de ejecución por etapa para cada método, se observa que el método de descomposición LU es capaz de realizar la construcción en la mitad del tiempo que demora la construcción por el método de inversión. Por otro lado, la etapa de simulación posee tiempos similares para ambos métodos. Por esta razón es que se decidió hacer uso del método de descomposición para la implementación del solver sobre CPU.

Adicionalmente, se compararon los resultados obtenidos entre ambos métodos y se computó el error absoluto medio, obteniendo el error detallado en el cuadro (Cuad. 1). Esto demuestra que, al menos para el dominio de este trabajo, ambos métodos poseen la misma estabilidad numérica.

4.5.4. Resolución en GPU

Llegado este punto se tiene una implementación funcional del solver que permite realizar simulaciones haciendo uso de la CPU. A partir de ahora, se buscará desarrollar una implementación

que aproveche la potencia de la GPU, con la finalidad de reducir significativamente los tiempos de ejecución.

En contraste con las CPUs, las GPUs cuentan con un gran número de núcleos de ejecución. Una GPU moderna puede superar fácilmente los 3000 núcleos, en comparación con los 12 núcleos de una CPU del mismo año. Sin embargo, es importante tener en cuenta que los núcleos de las GPUs son individualmente más simples y lentos en comparación con los núcleos de las CPUs. Además, se debe considerar la latencia en la transferencia de datos entre la memoria principal (RAM) y la memoria del dispositivo (GPU). Estas consideraciones se pueden resumir en las siguientes heurísticas clave al buscar optimizar la programación para GPU:

- El programa debe poder ser descompuesto en porciones capaces de ser ejecutadas de forma paralela.
- Se debe evitar interrumpir la ejecución de la GPU con transferencias entre la memoria principal y la memoria del dispositivo.
- Se deben limitar lo máximo posible los accesos del programa a memoria del dispositivo.

Los programas destinados a GPUs suelen ser mayoritariamente codificados en lenguajes basados en C, con pequeñas variaciones que añaden funcionalidades específicas, como la obtención de identificadores de proceso o la sincronización entre procesos. Estos programas se componen principalmente de funciones conocidas como kernels.

Para la programación en GPU, se ha optado por el estándar multiplataforma y de código abierto OpenCL. Con el fin de simplificar su implementación en el lenguaje Rust, se ha seleccionado la biblioteca ocl como binding, debido a su popularidad, documentación exhaustiva y su sencilla abstracción orientada a datos.

OpenCL posee un modelo de ejecución comprendido, entre otras abstracciones, por work items, work groups y processing elements (PE). Los PE son las unidades encargadas de llevar a cabo el cómputo y en las GPUs tienen una relación uno a uno con los núcleos. Los PE se reúnen en work groups, dentro de los cuales se compartirán las instrucciones a ejecutar y un espacio de memoria local. Las tareas que se asignen al dispositivo y los datos sobre los que operan serán divididas en particiones, denominadas work items. Cada work item será asignado a un work group, y a un PE particular dentro del mismo. La ejecución de los work items se realizará de forma paralela según la cantidad de PE disponibles en el dispositivo. Si la cantidad de work items generados superase la cantidad de PE disponibles, el cómputo no podrá resolverse en una sola etapa y los work items restantes quedarán a la espera de la liberación de los work groups. Esta es la principal restricción arquitectónica al grado de paralelización.

Como se ha mencionado, el solver puede dividirse en dos etapas: la construcción de los datos necesarios para la simulación y la ejecución propiamente dicha de la simulación. Esto implica que existen dos oportunidades distintas para lograr un aumento en la velocidad, especialmente si estas etapas son llevadas a cabo en una GPU.

El algoritmo de descomposición LU utilizado en la implementación para CPU presenta un comportamiento secuencial en la fase de solución de las ecuaciones lineales sobre las matrices triangulares. Esta característica reduce cualquier potencial mejora obtenida mediante la paralelización de operaciones en la GPU. Por este motivo, se tomó la decisión de implementar la solución en GPU utilizando el método de inversión de matriz.

El desarrollo de este método se dividió en dos fases. La primera tuvo como objetivo la inversión de la matriz de coeficientes, mientras que la segunda se dedicó a implementar las operaciones necesarias para realizar la simulación en la GPU.

En cuanto a la inversión de la matriz, se exploraron varios algoritmos, siendo el algoritmo de Gauss-Jordan el más destacado debido a su simplicidad y capacidad de paralelización. Sin embargo, tras dedicar varios días a su implementación, se optó por abandonar esta elección debido a limitaciones de tiempo y a la aparición de errores asociados a concurrencia.

Para la segunda etapa se implementaron kernels basados en las funciones de GeMM (General Matrix Multiply)^[32], provenientes de la conocida biblioteca de álgebra lineal BLAS^[33].

Los kernels implementados fueron

```

1 __kernel void fourth_elevation(
2     __global double *t,
3     __global double *t_4,
4     int n
5 );
6
7 __kernel void gemv1(
8     __global const double *a,
9     __global const double *x,
10    __global double *y,
11    int m,
12    int n
13 );
14
15 __kernel void vec_sum(
16    __global double *a,
17    __global double *b,
18    __global double *c,
19    int n
20 );

```

Figura 19: Kernels del solver, basados en funciones GeMM.

El kernel `fourth_elevation` (Fig. 19) recibe un vector `t` de tamaño `n` contenido las temperaturas de los nodos y escribe el resultado de elevar cada componente de éste en el vector `t_4` del mismo tamaño. Para su ejecución se divide el trabajo en `n` work items.

El kernel `gemv1` realiza la multiplicación de la matriz `a` de dimensionalidad $m \times n$ por el vector `x`, escribiendo el resultado en el vector `y`. Aquí se hace uso de una versión simplificada del algoritmo GEMM antes citado. Para su ejecución se divide el trabajo en `m` work items, cada una de las cuales realiza el cómputo de una fila de la matriz `a`.

El kernel `vec_sum` realiza una suma entre los dos vectores `a` y `b` de tamaño `n` y escribe el resultado en el vector `c`. Para su ejecución se divide el trabajo en `n` work items.

Los pasos que son llevados a cabo para realizar una iteración en la simulación son los siguientes

Primero se utiliza `fourth_elevation` para elevar a la cuarta el vector de temperaturas obtenido en la iteración anterior:

$$\{T\}_4 = \{T\}^4. \quad (50)$$

Luego, se utiliza `gemv1` para multiplicar la matriz H por el vector de temperaturas elevadas a la cuarta, almacenando el resultado en el vector f :

$$\{f\} = [H]\{T\}_4. \quad (51)$$

Posteriormente, se utiliza `vec_sum` para actualizar el valor de f con los valores de f_{Const} :

$$\{f\} = \{f\} + \{f_{\text{Const}}\}. \quad (52)$$

Por otro lado, se hace uso de `gemv1` para la multiplicación de la matriz D por el vector de temperaturas, almacenando el resultado en el vector b :

$$\{b\} = [D]\{T\}. \quad (53)$$

Luego, se utiliza `vec_sum` para actualizar el valor de b con la suma de b y f :

$$\{b\} = \{b\} + \{f\}. \quad (54)$$

Por último, se emplea `gemv1` para la multiplicación de la inversa de la matriz A por el vector b , obteniendo así el nuevo vector de temperaturas:

$$\{T\} = [A]^{-1}\{b\}. \quad (55)$$

Estas operaciones son encoladas por parte de la CPU a una cola de trabajo que provee OpenCL como medio de comunicación con la GPU. Gracias a la forma en la cual las operaciones fueron construidas, es posible ejecutar varias iteraciones de la simulación sin necesidad de realizar transferencias de datos entre la memoria de la GPU y la memoria principal. Antes de comenzar la ejecución, el programa en CPU calcula el número de iteraciones a realizar y encola exactamente el número de operaciones acordes a este, tal que la GPU se mantenga ocupada de manera ininterrumpida. El número de iteraciones a realizar estará relacionado con la necesidad de obtención de resultados parciales por parte de la CPU, o a la actualización del vector de flujos constante, el cual tiene dependencias con la posición del satélite en la órbita.

4.6. Postprocesado

Los resultados de la simulación térmica son exportados en VTK, que es un formato estándar directamente soportado por programas de análisis de datos y visualización.

Se decidió integrar Paraview directamente en la herramienta para evitar que el usuario deba buscar manualmente el archivo de resultados indicado entre los miles de archivos auxiliares que pueden generarse.

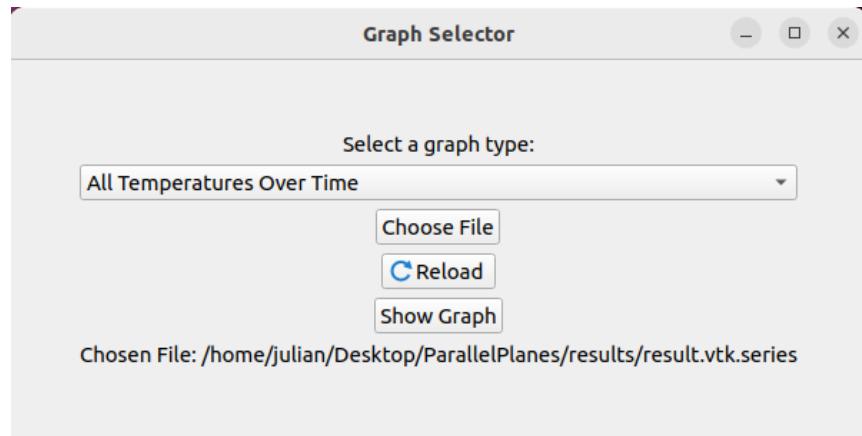


Figura 20: Interfaz de usuario del Plotter

Para las demás visualizaciones se desarrolló en Python, haciendo uso de Matplotlib, una herramienta denominada Plotter (Fig. 20). La misma permite realizar gráficos de la temperatura a través del tiempo a partir de los archivos VTK generados en el solver, ya sea para todos los nodos del modelo, o para alguno en particular. También permite graficar la temperatura promedio y el desvío estándar. Esta herramienta también se encuentra integrada en la GUI desarrollada y su implementación temprana fue de gran ayuda para la interpretación de los benchmarks durante el desarrollo del proyecto.

5. Resultados

La obtención de datos de validación para comparar los resultados del sistema (experimentales o provenientes de software de la industria) fue uno de los desafíos más grandes del proyecto. Fue imposible obtener resultados que englobasen el conjunto de las interacciones físicas teóricamente comprobables. Como alternativa, se presentan comparaciones individuales y ejemplos sencillos.

Con el objetivo de simplificar el presente documento, se incluirán los datos esenciales para el entendimiento de estas pruebas. Para datos adicionales, referirse al anexo “Resultados Extendidos”, donde también podrán encontrarse pruebas adicionales similares a las aquí presentadas.

5.1. Comparaciones

En esta sección se detallan las pruebas de conducción y radiación realizadas sobre Agni B3V, comparando los resultados obtenidos contra simulaciones equivalentes realizadas en el software libre de simulación térmica PrePoMax o, en su defecto, contra modelos teóricos.

5.1.1. Conducción

A continuación, se proporciona un desglose de las pruebas llevadas a cabo en simulaciones de conducción y flujos de calor constantes.

Los modelos de estas pruebas tienen en común las siguientes propiedades:

- Fenómenos de radiación desactivados:
 - Entre elementos
 - Perdida al espacio
 - Proveniente del Sol
 - Proveniente de la Tierra
- Espesor de 1 mm

5.1.1.1 Prueba 1: Conducción simple en cubo

Se modeló un cubo con las siguientes características:

- Tamaño de $1\text{ m} \times 1\text{ m} \times 1\text{ m}$
- Material aluminio, con:
 - Densidad de 2700 $\frac{\text{kg}}{\text{m}^3}$
 - Calor específico de 900 $\frac{\text{J}}{\text{kg K}}$
 - Conductividad térmica de 237 $\frac{\text{W}}{\text{K m}}$
- Una cara con temperatura inicial de 300 °C, el resto con temperatura inicial de 0 °C
- Sin flujos

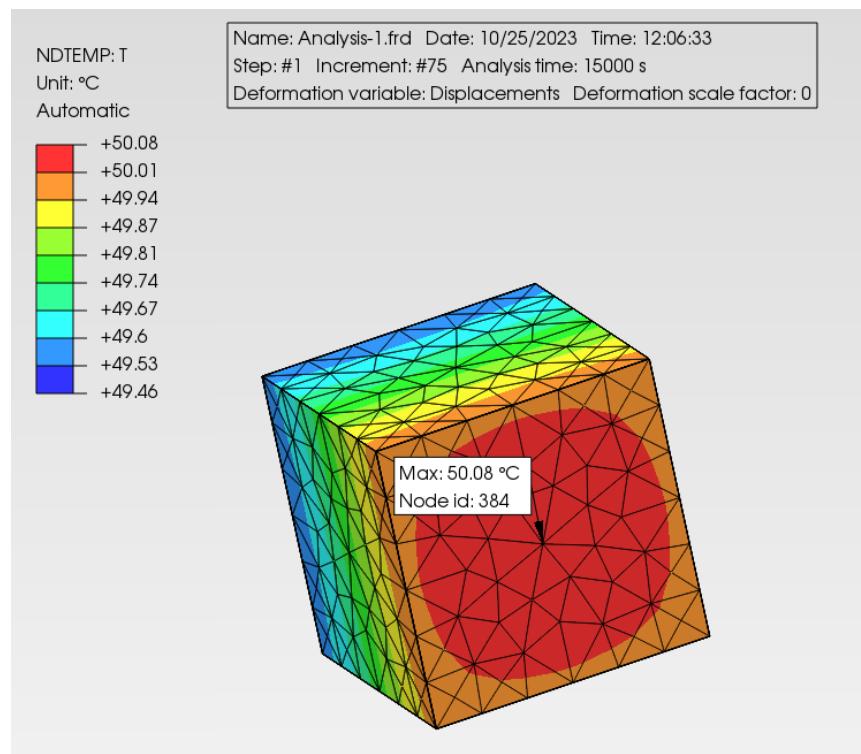


Figura 21: Vista volumétrica de temperatura - PrePoMax

Bajo PrePoMax (Fig. 21), tras 15000 segundos de simulación se llegó a una temperatura estable de entre 49,46 °C y 50,08 °C

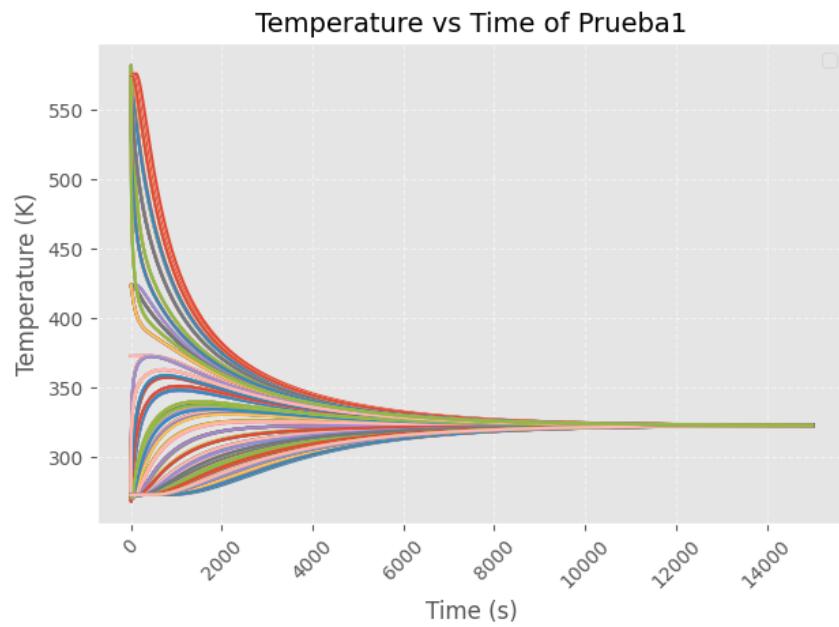


Figura 22: Evolución de la tempertaura de cada nodo - Agni B3V

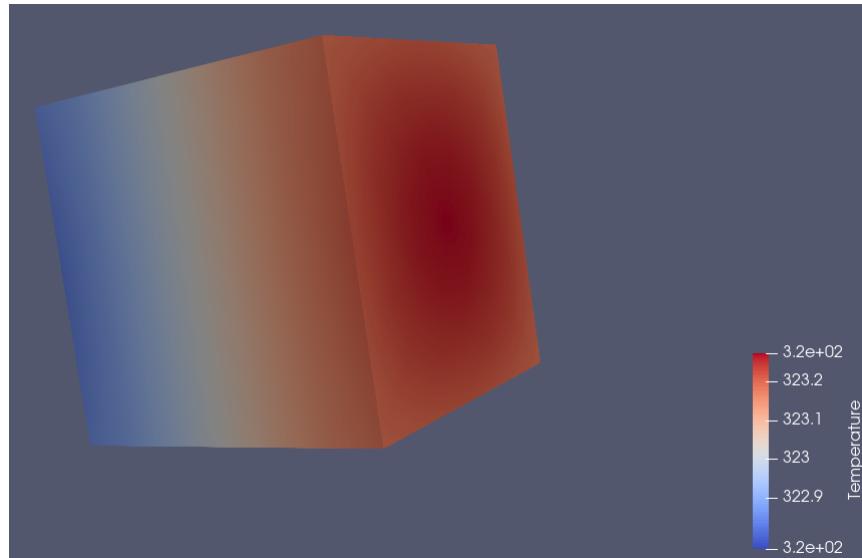


Figura 23: Vista volumétrica de temperatura - Agni B3V

Repetiendo la simulación con Agni B3V (Fig. 22) se pudo observar que a los 15000 segundos se llega al mismo resultado de 323 K (49,85 °C).

Ambos softwares convergen al mismo resultado dentro de un margen de 0,5°C. En la figura de convergencia temporal (Fig. 23) se observa como la energía adicional de la cara caliente se distribuye con el paso del tiempo, logrando así una temperatura final estable.

5.1.1.2 Prueba 2: Conducción en cubo con flujo constante

Se modeló un cubo con las siguientes características:

- Tamaño de $1\text{m} \times 1\text{m} \times 1\text{m}$
- Material aluminio, con:
 - Densidad de 2700 $\frac{\text{kg}}{\text{m}^3}$
 - Calor específico de 900 $\frac{\text{J}}{\text{kg K}}$
 - Conductividad térmica de 237 $\frac{\text{W}}{\text{K m}}$
- Las caras tienen una temperatura inicial de 0 °C
- Todo el cubo tiene un flujo constante de 200 $\frac{\text{W}}{\text{m}^2}$

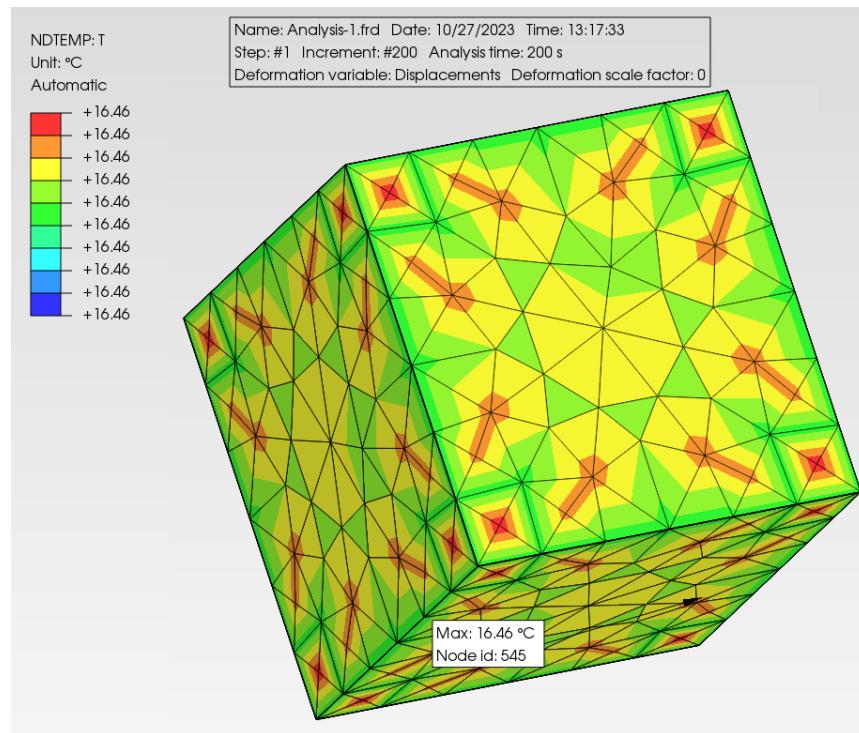


Figura 24: Vista volumétrica de temperatura - PrePoMax

En la simulación de PrePoMax (Fig. 24), luego de 200 segundos, la temperatura en todo el cubo es de 16,46 °C

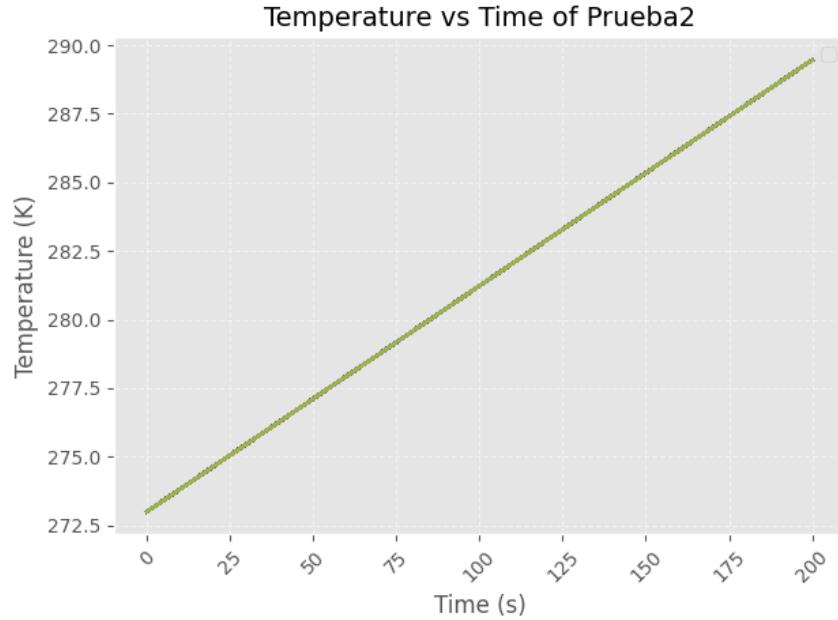


Figura 25: Evolución de temperatura de nodo 545 - Agni B3V

Con Agni B3V (Fig. 25), luego de 200 segundos de simulación se llega a una temperatura de 289,46 K (16,31 °C), lo que implica una diferencia de temperatura de 16,31 K. Como el cubo tiene la misma temperatura en todos sus puntos y el flujo es constante, la temperatura aumenta de forma lineal con el tiempo, lo que se verifica en Agni B3V.

Se presenta además una prueba teórica de este resultado. El cambio de temperatura dado un flujo en Joules se da de acuerdo a:

$$q = \rho V c \Delta T \quad (56)$$

Por otro lado, el calor recibido por una cara estará dado por:

$$q = Q A \Delta t \quad (57)$$

Donde:

- $\rho = 2700 \frac{\text{kg}}{\text{m}^3}$

- $V = 1\text{m} \times 1\text{m} \times 0,001\text{m} = 0,001\text{m}^3$
- $c = 900 \frac{\text{J}}{\text{kg K}}$
- $Q = 200 \frac{\text{W}}{\text{mm}^2}$
- $A = 1\text{m} \times 1\text{m} = 1\text{m}^2$
- $\Delta t = 200 \text{ s}$

Por ende, el cambio de temperatura tras 200 segundos será:

$$\begin{aligned}\Delta T &= \frac{Q A \Delta t}{\rho V c} \\ &= \frac{200 \frac{\text{W}}{\text{m}^2} \times 1\text{m}^2 \times 200 \text{ s}}{2700 \frac{\text{kg}}{\text{m}^3} \times 0,001\text{m}^3 \times 900 \frac{\text{J}}{\text{kg K}}} \\ &= 16,46\text{K}\end{aligned}\tag{58}$$

Resultando en una temperatura final de 289,46 K (16,31 °C).

En conclusión, a flujos constantes, Agni B3V se comporta de forma esperada al ser contrastado tanto con PrePoMax, como con la prueba teórica, dentro de un margen de error de 0,2°C.

5.1.1.3 Prueba 3: Conducción simple en cubo de dos materiales

Se modeló un cubo con las siguientes características:

- Tamaño de $1\text{m} \times 1\text{m} \times 1\text{m}$
- Una de las caras tiene como material cobre, con:
 - Densidad de $8960 \frac{\text{kg}}{\text{m}^3}$
 - Calor específico de $385 \frac{\text{J}}{\text{kg K}}$
 - Conductividad térmica de $400 \frac{\text{W}}{\text{K m}}$
- El resto de las caras tienen de material roble, con:
 - Densidad de $700 \frac{\text{kg}}{\text{m}^3}$
 - Calor específico de $2300 \frac{\text{J}}{\text{kg K}}$
 - Conductividad térmica de $0,23 \frac{\text{W}}{\text{K m}}$
- La temperatura inicial de la cara de cobre es 300°C
- La temperatura inicial de las caras de roble es 0°C
- Todo el cubo tiene un flujo constante de $200 \frac{\text{W}}{\text{m}^2}$

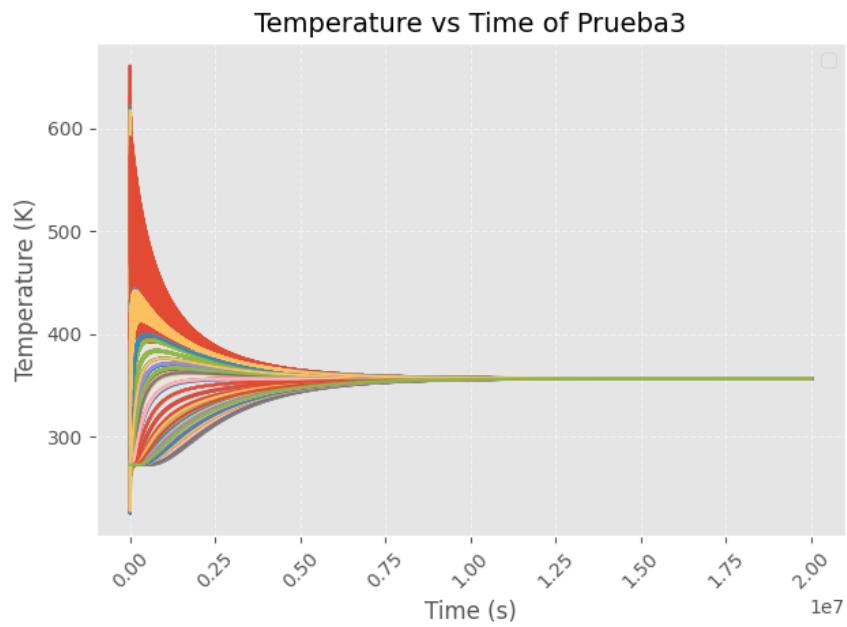


Figura 26: Evolución de la tempertaura de cada nodo - Agni B3V

Como el roble es un material con baja conductividad térmica, el sistema tarda más tiempo en converger a una temperatura estable (Fig. 26).

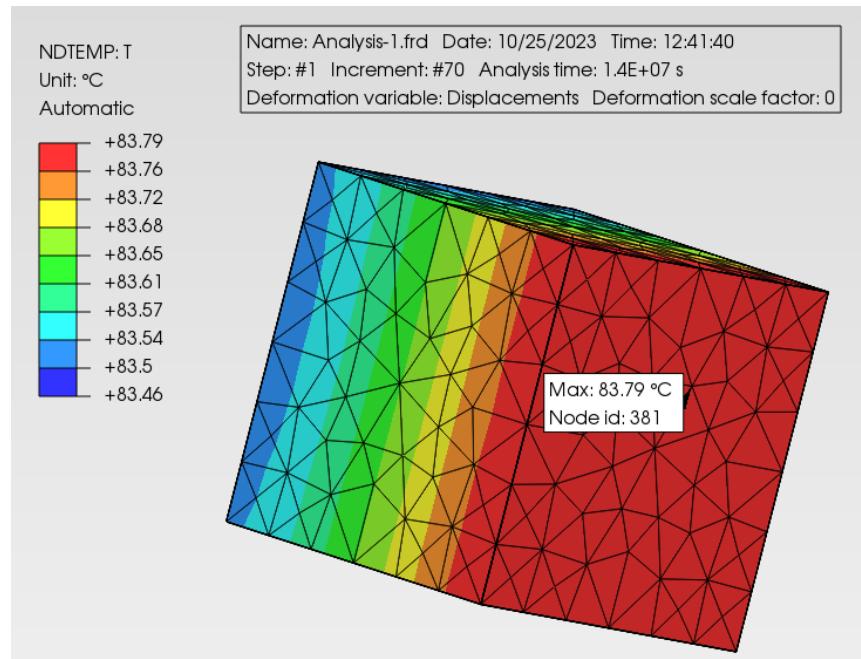


Figura 27: Vista volumétrica de temperatura - PrePoMax

Se llegó a una temperatura estable entre 83,46 °C y 83,79 °C, pasados los 2×10^7 segundos (231 días) de simulación en PrePoMax (Fig. 27).

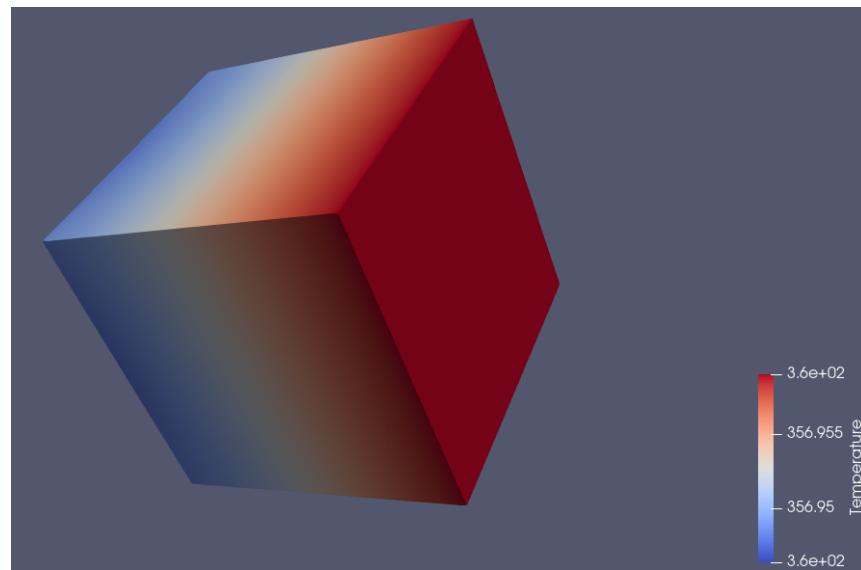


Figura 28: Vista volumétrica de temperatura - Agni B3V

Se puede observar que a los 2×10^7 segundos de simulación con Agni B3V (Fig. 28), la temperatura se estabilizó en 356,9 K (83,75 °C), una diferencia menor a 0,5°C.

5.1.2. Radiación

Se estudiará la transferencia de calor por radiación entre dos placas paralelas enfrentadas. Por un lado, se realizará un cálculo teórico considerando infinitas a ambas placas. Por el otro, se realizará una simulación con un tamaño para las placas mucho mayor al de la distancia que las separa. La radiación se restringe a la dirección de las normales, las cuales son orientadas de forma tal que puedan verse entre sí.

5.1.2.1 Prueba 1: Placas Paralelas

Se modelaron dos placas planas paralelas con las siguientes características:

- Tamaño de 1000 m × 1000 m
- Grosor de 0,001 m
- Distancia entre las placas de 0,1 m
- Las propiedades del material son:
 - Densidad de 2700 $\frac{\text{kg}}{\text{m}^3}$
 - Calor específico de 897 $\frac{\text{J}}{\text{kg K}}$
 - Conductividad térmica de 0 $\frac{\text{W}}{\text{K m}}$
- Absortancias para ambas placas igual a 1
- La temperatura inicial de la placa 1 es de 500 K
- La temperatura inicial de la placa 2 es de 300 K

Comenzando por la prueba teórica, el flujo de calor para la placa 1 será:

$$q_{\text{Total}_1} = q_{\text{Gain}_1} - q_{\text{Lost}_1} \quad (59)$$

Teniendo en cuenta la pérdida y ganancia de calor

$$\begin{aligned} q_{\text{Gain}_1} &= F_{2,1} \alpha_1 \alpha_2 \sigma A_2 T_2^4 \\ q_{\text{Lost}_1} &= \alpha_1 \sigma A_1 T_1^4 \end{aligned} \quad (60)$$

Luego, la variación de temperatura es obtenida a partir de la siguiente ecuación:

$$\begin{aligned}
\Delta T_1 &= \frac{q_{\text{Total}_1} \Delta t}{\rho V c} \\
&= \frac{\sigma A \alpha_1 (F_{2,1} \alpha_2 T_2^4 - T_1^4) \Delta t}{\rho A G c} \\
&= \frac{\sigma \alpha_1 (F_{2,1} \alpha_2 T_2^4 - T_1^4) \Delta t}{\rho G c}
\end{aligned} \tag{61}$$

Esta expresión asume que los flujos de calor ganados y perdidos por la placa permanecen constantes a lo largo del lapso de tiempo Δt . Esto no es cierto ya que, como puede observarse, los flujos dependen directamente de las temperaturas de ambas placas, las cuales a su vez se ven afectadas por los flujos de calor. No obstante, para lapsos de tiempo cortos se trata de una buena aproximación.

Entonces, calculando la diferencia de temperatura para la placa 1 transcurridos unos 10 segundos.

$$\begin{aligned}
\Delta T_1 &= \frac{5,63 \times 10^{-8} \frac{W}{m^2 K^4} \times 1 \times (1 \times 1 \times (300 \text{ K})^4 - (500 \text{ K})^4) \times 10 \text{ s}}{2700 \frac{\text{kg}}{\text{m}^3} \times 0,001 \text{ m} \times 897 \frac{\text{J}}{\text{kg K}}} \\
&= -12,65 \text{ K}
\end{aligned} \tag{62}$$

De forma análoga, para la placa 2 se tiene que:

$$\begin{aligned}
\Delta T_2 &= \frac{5,63 \times 10^{-8} \frac{W}{m^2 K^4} \times 1 \times (1 \times 1 \times (500 \text{ K})^4 - (300 \text{ K})^4) \times 10 \text{ s}}{2700 \frac{\text{kg}}{\text{m}^3} \times 0,001 \text{ m} \times 897 \frac{\text{J}}{\text{kg K}}} \\
&= 12,65 \text{ K}
\end{aligned} \tag{63}$$

Por lo que las nuevas temperaturas serían:

$$\begin{aligned}
T_1 &= 500 \text{ K} - 12,65 \text{ K} = 487,35 \text{ K} \\
T_2 &= 300 \text{ K} + 12,65 \text{ K} = 312,65 \text{ K}
\end{aligned} \tag{64}$$

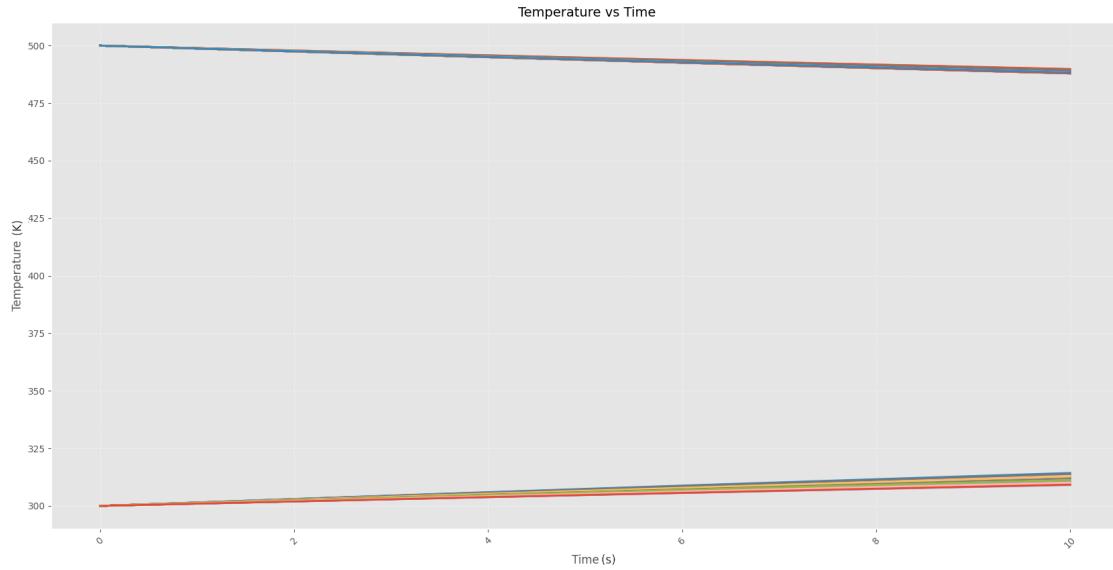


Figura 29: Evolución de la tempertaura de cada nodo - Agni B3V

Luego de 10 segundos de simulación (Fig. 29), el plano de temperatura 500 K paso a una temperatura de 488,5 K, mientras que el plano de 300 K paso a una temperatura de 311,5 K. Las temperaturas son similares, aunque existe un margen de error de 1,5 K, fácilmente atribuible a la aproximación por flujos constantes de calor antes mencionada, o al hecho de que las placas no son realmente infinitas.

5.1.3. Conclusiones

En base a las pruebas de las secciones anteriores y el anexo de resultados adicionales, los cuales cubren modelos simples y aislados, se comprueba que Agni B3V es capaz de calcular la temperatura de un mallado dados los materiales y los flujos incidentes con errores menores a 1 K.

5.2. Conservación de la Energía

Se calculó la energía del sistema para cada instante de tiempo, considerando a esta como la sumatoria de la energía de cada nodo en base a:

$$E = T \ c \ \rho \ G \ A \quad (65)$$

Siendo

- T la temperatura del nodo.
- c el calor específico del material.

- ρ la densidad del material.
- G el grosor del material.
- A el área del nodo.

El área del nodo se calcula como la sumatoria de las áreas de los elementos a los que pertenece el nodo, dividido 3.

Luego se realiza la resta de las energías entre instantes consecutivos para obtener la diferencia de energías. A lo que se le suma el flujo (entrante y saliente) entre esos instantes para cada nodo. A partir de esta información, se realizó el siguiente gráfico:

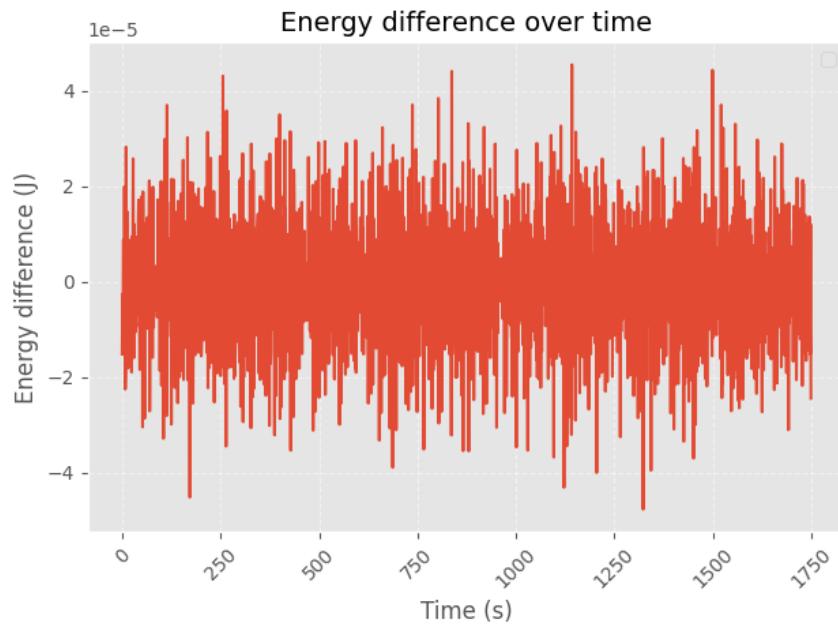


Figura 30: Diferencias de energía en el tiempo - Agni B3V

Como se puede observar (Fig. 30), la diferencia entre los niveles de energía del sistema instante a instante se encuentra en valores sumamente bajos, inferiores a 1×10^{-4} J. Por ende, el modelo realizado conserva la energía del sistema a lo largo del tiempo.

5.3. Convergencia

Se realizaron simulaciones para distintas condiciones de mallado y paso de tiempo sobre un modelo de geometría compleja (Fig. 31).

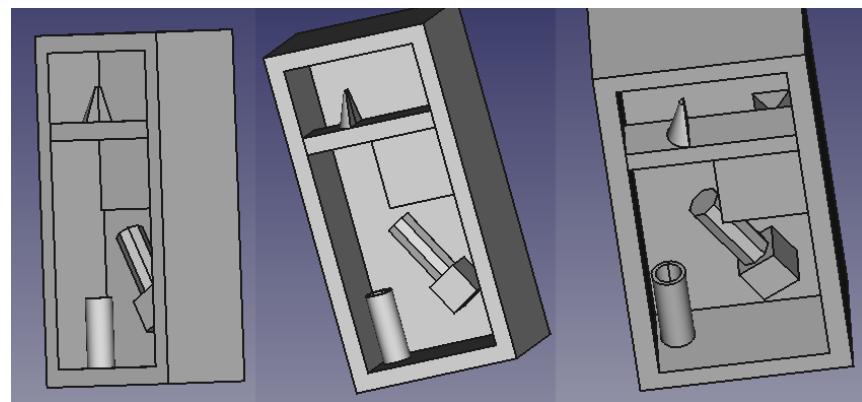


Figura 31: Vista del Modelo

Las propiedades físicas del modelo son:

- Material con:
 - Espesor de 0,05 m,
 - Densidad de $2700 \frac{\text{kg}}{\text{m}^3}$,
 - Calor específico de $900 \frac{\text{J}}{\text{kg K}}$,
 - Conductividad térmica de $237 \frac{\text{W}}{\text{K m}}$,
 - Absortividad en espectro solar de 1,
 - Absortividad en espectro infrarrojo de 1.
- Temperatura inicial de 293,15 K,
- Sin flujos,
- Factor de Albedo de 1,
- Radiación Infrarroja de la Tierra de $225 \frac{\text{W}}{\text{m}^2}$,
- Constante Solar de $1361 \frac{\text{W}}{\text{m}^2}$,
- Órbita con las siguientes características:
 - SMA de 7000 km,
 - ECC de 0,
 - INC de 0,

- RAN de 0,
- AOP de 0,
- TA de 0.

La órbita es Sun Pointing, con el Sol apuntando hacia el modelo del mismo modo que el espectador ve al modelo en la figura (Fig. 32).

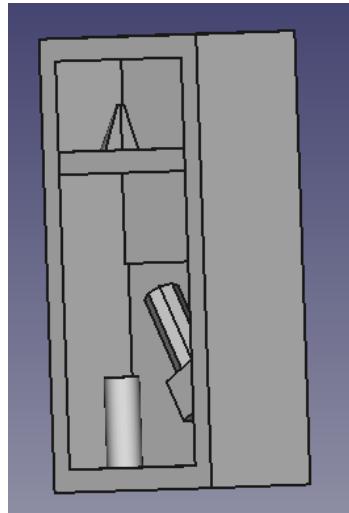


Figura 32: Vista del Modelo desde el Sol

Con el objetivo de realizar un estudio se eligió un punto arbitrario, correspondiente al centro del panel frontal (Fig. 33).

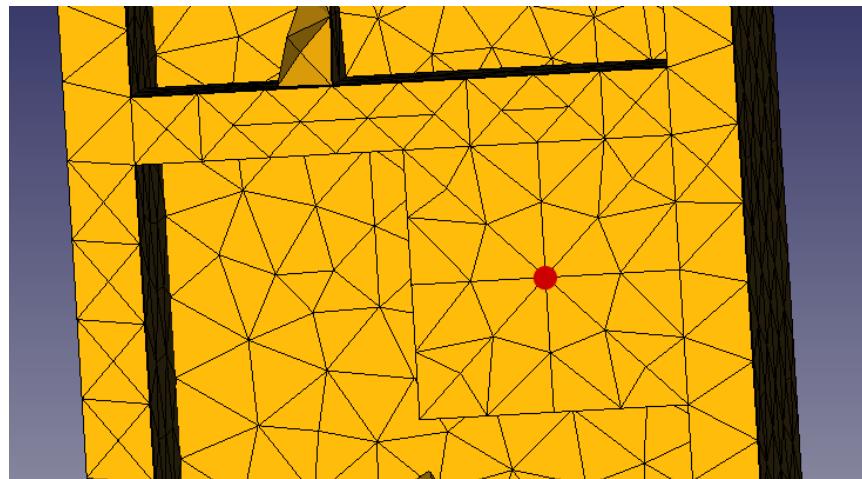


Figura 33: Centro del Panel del modelo

5.3.1. Convergencia en Tiempo

Se realizaron simulaciones para un mallado con elementos de lado promedio de 0,687 m. Con pasos de tiempo de 1, 10, 100 y 1000 segundos y números de Courant-Friedrichs-Levy 0,00284; 0,0284; 0,284 y 2,84 respectivamente. Se tomó como caso base la simulación que hace uso de un paso de tiempo de 1 segundo para comparar el resto de los resultados obtenidos.

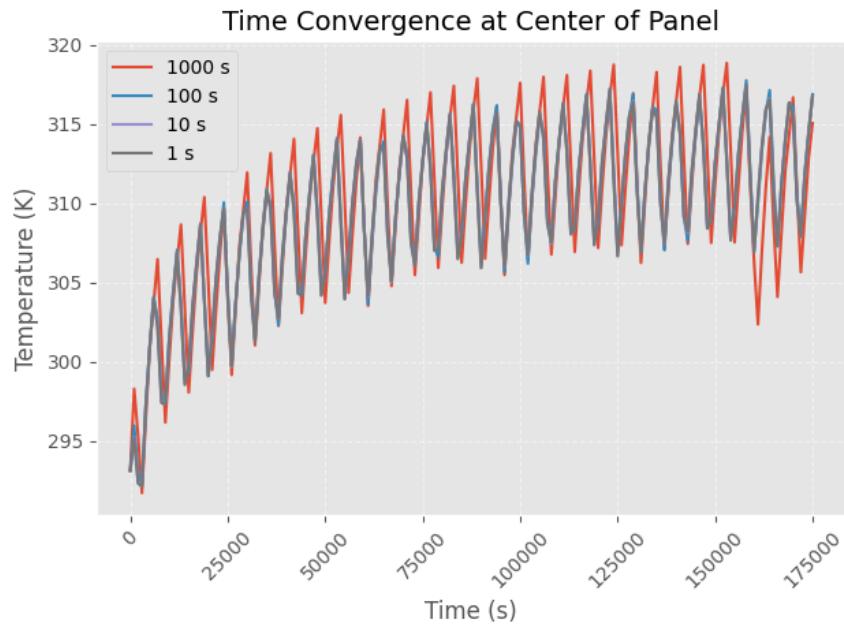


Figura 34: Convergencia en tiempo - Centro del Panel

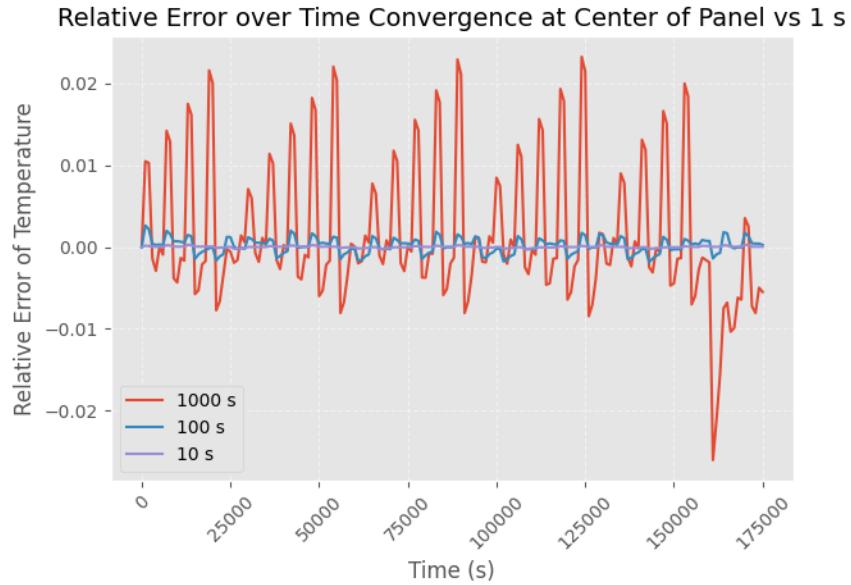


Figura 35: Error relativo de la convergencia en tiempo - Centro del Panel

Como se observa de las figuras (Fig. 34) (Fig. 35), el error del modelo debido a la utilización de distintos pasos de tiempo no es apreciable hasta los 1000 segundos, donde el número de Courant es mayor a uno.

5.3.2. Convergencia en Malla

Se realizó una simulación con paso de tiempo de 1 segundo y un mallado de 622 elementos, la mínima cantidad necesaria para representar correctamente el modelo bajo FEM. Luego, se duplicó la cantidad de elementos y se ejecutaron simulaciones con mallas de 1178, 2841 y 5539 elementos, manteniendo constante el paso de tiempo. Aquí los números de Courant-Friedrichs-Levy son muy cercanos a cero, por lo que no se consideran. Los cálculos de error fueron realizados tomando como base la malla de mayor cantidad de elementos.

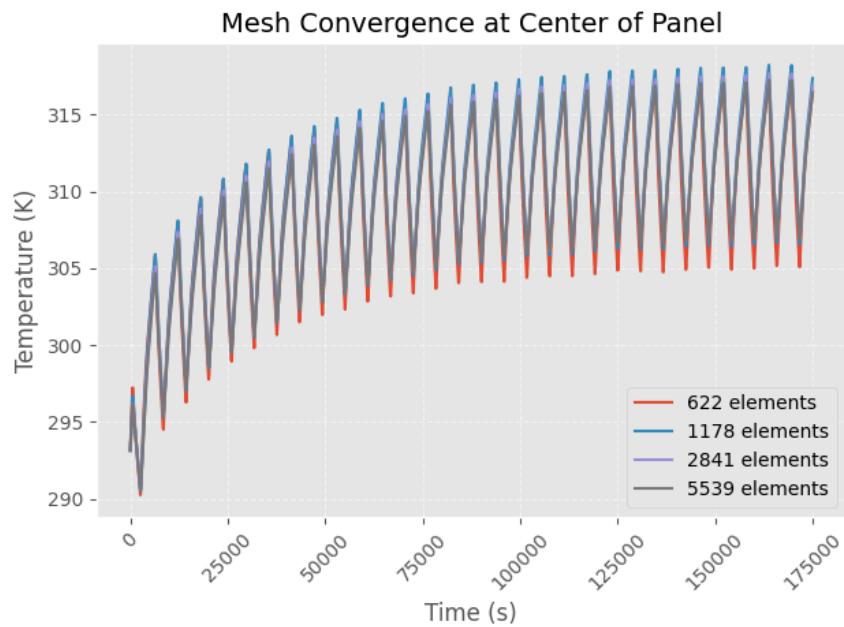


Figura 36: Convergencia en malla - Centro del Panel

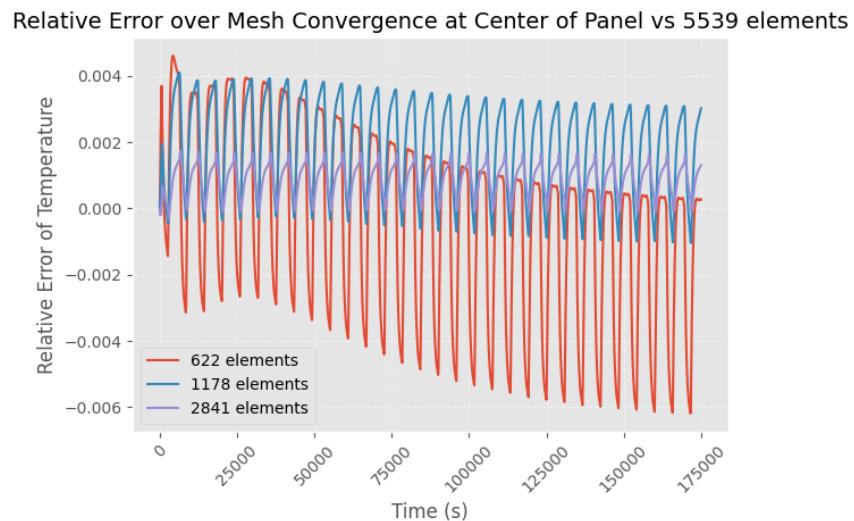


Figura 37: Error relativo de la convergencia en malla - Centro del Panel

Como es de esperarse, la disminución del error de simulación se observa a medida que incrementa la cantidad de elementos en la malla, ya que se perfecciona la discretización de la geometría modelada

(Fig. 36) (Fig. 37). No obstante, aun con 622 elementos el modelo numérico parece converger, lo que es destacable para una malla tan compleja.

5.4. Tiempo de Ejecución

Se realizaron pruebas en las cuales se comparó el tiempo de ejecución total y parcial del software bajo distintas configuraciones de hardware (Cuad. 2).

Equipo	Componentes		
	Procesador	Tarjeta Gráfica	Memoria RAM
PC1	AMD Ryzen 5 5600G 6C-12T 4,4GHz	AMD Radeon RX 6650 XT	16 GB 3200 MHz
PC2	AMD Ryzen 9 3900X 12C-24T 4,6GHz	AMD Radeon RX 5600 XT	16 GB 3200 MHz
PC3	Intel Core i7 8700 6C-12T 4,6GHz	Nvidia GTX 1080 Ti	16 GB 2400 MHz
PC4	Intel Core i7-9700F 8C-8T 4,7GHz	Nvidia GT 730	16 GB 2400 MHz

Cuadro 2: Componentes de los equipos utilizados.

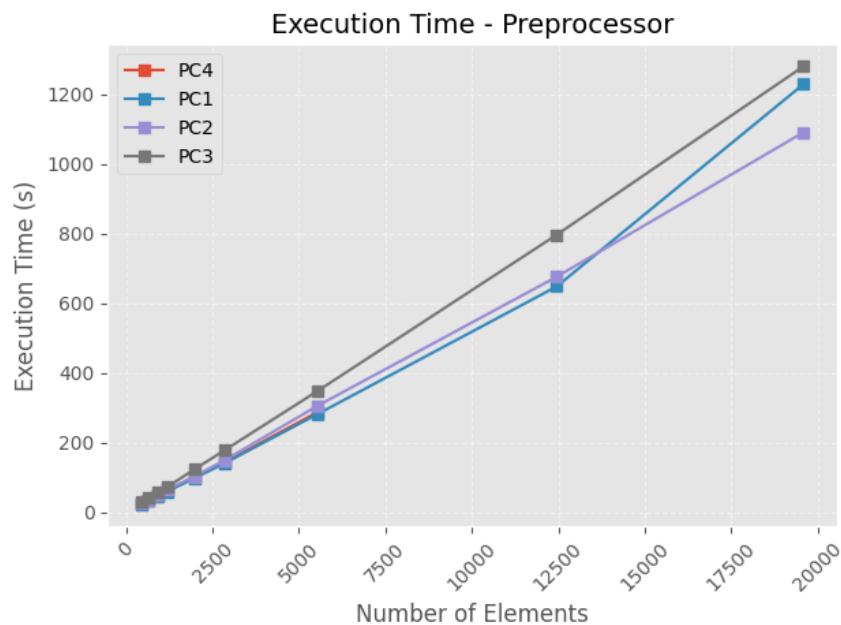


Figura 38: Tiempo de ejecución del Preprocesador

El tiempo de ejecución del preprocesador (Fig. 38) escala linealmente con la cantidad de elementos, ya que la mayor parte de las operaciones consisten en emitir rayos desde cada elemento. El cálculo de radiación elemento a elemento escala cuadráticamente y es esperable que domine la tendencia del sistema general para una cantidad muy grande de elementos, pero esta función es a su vez la que mejor aprovecha los núcleos del CPU.

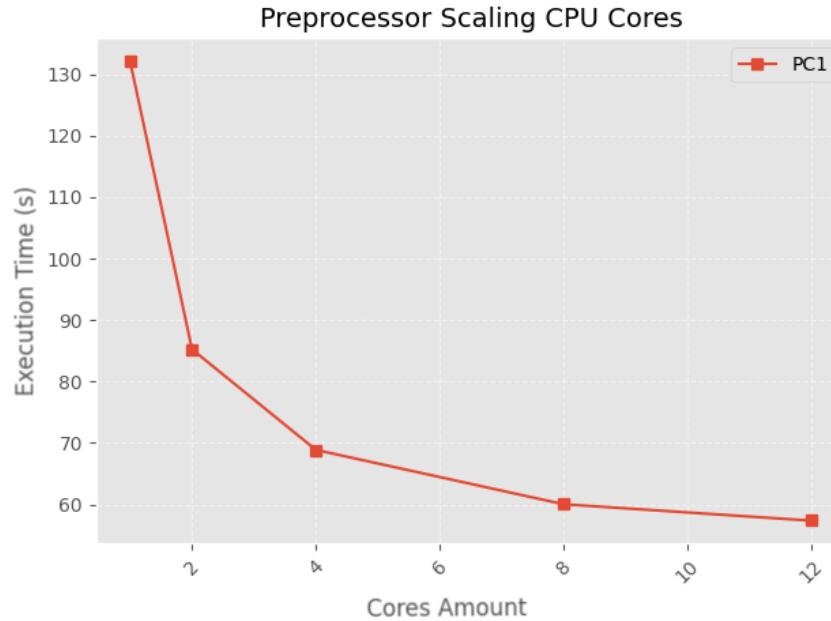


Figura 39: Escalado de Preprocesador según cores de la CPU

Se observa (Fig. 39) como, conforme incrementan la cantidad de núcleos de CPU disponibles, se reduce significativamente el tiempo total de ejecución del preprocesador. Es importante notar que el rendimiento ganado decrece a medida que se suman núcleos, dado que priman las funciones estrictamente secuenciales.

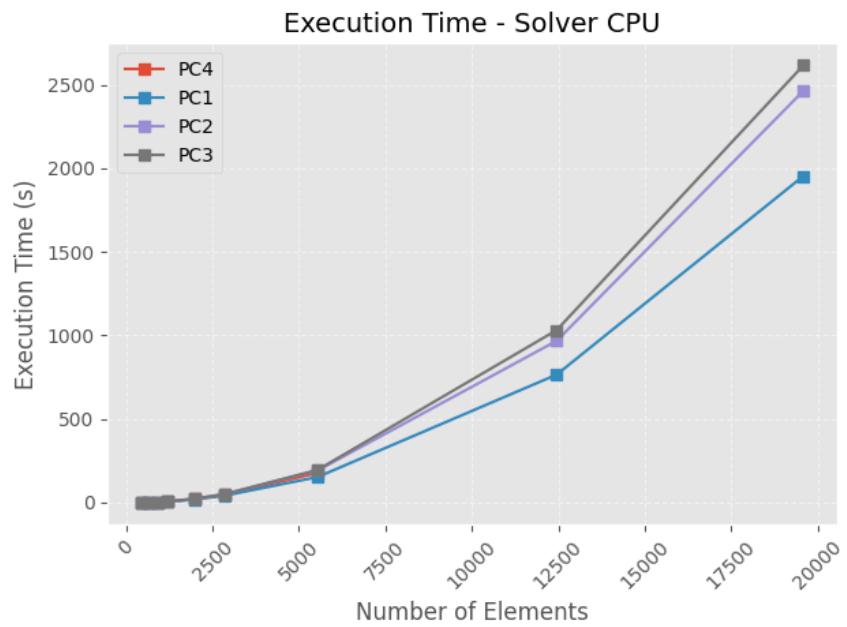


Figura 40: Tiempo de ejecución del Solver en CPU

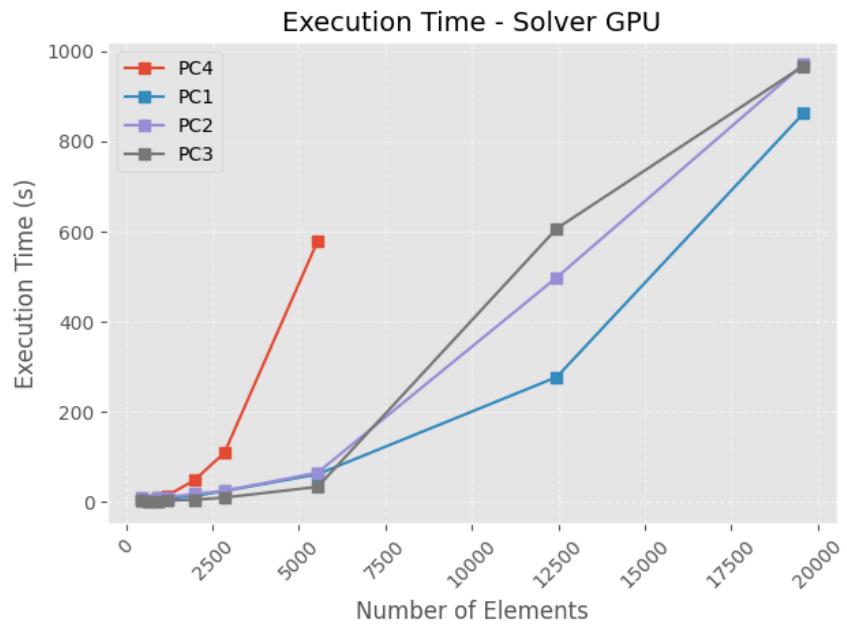


Figura 41: Tiempo de ejecución del Solver en GPU



Figura 42: Tiempo de ejecución total en CPU



Figura 43: Tiempo de ejecución total en GPU

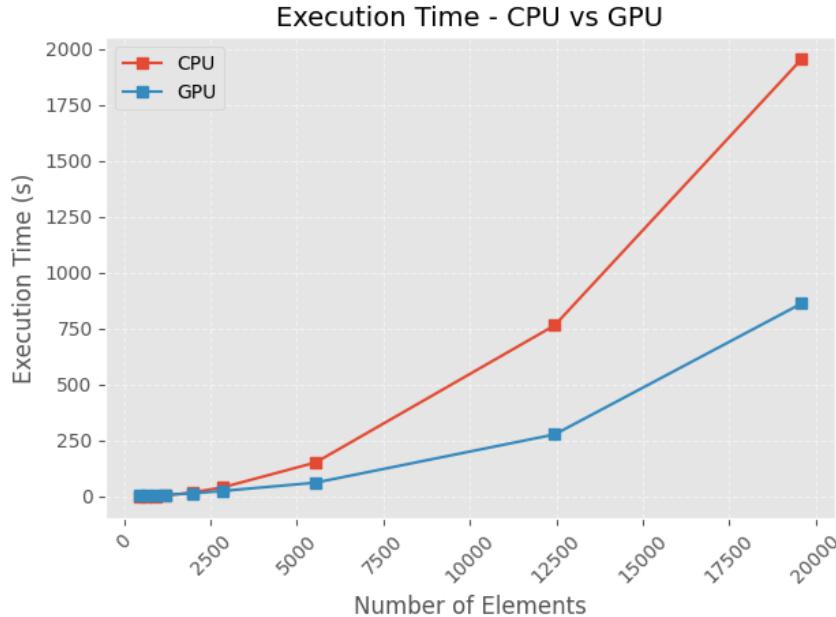


Figura 44: Tiempo de ejecución Solver CPU vs GPU

En cuanto al solver, la complejidad esperada es cúbica, pero gracias a los esfuerzos de optimización puede mitigarse el impacto en la performance lo suficiente para que no empeore la tendencia que ya exhibe el preprocesador (Fig. 40) (Fig. 42). Se percibieron speedups de hasta 2,5 en casi todas las configuraciones de hardware expuestas al utilizar la GPU (Fig. 41) (Fig. 43) (Fig. 44). Al observar las figuras con detenimiento pueden observarse “codos” en los tiempos de ejecución. Esto ocurre porque, al alcanzarse el límite de núcleos bajo uso de las GPUs, debe fragmentarse el trabajo en al menos dos etapas de trabajo secuenciales. El mejor ejemplo de ello es la PC4, que por presentar una tarjeta de video con cantidad de núcleos muy reducidos pasa a comportarse secuencialmente y es superada con creces por el propio CPU instalado.

5.5. Profiling

Continuando el objetivo de la sección anterior, se realizará un análisis más detallado de las secciones del código que se ejecutan con mayor frecuencia y que presentan un costo de cómputo lo suficientemente grande para justificar esfuerzos en su optimización. No es relevante detenerse en el tiempo, sino en la proporción del tiempo total que significa.

```

ncalls  tottime  cumtime  filename:lineno(function)
    71859  22.816  22.816 {method 'run' of 'embreeex.rtcore_scene.EmbreeScene'
                           objects}
    71858   7.042  7.042 {method 'normal' of 'numpy.random.mtrand.RandomState'
                           objects}
    71858   5.481 11.012 elements.py:27(random_points_in_element)
    257804   5.102  5.102 {method 'reduce' of 'numpy.ufunc' objects}
                           60  3.988 69.866 view_factors.py:21(element_earth)
    71859   1.998 25.445 ray_pyembree.py:336(run)
    73036   1.831  1.831 {method 'rand' of 'numpy.random.mtrand.RandomState'
                           objects}
    71858   1.768  1.768 vector_math.py:84(orient_towards_direction)
    71858   1.737 12.844 vector_math.py:47(random_unit_vectors)
    292288   1.544  1.544 {built-in method numpy.array}
    147394   1.330  3.952 caching.py:694(<listcomp>)
    147394   1.261  6.186 caching.py:681(__hash__)
    70680   1.203  1.576 vector_math.py:92(_rotation_matrix)
    71861   1.179  3.975 linalg.py:2383(norm)
                           1  1.117 6.172 view_factors.py:127(element_element)
    183704   1.101  4.384 fromnumeric.py:71(_wrapreduction)
    183700   0.830  5.455 fromnumeric.py:2177(sum)
    71860   0.811  2.028 copy.py:128(deepcopy)
    295116   0.781  1.872 {built-in method builtins.hash}
    592298   0.749  0.749 {built-in method builtins.hasattr}
    294786   0.733  1.091 caching.py:209(__hash__)
    167763   0.658  0.658 vector_math.py:55(array_dot)
    71859   0.644 35.540 constants.py:145(timed)
    70681   0.623 30.744 ray_pyembree.py:248(intersects_first)
    70680   0.570  2.234 vector_math.py:112(flip_around_axis)
    1178   0.568  3.514 ray_pyembree.py:111(intersects_id)

```

Figura 45: Profiling de Preprocessdor

El reporte de cProfiler (Fig. 45) indica que la construcción de la escena de Pyembree (preparado de mesh y raycasting) es por lejos la tarea que mayor tiempo requiere. Le siguen funciones internas de Numpy, a las que se delega las operaciones entre vectores y matrices, y que ya son resueltas eficientemente en el backend de C. Finalmente aparecen dos funciones del preprocesador: `element_earth` y `random_points_in_element`. La primera de ellas no es más que el fragmento de código en donde se calcula la interacción elemento Tierra, que como se mencionó con anterioridad es efectivamente donde más tiempo se invierte. La segunda produce puntos al azar en el interior de un triángulo, y es empleada como punto de partida de los rayos. La misma ya ha recibido optimizaciones motivadas por la frecuencia con la que se invoca, pero aun así sería interesante explorar si cambiarla por una disposición uniforme determinística lograría mejorar la performance sin afectar la calidad de los resultados.

```

Overhead Symbol
42,04% nalgebra::base::ops::<impl core::ops::arith::Mul<&nalgebra::base::
    matrix::Matrix<T,R2,C2,SB>> for &nalgebra::base::matrix::M
39,44% nalgebra::linalg::lu::LU<T,D,D>::solve
4,33% core::num::flt2dec::strategy::grisu::format_shortest_opt
3,05% core::fmt::write
1,32% __memmove_avx_unaligned_erms
1,14% ::fem::implicit_::ImplicitSolver::step
1,07% core::fmt::Formatter::pad_integral
0,92% <&mut W as core::fmt::Write>::write_str
0,90% core::fmt::num::imp::<impl core::fmt::Display for u32>::fmt
0,75% core::fmt::float::float_to_decimal_common_shortest
0,45% 0xffffffff83aad633

```

Figura 46: Profiling de Solver CPU

El solver fue cuidadosamente programado para expresar el problema matricialmente y delegar las operaciones algebraicas a la biblioteca Nalgebra, en cuyas funciones se invierte casi el 82 % del tiempo de ejecución, según el reporte de Perf (Fig. 46). Para mejorar la performance podrían evaluarse un eventual cambio de algoritmo o bien continuar los esfuerzos de paralelización en GPU que, como se mostró con anterioridad, ya han sido fructíferos.

```

Overhead Shared Object/ Symbol
11,33% libhsa-runtime64.so.1.5.50102 / 000000000004d2c9
6,55% solver / core::fmt::float::float_to_decimal_common_shortest
6,35% solver / core::num::flt2dec::strategy::grisu::format_shortest_opt
6,03% libamd_comgr.so.2.4.50102 / 00000000041a05b5
5,89% libhsa-runtime64.so.1.5.50102 / 000000000004d2fb
5,85% libhsa-runtime64.so.1.5.50102 / 000000000004d2df
5,25% solver / core::fmt::write
4,48% libhsa-runtime64.so.1.5.50102 / 000000000004d2ea
3,15% solver / <&mut W as core::fmt::Write>::write_str
2,77% libc.so.6 / _IO_str_init_static_internal
2,58% solver / core::fmt::num::imp::<impl core::fmt::Display for u32>::fmt
1,85% libhsa-runtime64.so.1.5.50102 / 000000000004d2f5
1,75% solver / core::fmt::Formatter::pad_integral
1,64% solver / <core::fmt::Formatter as core::fmt::Write>::write_fmt
1,51% libhsa-runtime64.so.1.5.50102 / 000000000004d2cd
1,39% libamdocl64.so / 0000000000dab24
1,37% libc.so.6 / __memmove_avx_unaligned_erms
1,26% [kernel.kallsyms] / ffffffff83496e96
1,10% libhsa-runtime64.so.1.5.50102 / 000000000004d2d0
1,06% libhsa-runtime64.so.1.5.50102 / 000000000004d2f0
0,98% libamdocl64.so / 000000000008c79b

```

Figura 47: Profiling de Solver GPU

El reporte de Perf para la variante del solver que aprovecha la GPU (Fig. 47) incurre en imprecisiones sobre el tiempo total, dado que lo mide desde la CPU, pero nos permite investigar las operaciones más costosas que aún se realizan en CPU. Los símbolos cuyo Shared Object pertenece al runtime de libhsa se vinculan al proceso de delegación de trabajo en GPU y en suma representan el 33 % del tiempo de ejecución total de CPU. Los símbolos que se agrupan en fmt se vinculan al formateo de escritura de estructuras internas. Se intentó paralelizar en GPU la inversión de la matriz con prometedores resultados, pero en última instancia no fue posible encontrar la causa de un problema de concurrencia.

5.6. Consumo de Memoria

El tipo de software desarrollado tiende hacia intensividad en cómputo más que en memoria. La memoria disponible no fue un limitante en las pruebas realizadas, salvo contadas ocasiones, por lo que en general se ha dejado de lado, en favor del incremento de la performance. A continuación se enseña la tendencia general del máximo consumo de memoria por cantidad de elemento (medido con la opción verbose de time) y un detalle de la memoria consumida instantáneamente para un caso particular (medido con la herramienta massif de Valgrind).

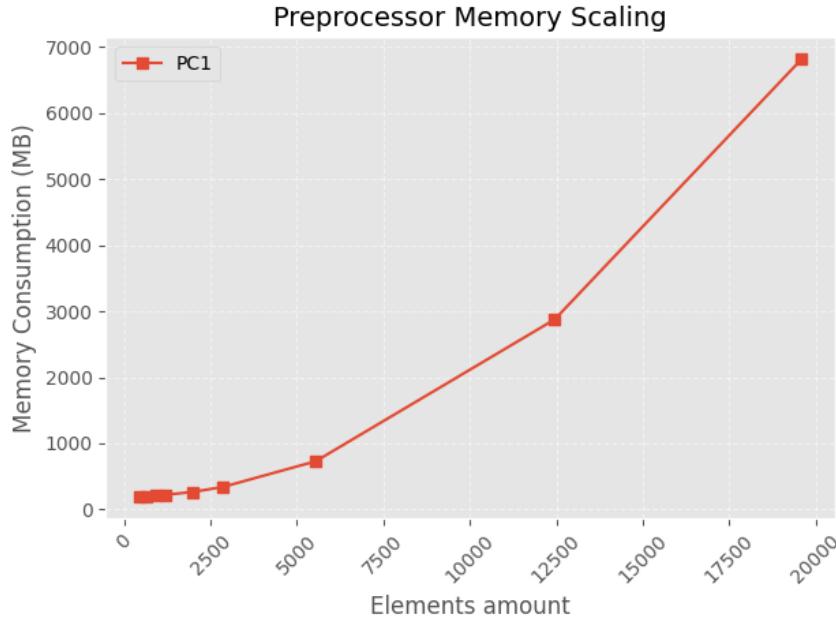


Figura 48: Tendencia de Máximo Consumo de Memoria - Preprocesador

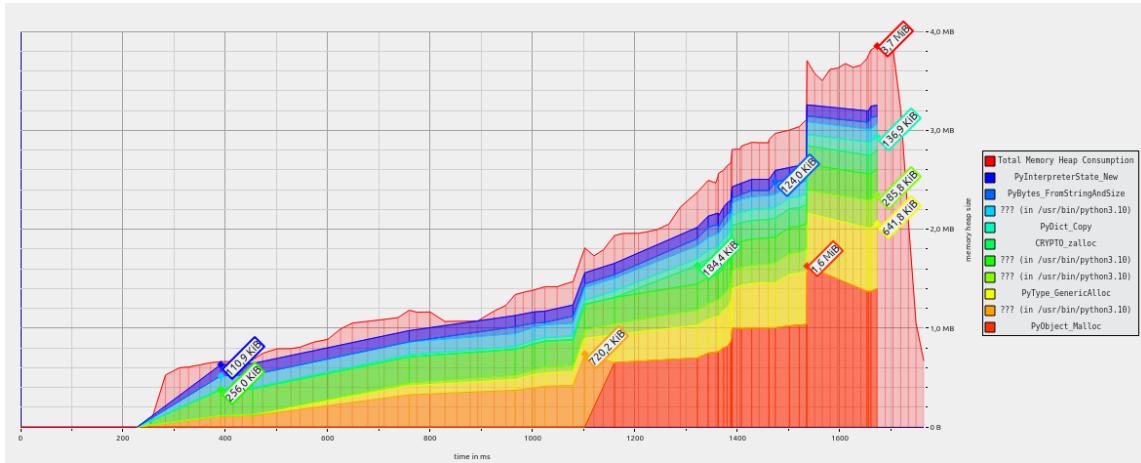


Figura 49: Detalle de Consumo de Memoria - Preprocesador

En el preprocesador, los factores de vista de todas las interacciones se almacenan en memoria hasta el final de la ejecución, en donde son volcados en un archivo binario. La tendencia del consumo de memoria es cuadrática (Fig. 48) (Fig. 49) debido a la interacción elemento-elemento, pero la interacción elemento-Tierra es la que más memoria requiere, ya que debe computarse para cada punto de la órbita. Almacenar los factores de vista inmediatamente tras su cálculo debería

ser sencillo y controlaría la creciente necesidad de memoria RAM, simplemente no ha sido una optimización prioritaria.

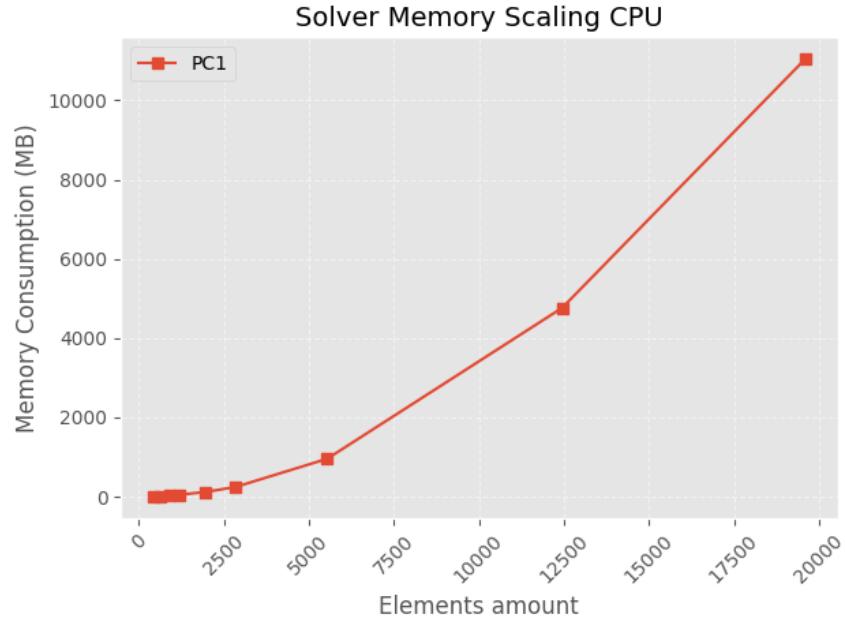


Figura 50: Tendencia de Máximo Consumo de Memoria - Solver CPU

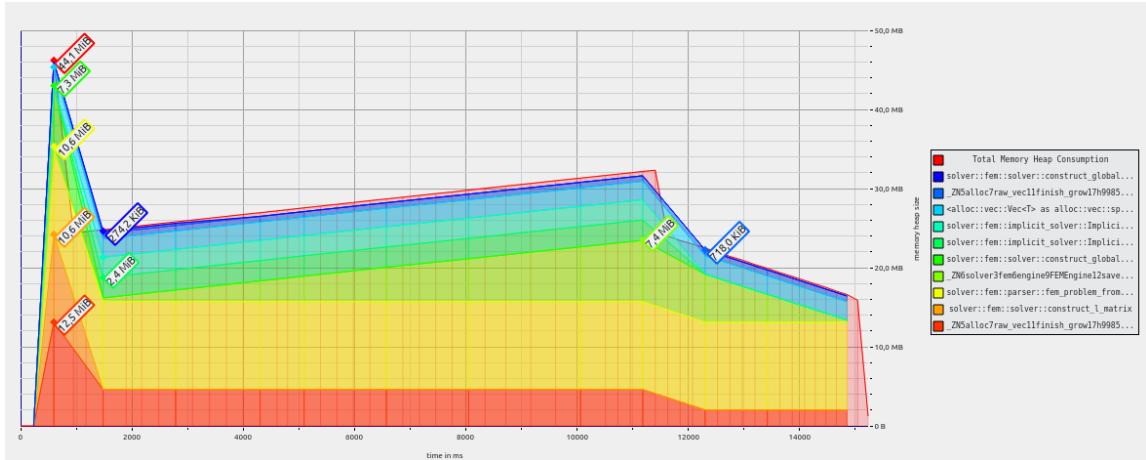


Figura 51: Detalle de Consumo de Memoria - Solver CPU

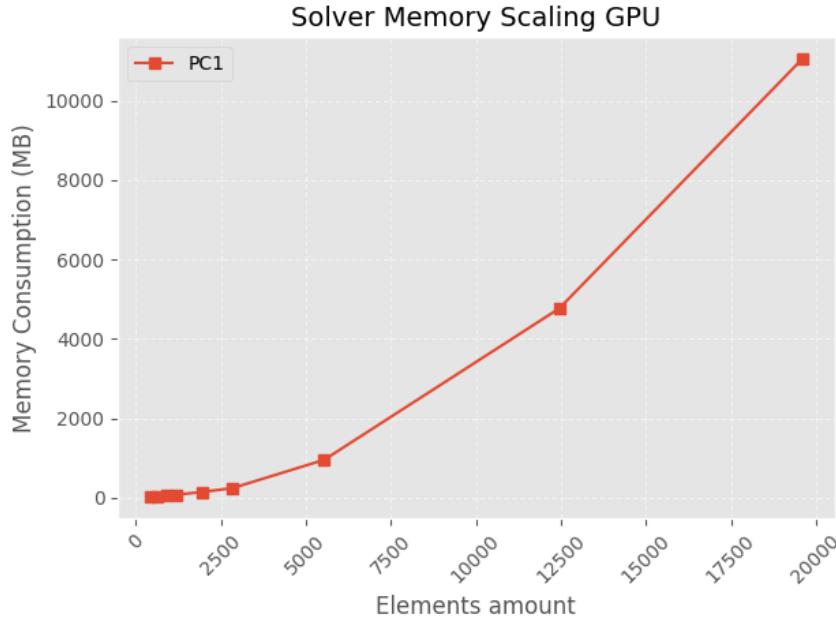


Figura 52: Tendencia de Máximo Consumo de Memoria - Solver GPU

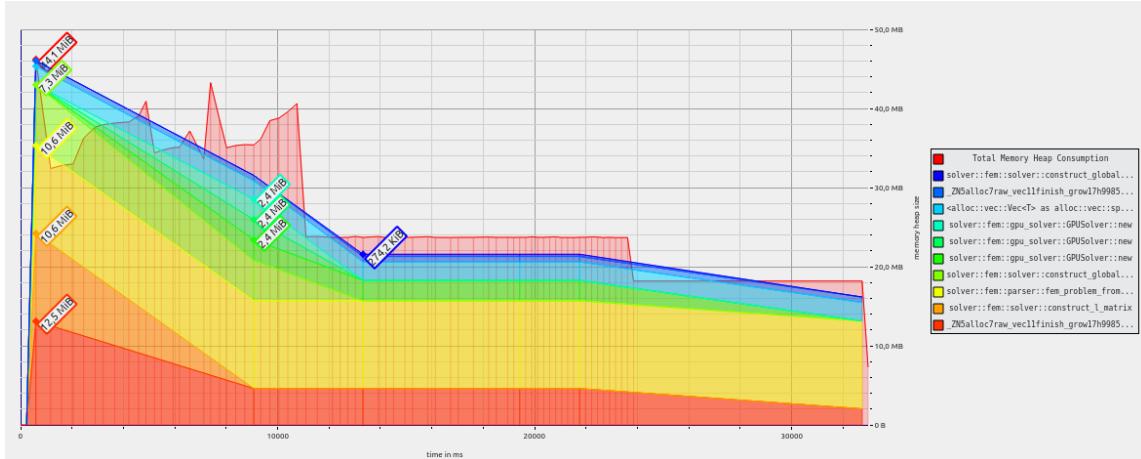


Figura 53: Detalle de Consumo de Memoria - Solver GPU

En el solver es mucho más importante mantener resultados en memoria para reducir el overhead de transferencia de datos desde y hacia la GPU. No obstante, aquí se emplean muchas matrices que escalan cuadráticamente y pese a que se almacenan en disco los resultados progresivamente, ya se requieren 1 Gb para 5000 elementos (Fig. 50) (Fig. 51). No existe gran diferencia entre el consumo de memoria del modo en CPU y GPU, pero en el segundo (Fig. 52) (Fig. 53) es ligeramente inferior

para pocos elementos, posiblemente debido al uso de buffers en GPU que no son medidos por las herramientas empleadas.

5.7. Validación con Expertos

Fue posible coordinar una reunión en el Centro Tecnológico Aeroespacial (CTA), de la Universidad Nacional de la Plata para exponer el sistema ante profesionales familiarizados con los problemas asociados a la construcción de un satélite y las herramientas de software que asisten a su resolución. Allí se expusieron filminas presentando Agni B3V y se discutió el alcance, modelo y organización del software. Más adelante se acompañó a uno de los integrantes del equipo de CTA por un flujo de muestra del software y se profundizó en aspectos de usabilidad y escenarios cubiertos.

La recepción general fue positiva. Se valoraron particularmente el mallado automático y la capacidad de escalamiento. Entre los puntos a mejorar más importantes se destacaron: Soportar otras aptitudes, permitir uniones entre elementos con conductividades específicas y presentar los resultados en componentes del dominio (batería, estructura, antena, etc.).

6. Conclusión

De la retroalimentación de los expertos y el encuentro en la Universidad Nacional de la Plata se tiene la certeza de que un profesional puede configurar y simular un escenario, limitado por los supuestos asumidos, con facilidad. El modelo numérico y el código en principio podrían extenderse sin inconvenientes para cubrir otras órbitas o aptitudes, aunque no sería sencillo obtener datos para su validación. La escasez de resultados publicados, documentación de diseño y de aproximaciones numéricas ha sido el mayor desafío para llevar adelante este trabajo. No obstante, la metodología que se adoptó a lo largo del desarrollo permite suponer que el software debiera comportarse de forma correcta bajo escenarios de mayor complejidad.

Modificar el modelo para que soporte la creación de volúmenes sólidos y materiales anisotrópicos, o con propiedades físicas que varíen a lo largo del tiempo, requerirían de una reformulación más profunda del modo en que se acoplan conducción y radiación. Además, la experiencia de usuario aún puede mejorarse añadiendo funcionalidades a nivel de GUI, como puede ser la comunicación con el preprocesador y el solver para mostrar el progreso en la interfaz gráfica.

En cuanto al rendimiento del sistema, ha quedado demostrado que existe un amplio margen de ganancia al parallelizar en GPU, a nuestro entender inexplorado por los simuladores disponibles. En las tendencias de coste de cómputo nuestro análisis resultó acertado, en tanto el solver cubre una porción creciente del tiempo de ejecución total. Aun así, dado que el rango de cero a quince mil elementos es usual en la industria y suficiente para la enseñanza, sería atractivo que el preprocesador pudiese aprovechar también la GPU. Para mejorar la velocidad de cómputo del solver, se podría considerar una paralelización más intensiva de las operaciones que realiza. Se podría resolver en GPU la inversión de la matriz o explorar el aprovechamiento de todos los núcleos de la CPU para ejecuciones que no interfieran con la GPU, optimizaciones que se intentaron, pero tuvieron que abandonarse por limitaciones de tiempo y la necesidad de disminuir el error de la simulación.

Aún hace falta incorporar características y ajustar las existentes para que Agni B3V pueda competir efectivamente con los productos comerciales a la hora de asistir en la construcción precisa de un satélite. Al margen de ello, se pudieron cumplir todos los objetivos del trabajo y se desarrolló un sistema flexible y usable que en su estado actual ya es capaz de guiar el análisis de viabilidad de proyectos pequeños y asistir la enseñanza académica. Se espera que la documentación de las decisiones de diseño y su fundamento permitan a futuros interesados en el área construir proyectos libres y abiertos sobre las bases planteadas.

Agradecemos el acompañamiento del tutor y los expertos de consulta, sin los que el trabajo no hubiese podido madurar hasta el punto en el que se encuentra.

7. Bibliografía

Referencias

- [1] National Aeronautics and Space Administration. «Measuring Earth's Albedo.» (2014), dirección: <https://earthobservatory.nasa.gov/images/84499/measuring-earths-albedo>.
- [2] H. G. Işık, A. B. Uygur, C. Ömür y E. Solakoğlu, «The thermal analysis of a satellite by an in-house computer code based on Thermal Network Method and Monte Carlo Ray Tracing Technique,» en *Proceedings of 5th International Conference on Recent Advances in Space Technologies - RAST2011*, 2011. dirección: <https://doi.org/10.1109/RAST.2011.5966988>.
- [3] Pixologic. «ZBrush.» (1999), dirección: <https://www.maxon.net/es/zbrush>.
- [4] P. Losch y C. Losch. «Cinema4D.» (1990), dirección: <https://www.maxon.net/es/cinema-4d>.
- [5] Alias Systems Corporation. «Maya.» (1998), dirección: <https://www.autodesk.es/products/maya/>.
- [6] T. Roosendaal. «Blender.» (1994), dirección: <https://www.blender.org/>.
- [7] J. Riegel. «FreeCAD.» (2001), dirección: <https://www.freecad.org/>.
- [8] M. Kintel y C. Wolf. «OpenSCAD.» (2010), dirección: <https://openscad.org/>.
- [9] Google. «Sketchup.» (2000), dirección: <https://www.sketchup.com/es>.
- [10] AutoDesk. «AutoCAD.» (1982), dirección: <https://www.autodesk.es/products/autocad>.
- [11] C. Geuzaine y J.-F. Remacle. «Gmsh: a three-dimensional finite element mesh generator with built-in pre- and post-processing facilities.» (1997), dirección: https://gmsh.info/doc/preprints/gmsh_paper_preprint.pdf.
- [12] J. Schöberl. «Netgen.» (1997), dirección: <https://ngsolve.org/>.
- [13] National Aeronautics and Space Administration. «42.» (2015), dirección: <https://software.nasa.gov/software/GSC-16720-1>.
- [14] National Aeronautics and Space Administration. «GMAT.» (2007), dirección: <https://software.nasa.gov/software/GSC-18094-1>.
- [15] Nvidia. «CUDA.» (2007), dirección: <https://developer.nvidia.com/cuda-toolkit>.
- [16] Khronos Group. «OpenCL.» (2009), dirección: <https://www.khronos.org/opencl>.
- [17] Khronos Group. «Vulkan.» (2016), dirección: <https://www.vulkan.org>.
- [18] Architecture Review Board. «OpenGL.» (1992), dirección: [https://www.opengl.org/](https://www.opengl.org).
- [19] Khronos Group. «COLLADA.» (2004), dirección: <https://www.khronos.org/collada>.
- [20] Z. Qian-Yi, P. Jaesik y K. Vladlen. «Open3D: A Modern Library for 3D Data Processing.» (2018).
- [21] R. Cabello. «Three.js.» (2010), dirección: <https://threejs.org>.
- [22] H. e. a. Dawson. «Trimesh.» (2019), dirección: <https://trimesh.org>.
- [23] Microsoft. «Microsoft Excel.» (1985), dirección: <https://www.microsoft.com/es-mx/microsoft-365/excel>.

- [24] The Document Foundation. «LibreOffice Calc.» (2010), dirección: <https://es.libreoffice.org/descubre/calc/>.
- [25] Kitware Inc y Los Alamos National Laboratory. «ParaView.» (2002), dirección: <https://www.paraview.org/>.
- [26] Enthought Inc. «MayaVi.» (2019), dirección: <https://docs.enthought.com/mayavi/mayavi/>.
- [27] H. J. D. «Matplotlib.» (2003), dirección: <https://matplotlib.org/>.
- [28] W. McKinney. «Pandas.» (2008), dirección: <https://pandas.pydata.org/>.
- [29] T. Nishita, T. Sirai, K. Tadamura y E. Nakamae, «Display of the earth taking into account atmospheric scattering,» New York, NY, USA: Association for Computing Machinery, 1993. dirección: <https://doi.org/10.1145/166117.166140>.
- [30] J. Taler y P. Ocloń, «Finite Element Method in Steady-State and Transient Heat Conduction,» en *Encyclopedia of Thermal Stresses*, R. B. Hetnarski, ed. Dordrecht: Springer Netherlands, 2014, ISBN: 978-94-007-2739-7. dirección: https://doi.org/10.1007/978-94-007-2739-7_897.
- [31] G. Gundersen. «Why Shouldn't I Invert That Matrix?» (2020), dirección: <https://gregorygundersen.com/blog/2020/12/09/matrix-inversion/>.
- [32] E. Bainville. «GPU matrix-vector product (gemv).» (2010), dirección: <https://www.bealto.com/gpu-gemv.html>.
- [33] National Science Foundation y Department of Energy. «(BLAS) Basic Linear Algebra Subprograms.» (1979), dirección: <https://www.netlib.org/blas>.

8. Anexos

8.1. Resultados Extendidos

En este anexo se incluyen casos adicionales a los expuestos en la sección de Resultados.

8.1.1. Conducción

8.1.1.1 Prueba 1: Conducción simple en cubo

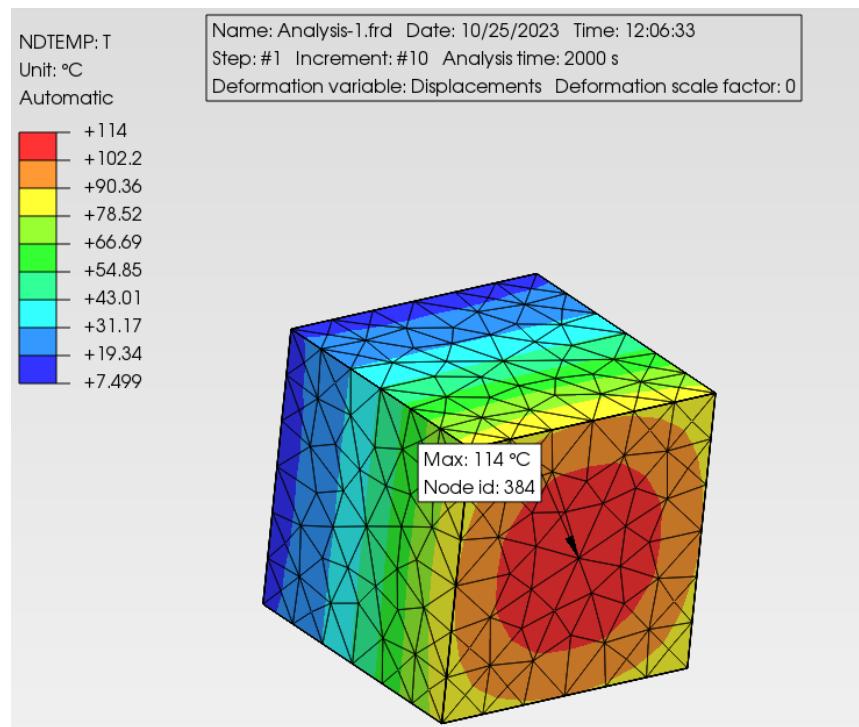


Figura 54: Vista volumétrica de temperatura a los 2000 segundos - PrePoMax

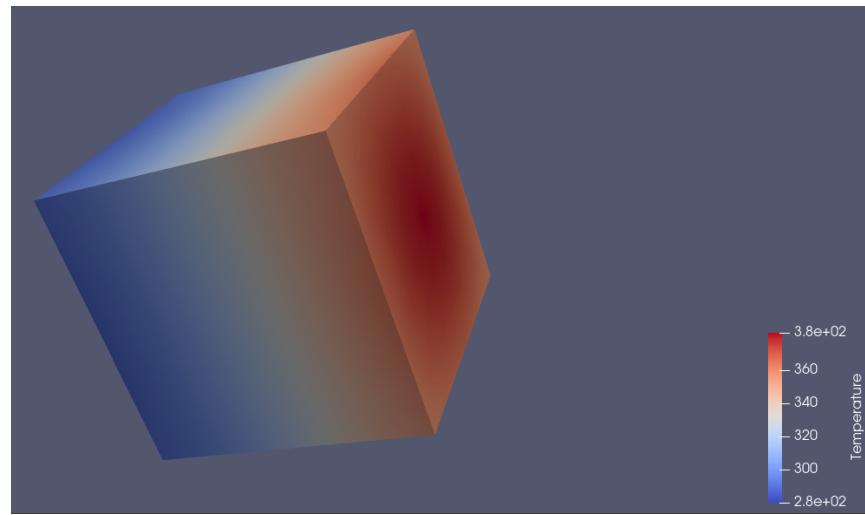


Figura 55: Vista volumétrica de temperaturas a los 2000 segundos - Agni B3V

Luego de 2000 segundos de simulación en PrePoMax (Fig. 54), las temperaturas varían entre 7,5 °C y 114 °C, mientras que en las simulaciones con Agni B3V (Fig. 55) las temperaturas varían entre 280 K (6,85 °C) y 382 K (108,85 °C).

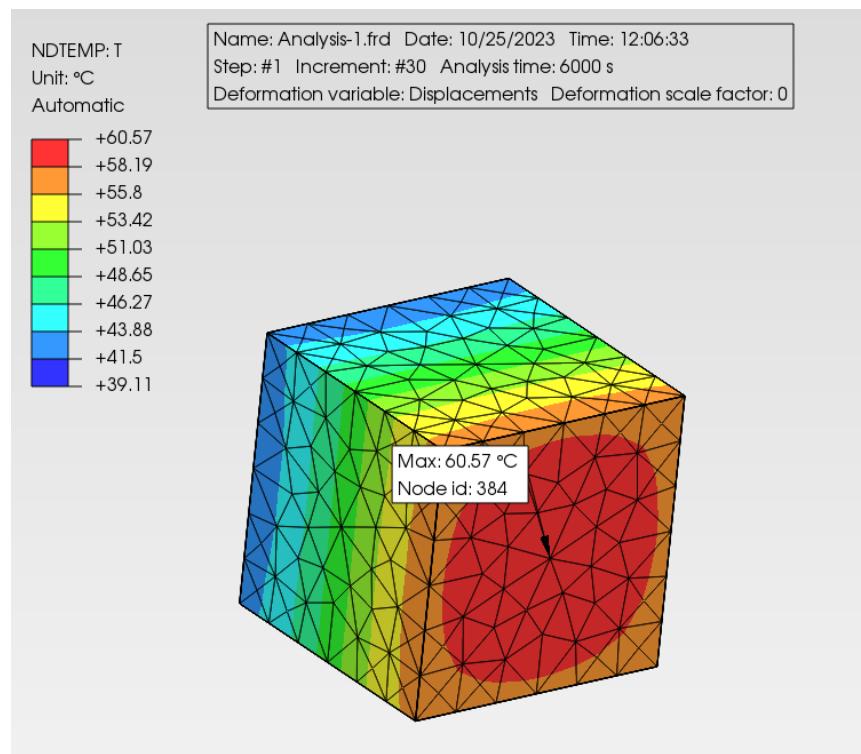


Figura 56: Vista volumétrica de temperatura a los 6000 segundos - PrePoMax

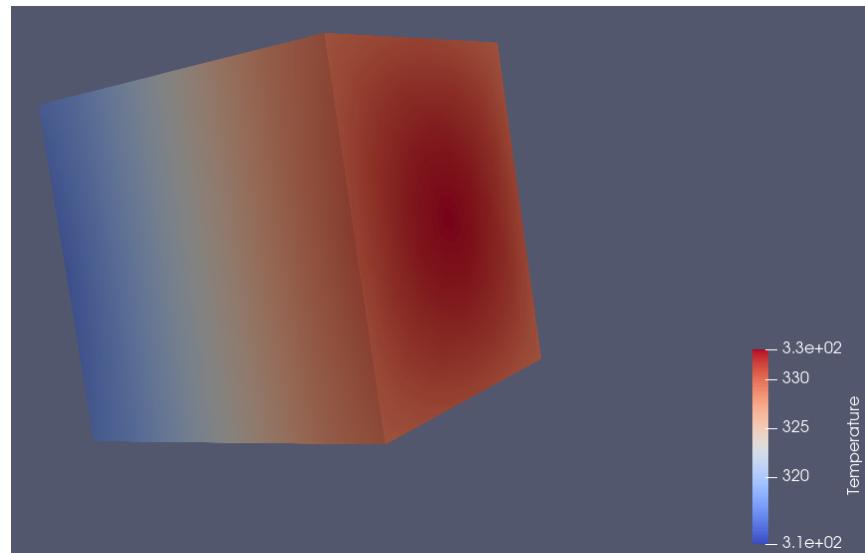


Figura 57: Vista volumétrica de temperaturas a los 6000 segundos - Agni B3V

Tras 6000 segundos de simulación en PrePoMax (Fig. 56), las temperaturas varían entre 39,11 °C y 60,57 °C. En el mismo instante de la simulación con Agni (Fig. 57) las temperaturas varían entre 314 K (40,85 °C) y 332 K (58,85 °C).

8.1.1.2 Prueba 3: Conducción simple en cubo de dos materiales

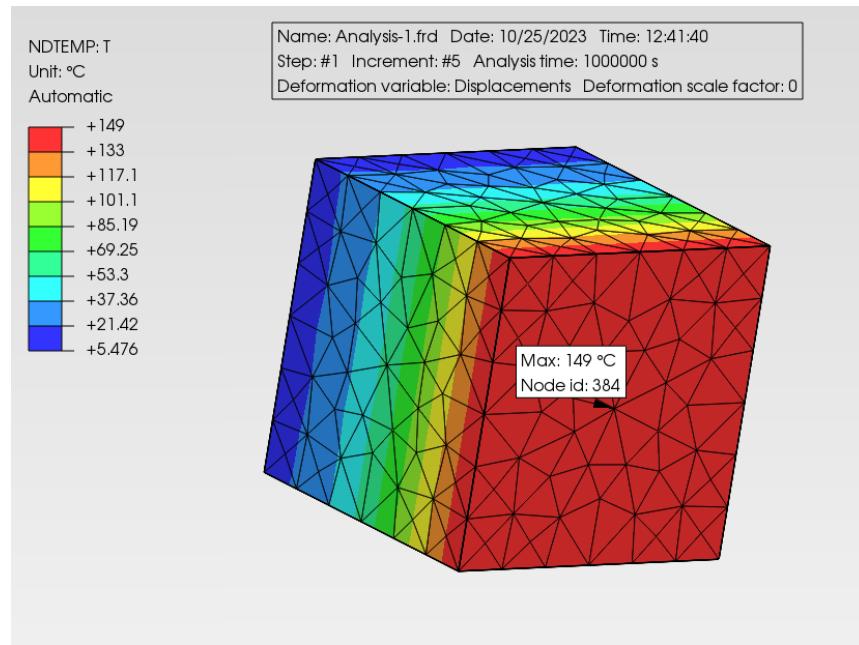


Figura 58: Vista volumétrica de temperatura a los 1×10^6 segundos - PrePoMax

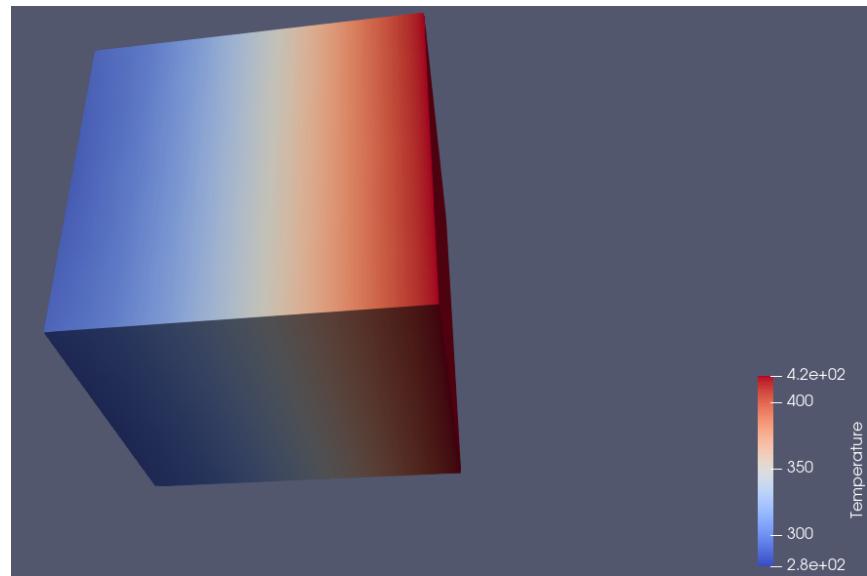


Figura 59: Vista volumétrica de temperaturas a los 1×10^6 segundos - Agni B3V

Luego de 1×10^6 segundos (11 días) de simulación en PrePoMax (Fig. 58) la temperatura se encuentra entre 5,4 °C y 149 °C. En la simulación con Agni B3V (Fig. 59) la temperatura está entre 277 K (3,85 °C) y 419 K (145,85 °C).

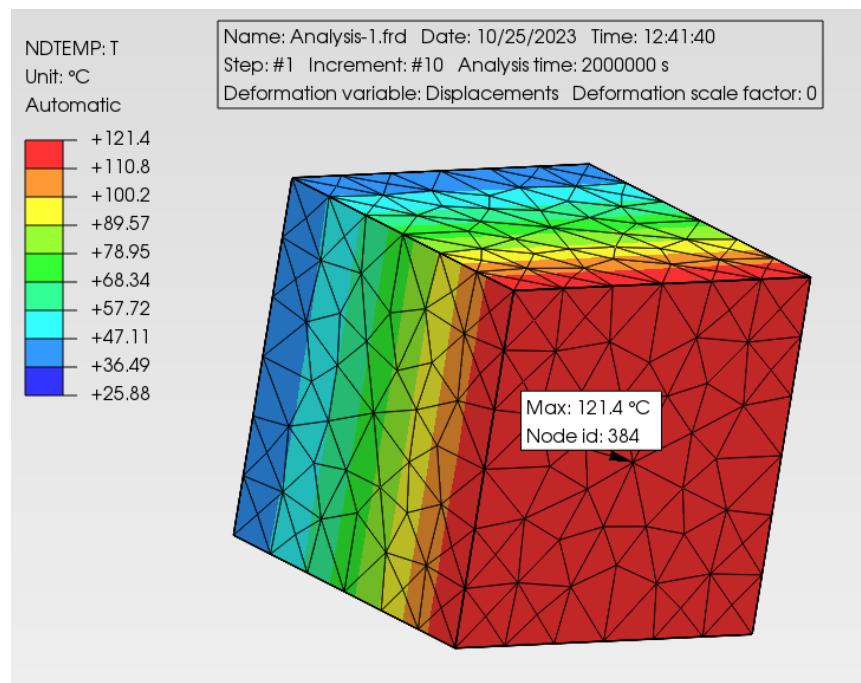


Figura 60: Vista volumétrica de temperatura a los 2×10^6 segundos - PrePoMax

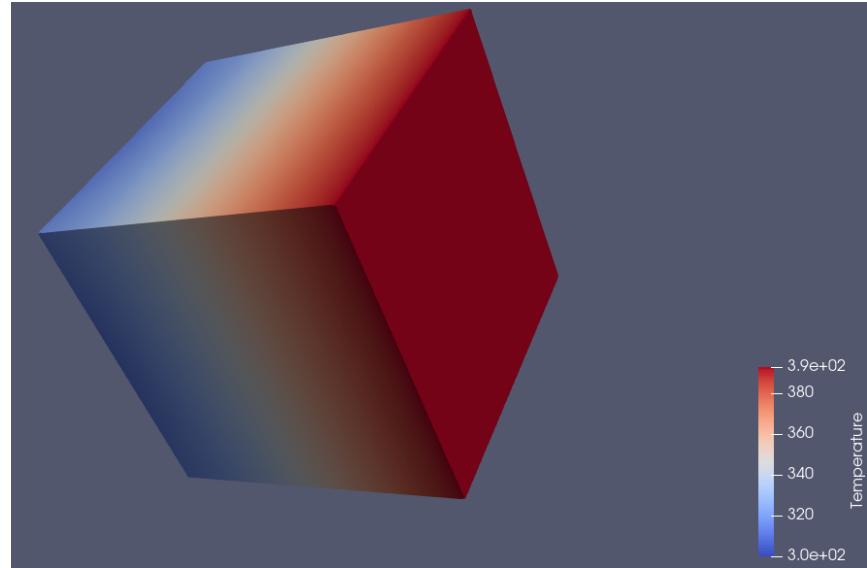


Figura 61: Vista volumétrica de temperatura a los 2×10^6 segundos - Agni B3V

Tras 2×10^6 segundos (23 días) de simulación en PrePoMax (Fig. 60) la temperatura se encuentra entre 25,88 °C y 121,4 °C. Pasados 2000000 segundos de la simulación en Agni (Fig. 61), la temperatura se encuentra entre 300 K (26,85 °C) y 392 K (118,85 °C).

8.1.1.3 Prueba 4: Conducción simple en cubo de aluminio y cobre

Se modeló un cubo con las siguientes características:

- Tamaño de 1 m × 1 m × 1 m
- Una de las caras tiene como material cobre, con:
 - Densidad de 8960 $\frac{\text{kg}}{\text{m}^3}$
 - Calor específico de 385 $\frac{\text{J}}{\text{kg K}}$
 - Conductividad térmica de 400 $\frac{\text{W}}{\text{K m}}$
- El resto de las caras tienen de material aluminio, con:
 - Densidad de 2700 $\frac{\text{kg}}{\text{m}^3}$
 - Calor específico de 900 $\frac{\text{J}}{\text{kg K}}$
 - Conductividad térmica de 237 $\frac{\text{W}}{\text{K m}}$
- La temperatura inicial de la cara de cobre es 300 °C
- La temperatura inicial de las caras de aluminio es 0 °C
- No hay flujos

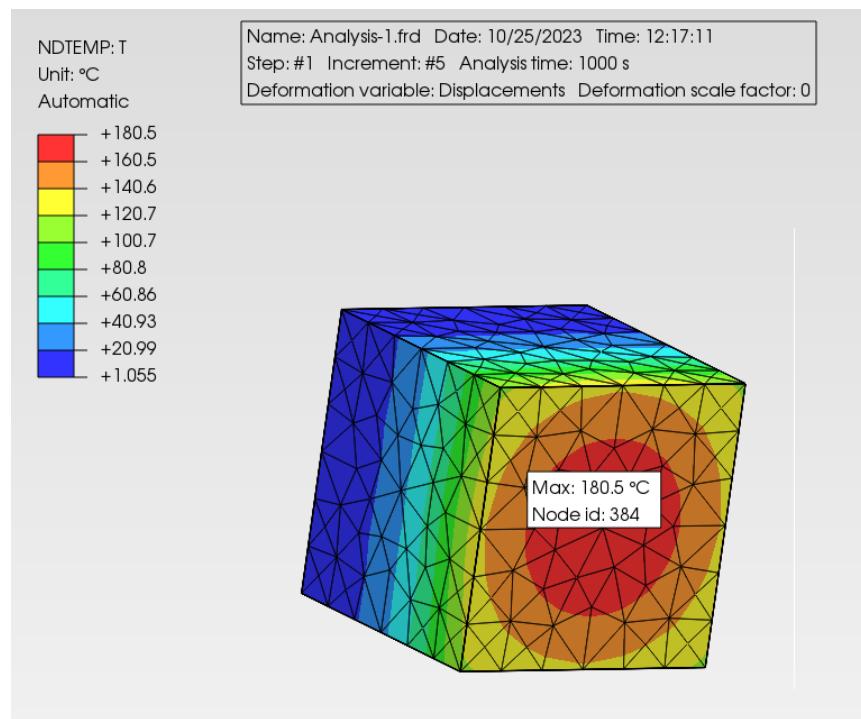


Figura 62: Vista volumétrica de temperatura a los 1000 segundos - PrePoMax

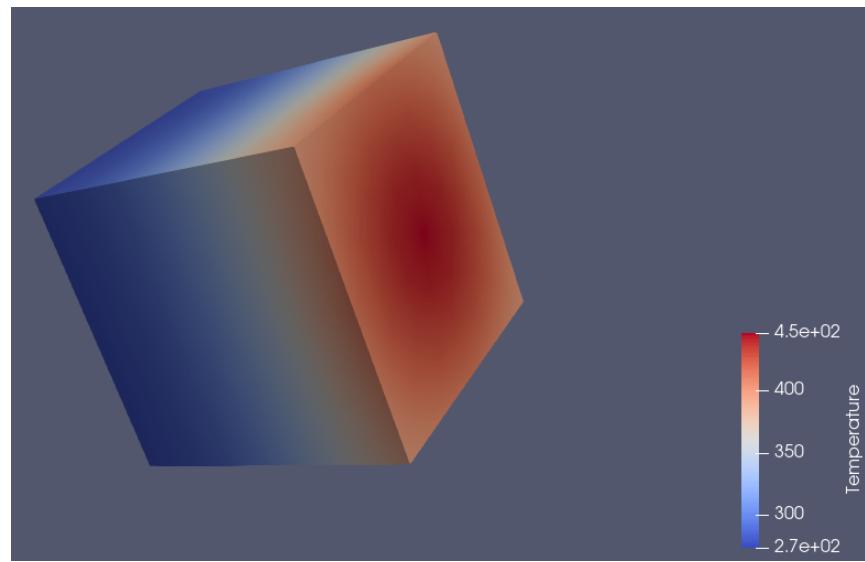


Figura 63: Vista volumétrica de temperatura a los 1000 segundos - Agni B3V

Luego de 1000 segundos de simulación, se llegó a una temperatura de entre 1,055 °C y 180,5 °C en PrePoMax(Fig. 62). Mientras que en la simulación con Agni B3V (Fig. 63), se llegó a una temperatura entre 274 K (0,85 °C) y 445 K (171,85 °C).

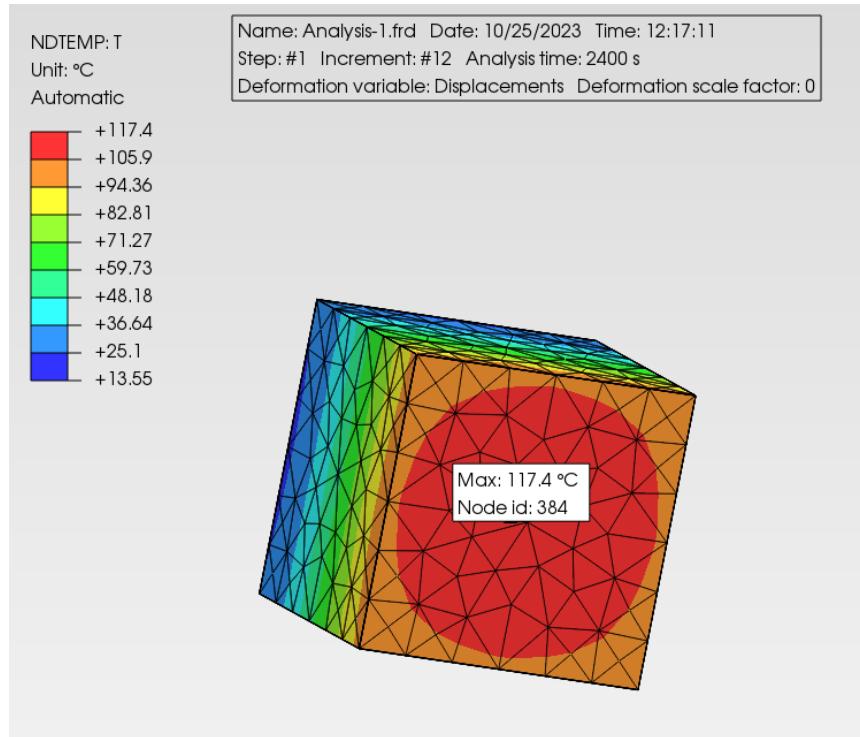


Figura 64: Vista volumétrica de temperatura a los 2400 segundos - PrePoMax

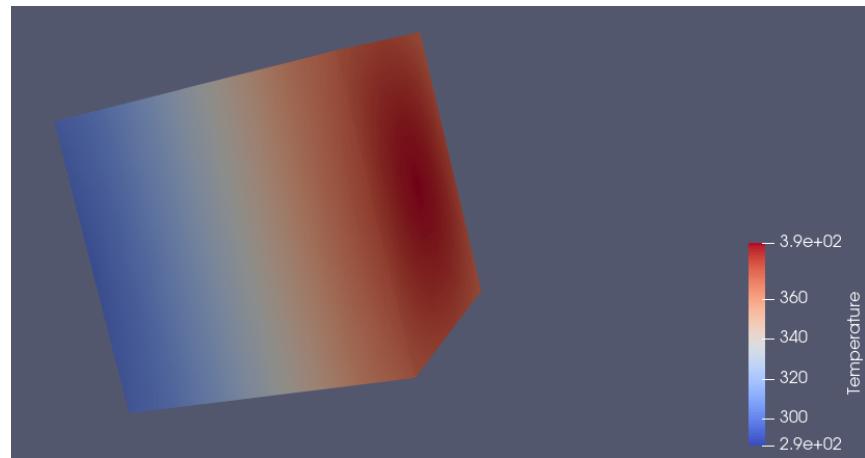


Figura 65: Vista volumétrica de temperatura a los 2400 segundos - Agni B3V

Tras 2400 segundos de simulación en PrePoMax (Fig. 64), se llegó a una temperatura entre 13,55 °C y 117,4 °C. En cambio, en la simulación con Agni B3V (Fig. 65), se llegó a una temperatura entre 287 K (13,85 °C) y 387 K (113,85 °C).

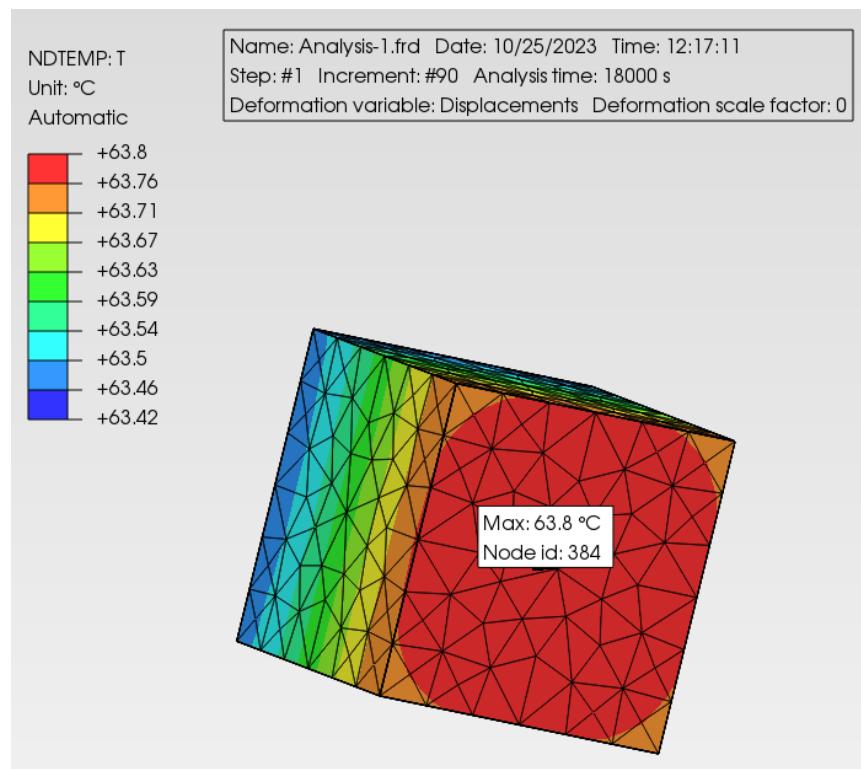


Figura 66: Vista volumétrica de temperatura a los 18000 segundos - PrePoMax

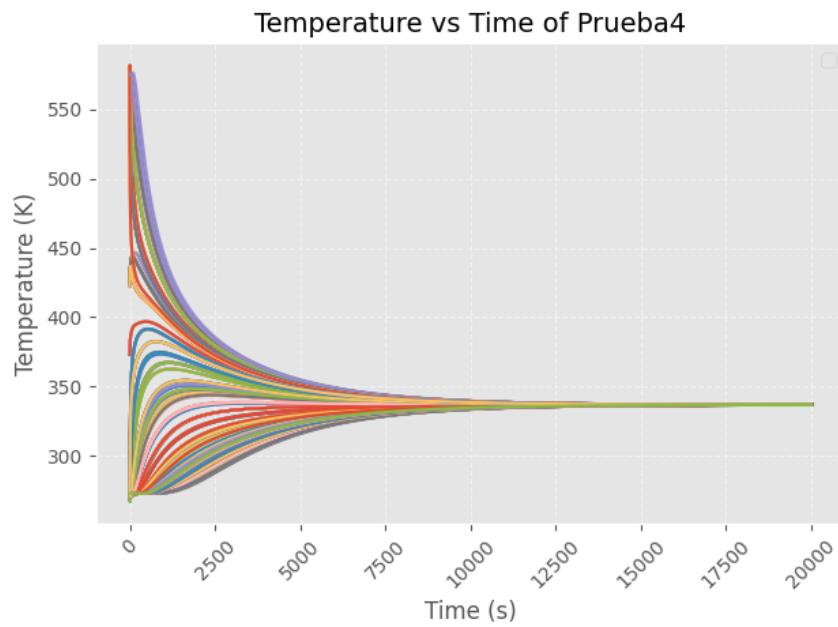


Figura 67: Evolución de la temepratura de los nodos en Agni B3V

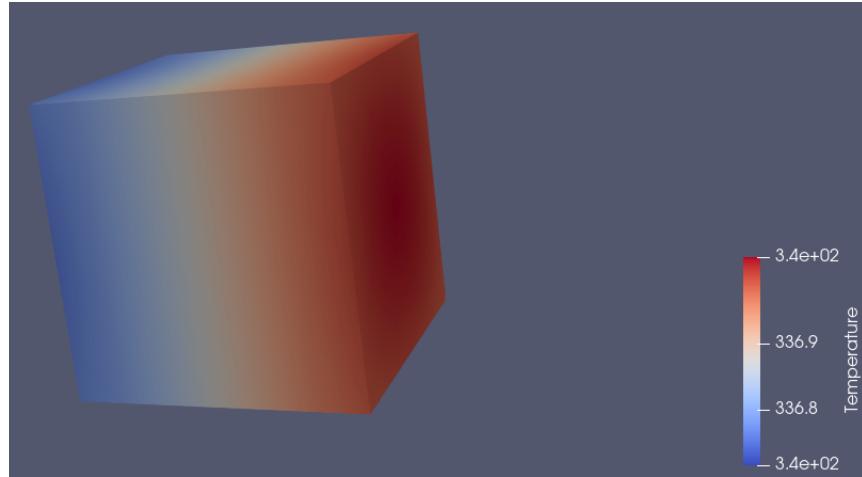


Figura 68: Vista volumétrica de temperatura a los 18000 segundos - Agni B3V

Luego de 18000 segundos de simulación en PrePoMax (Fig. 66), se llegó a una temperatura estable entre 63,42 °C y 63,8 °C. Se puede observar que a los 18000 de segundos de simulación en Agni B3V (Fig. 67) se estabilizó en la misma temperatura de 336,6 K (63,45 °C). Además, se valida que desde Paraview se interpreten los mismos resultados (Fig. 68).

En conclusión, los resultados son compatibles con lo que se espera de las condiciones planteadas. Como la temperatura inicial del cobre es mayor, se transmite la energía hacia el aluminio y el sistema se estabiliza. Adicionalmente, se comprueba que los resultados difieren del primer caso, donde las condiciones iniciales eran idénticas, a excepción de los materiales asignados.

8.1.1.4 Prueba 5: Conducción en cubo con flujo constante y dos materiales

Se modeló un cubo con las siguientes características:

- Tamaño de $1 \text{ m} \times 1 \text{ m} \times 1 \text{ m}$
- Una de las caras tiene como material cobre, con:
 - Densidad de $8960 \frac{\text{kg}}{\text{m}^3}$
 - Calor específico de $385 \frac{\text{J}}{\text{kg K}}$
 - Conductividad térmica de $400 \frac{\text{W}}{\text{K m}}$
- El resto de las caras tienen de material roble, con:
 - Densidad de $700 \frac{\text{kg}}{\text{m}^3}$
 - Calor específico de $2300 \frac{\text{J}}{\text{kg K}}$
 - Conductividad térmica de $0,23 \frac{\text{W}}{\text{K m}}$
- La temperatura inicial es de 0°C para todo el cubo.
- Flujo constante de $100 \frac{\text{W}}{\text{m}^2}$ para todo el cubo

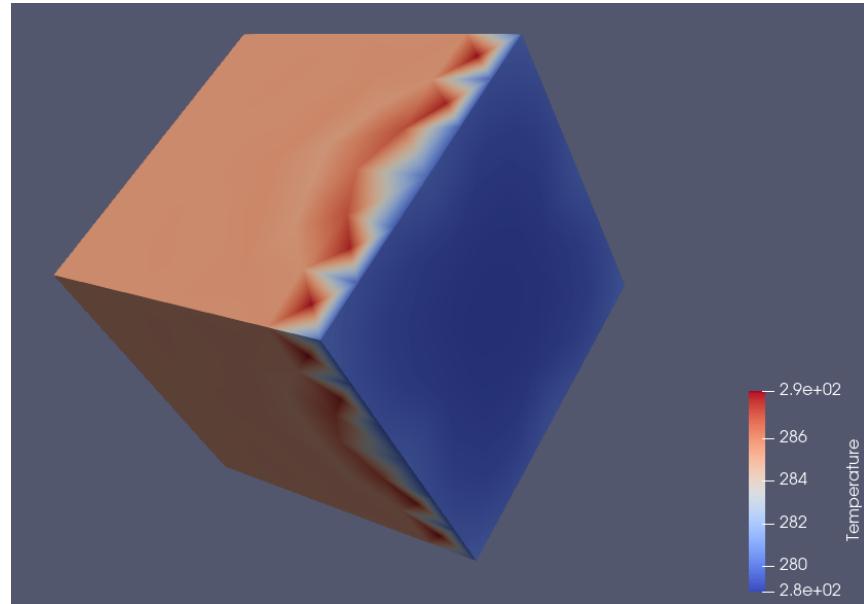


Figura 69: Vista volumétrica de temperatura a los 200 segundos - Agni B3V

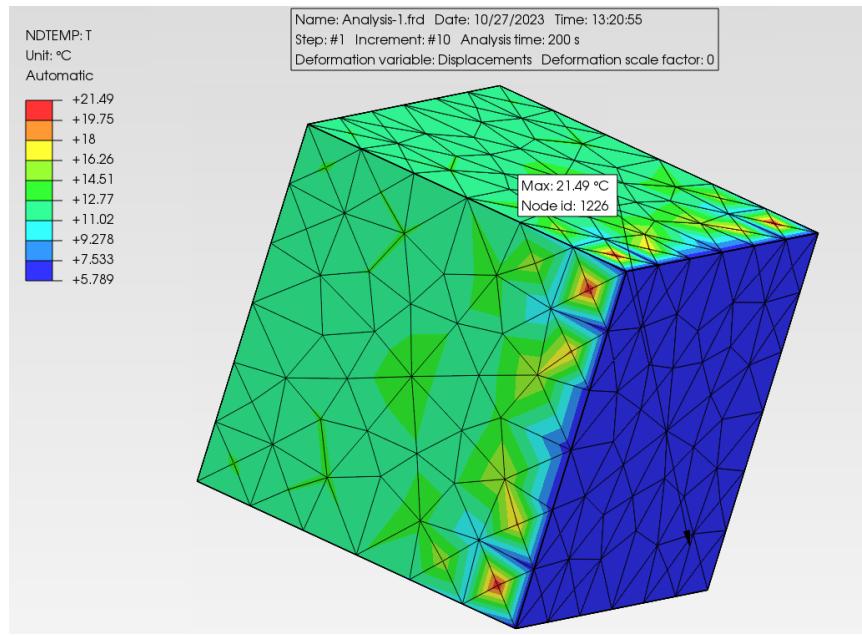


Figura 70: Vista volumétrica de temperatura a los 200 segundos - PrePoMax

Luego de 200 segundos de simulación en PrePoMax (Fig. 69), la cara de cobre tiene una temperatura de 5,78 °C, mientras que el roble tiene una temperatura de aproximadamente 11,89 °C. En la simulación con Agni B3V (Fig. 70), la cara de cobre tiene una temperatura de 279 K (5,85 °C), mientras que las de roble alrededor de 285 K (11,85 °C).

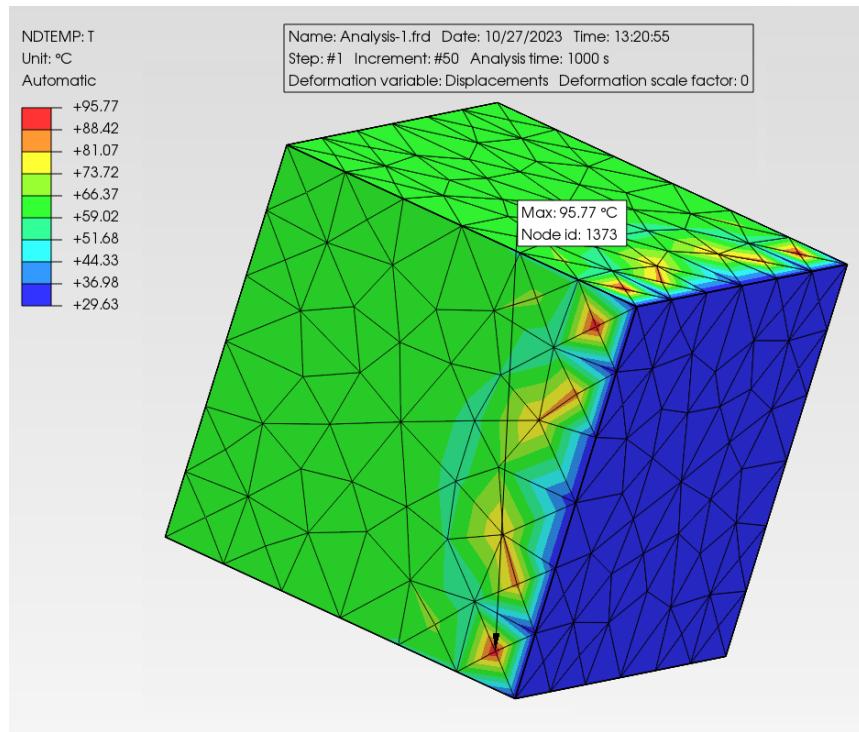


Figura 71: Vista volumétrica de temperatura a los 1000 segundos - PrePoMax

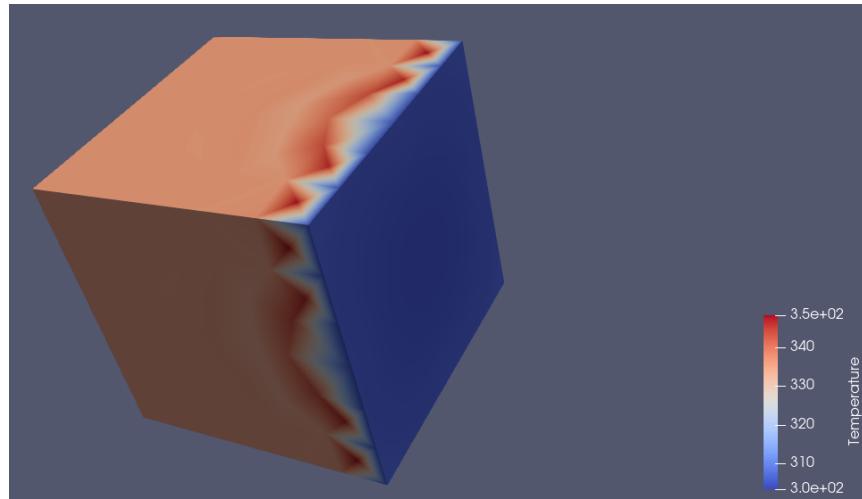


Figura 72: Vista volumétrica de temperatura a los 1000 segundos - Agni B3V

Tras 1000 segundos de simulación en PrePoMax (Fig. 71), la cara de cobre tiene una temperatura

de 29,63 °C, mientras que el roble tiene una temperatura de aproximadamente 62,69 °C. Pasado el mismo intervalo de tiempo en la simulación de Agni B3V (Fig. 72), la cara de cobre tiene una temperatura de 304 K (30,85 °C), mientras que las de roble de 335 K (61,85 °C).

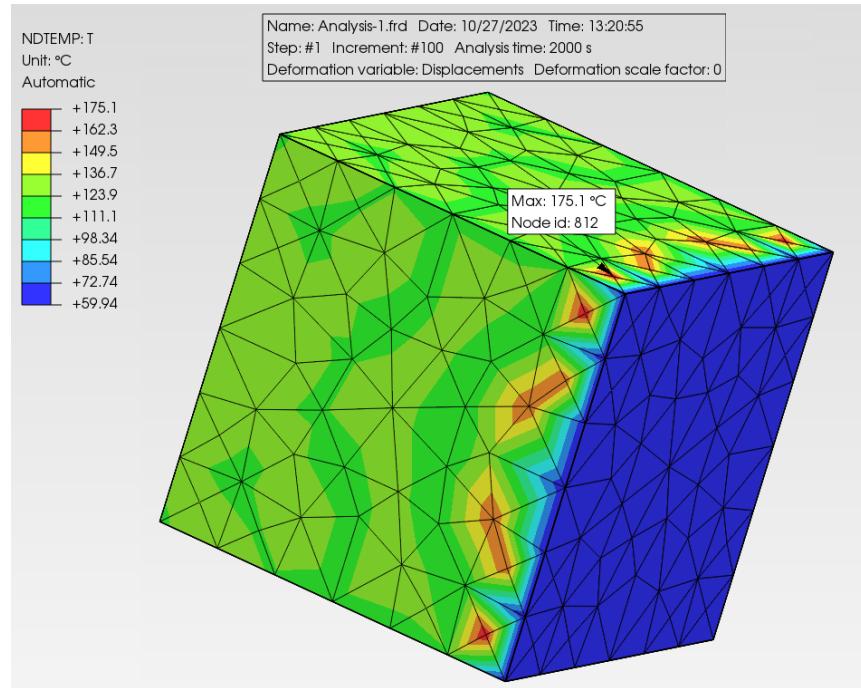


Figura 73: Vista volumétrica de temperatura a los 2000 segundos - PrePoMax

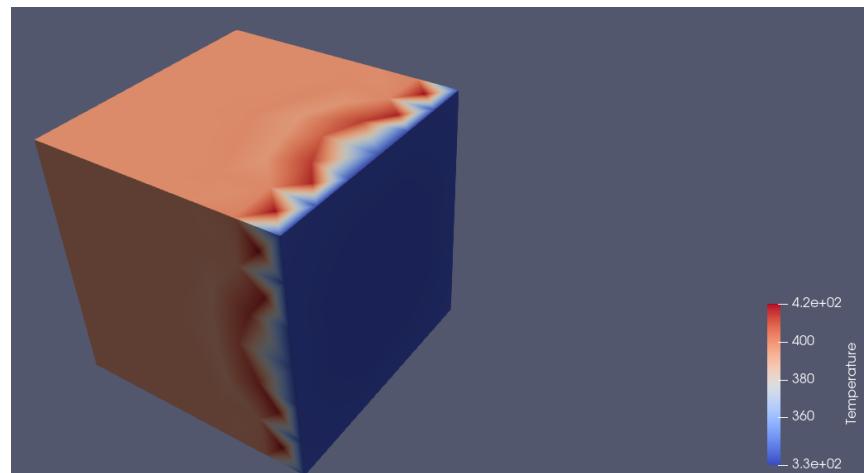


Figura 74: Vista volumétrica de temperatura a los 2000 segundos - Agni B3V

Habiendo transcurrido 2000 segundos de simulación en PrePoMax (Fig. 73), la cara de cobre tiene una temperatura de 59,94 °C, mientras que el roble tiene una temperatura de aproximadamente 123,9 °C. Luego de 2000 segundos de simulación en Agni B3V (Fig. 74), la cara de cobre tiene una temperatura de 335 K (61,85 °C), mientras que las de roble de 397 K (123,85 °C).

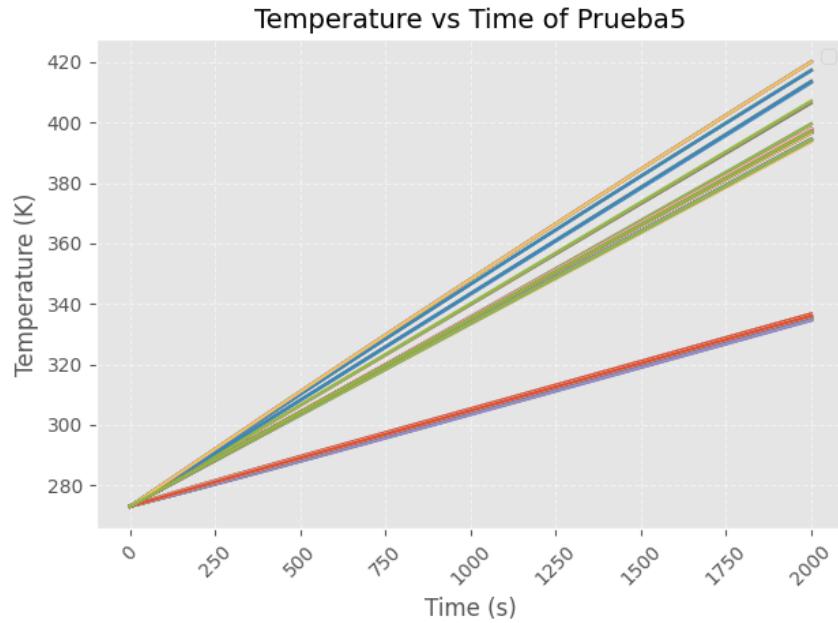


Figura 75: Evolución de temperatura de los nodos en Agni B3V

Este caso (Fig. 75) permite observar las diferencias en la conductividad de los materiales. Las líneas amarillas representan el roble, las rojas el cobre, y las demás la frontera entre ambas. Las caras de roble incrementan su temperatura más rápido que el cobre, pero aun así la zona lindera con las caras de cobre se enfrián progresivamente.

Se puede estimar analíticamente cuánto sería el cambio de temperatura de una de las caras del cubo a partir del flujo que recibe.

Para la cara de cobre, a los 200 segundos se tiene:

$$\begin{aligned}
 \Delta T &= \frac{q}{mc} \\
 &= \frac{Q A t}{V \rho c} \\
 &= \frac{100 \frac{\text{W}}{\text{m}^2} \times 1 \text{ m}^2 \times 200 \text{ s}}{0,001 \text{ m}^3 \times 8960 \frac{\text{kg}}{\text{m}^3} \times 385 \frac{\text{J}}{\text{kg K}}} \\
 &= 5,79 \text{ K}
 \end{aligned} \tag{66}$$

Que es efectivamente el cambio de temperatura a los 200 segundos.

Para 2000 segundos, el cambio de temperatura debería ser 57,9 K. Sin embargo, esta es 59,94 K, esto se explica porque, como también está el roble, cuya temperatura es mayor, hay conducción, por lo que aumenta la temperatura del cobre.

A pesar de haber empezado con la misma temperatura, el roble y el cobre tienen constantes térmicas diferentes, lo que implica distinto cambio de temperatura ante igual flujo térmico. En consecuencia, y teniendo en cuenta los resultados de la prueba 4, el tiempo necesario para que se estabilicen las temperaturas por conducción es mucho mayor, por lo que no se llega a apreciar en el intervalo establecido. El calor específico del roble es mayor al del cobre, pero su densidad es menor y, por tanto, también lo es su masa. Considerando ambos factores es razonable que la temperatura final de la cara de roble sea superior al de las de cobre.

8.1.2. Radiación

8.1.2.1 Prueba 2: Placas paralelas distinta emisividad

Se modelaron dos placas planas paralelas con las siguientes características:

- Tamaño de 1000 m × 1000 m
- Grosor de 0,001 m
- Distancia entre las placas de 0,1 m
- Las propiedades del material son:
 - Densidad de 2700 $\frac{\text{kg}}{\text{m}^3}$
 - Calor específico de 897 $\frac{\text{J}}{\text{kg K}}$
 - Conductividad térmica de 0 $\frac{\text{W}}{\text{K m}}$
- Absortancia de la placa 1 de 0,7
- Absortancia de la placa 2 de 1,0
- La temperatura inicial de la placa 1 es de 500 K
- La temperatura inicial de la placa 2 es de 300 K

Comenzando por la interpretación teórica, el flujo de calor para la placa 1 será:

$$q_{\text{Total}_1} = q_{\text{Gain}_1} - q_{\text{Lost}_1} \quad (67)$$

Teniendo en cuenta la pérdida y ganancia de calor

$$\begin{aligned} q_{\text{Gain}_1} &= F_{1,1} \alpha_1 \alpha_1 \sigma A_1 T_1^4 + F_{2,1} \alpha_1 \alpha_2 \sigma A_2 T_2^4 \\ q_{\text{Lost}_1} &= \alpha_1 \sigma A_1 T_1^4 \end{aligned} \tag{68}$$

Luego, la variación de temperatura es obtenida a partir de la siguiente ecuación:

$$\begin{aligned} \Delta T_1 &= \frac{q_{\text{Total}_1} \Delta t}{\rho V c} \\ &= \frac{\sigma A (F_{1,1} \alpha_1 \alpha_1 T_1^4 + F_{2,1} \alpha_2 \alpha_1 T_2^4 - \alpha_1 T_1^4) \Delta t}{\rho A G c} \\ &= \frac{\sigma (F_{1,1} \alpha_1 \alpha_1 T_1^4 + F_{2,1} \alpha_2 \alpha_1 T_2^4 - \alpha_1 T_1^4) \Delta t}{\rho G c} \end{aligned} \tag{69}$$

Debido a que en este caso se tiene valores de absorbtividad menores a uno, los rayos que inciden en los cuerpos se verán reflejados. Por esta razón resulta práctico hacer un cambio de variable en las ecuaciones antes presentadas, tal que:

$$\begin{aligned} \hat{Y}_{1,1} &= F_{1,1} \alpha_1 \\ \hat{Y}_{1,2} &= F_{1,2} \alpha_2 \end{aligned} \tag{70}$$

El operador $\hat{Y}_{i,j}$ compone el factor de vista y la absorbtividad del cuerpo sobre el cual inciden los rayos. Debido a que la geometría del problema es sencilla, es posible calcular este operador para ambas placas haciendo uso de las siguientes expresiones:

$$\begin{aligned} \hat{Y}_{1,1} &= \sum_{i=0}^{\infty} \alpha_1 (1 - \alpha_2)^{i+1} (1 - \alpha_1)^i \\ \hat{Y}_{1,2} &= \sum_{i=0}^{\infty} \alpha_2 (1 - \alpha_1)^i (1 - \alpha_2)^i \end{aligned} \tag{71}$$

De forma análoga se puede obtener una expresión para los factores de la segunda placa, llegando a los siguientes valores:

$$\begin{aligned}
Y_{1,1}^{\wedge} &\approx 0 \\
Y_{1,2}^{\wedge} &\approx 1 \\
Y_{2,2}^{\wedge} &\approx 0,3 \\
Y_{2,1}^{\wedge} &\approx 0,7
\end{aligned} \tag{72}$$

Teniendo en cuenta estos valores, la diferencia de temperatura para ambas placas tras 10 segundos de simulación será:

$$\begin{aligned}
\Delta T_1 &= \frac{\sigma (0,0 \times 0,7 \times T_1^4 + 0,7 \times 1 \times T_2^4 - 0,7 \times T_1^4) \Delta t}{\rho G c} \\
&= -8,85 \text{ K} \\
\Delta T_2 &= \frac{\sigma (0,3 \times 1 \times T_2^4 + 1,0 \times 0,7 \times T_1^4 - 1,0 \times T_2^4) \Delta t}{\rho G c} \\
&= 8,85 \text{ K}
\end{aligned} \tag{73}$$

Por lo que las temperaturas esperadas serían:

$$\begin{aligned}
T_1 &= 500 \text{ K} - 8,852 \text{ K} = 491,15 \text{ K} \\
T_2 &= 300 \text{ K} + 8,852 \text{ K} = 308,85 \text{ K}
\end{aligned} \tag{74}$$

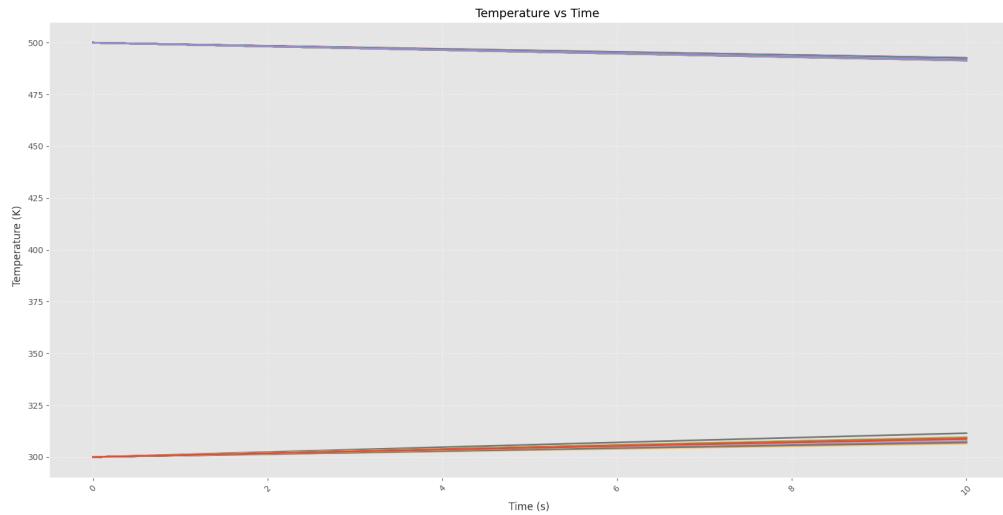


Figura 76: Evolución de temperatura de los nodos en Agni B3V

Luego de 10 segundos de simulación con el solver de Agni B3V (Fig. 76), el plano de temperatura 500 K pasa a una temperatura de 491,85 K, mientras que el plano de 300 K pasa a una temperatura de 308,15 K. Las temperaturas son similares, aunque existe un margen de error fácilmente atribuible a la aproximación por flujos constantes de calor o al hecho de que las placas no son realmente infinitas.

8.1.2.2 Prueba 3: Placas paralelas distinta emisividad

Se modelaron dos placas planas paralelas con las siguientes características:

- Tamaño de 1000 m × 1000 m
- Grosor de 0,001 m
- Distancia entre las placas de 0,1 m
- Las propiedades del material son:
 - Densidad de 2700 $\frac{\text{kg}}{\text{m}^3}$
 - Calor específico de 897 $\frac{\text{J}}{\text{kg K}}$
 - Conductividad térmica de 0 $\frac{\text{W}}{\text{K m}}$
- Absortancia de la placa 1 de 0,7
- Absortancia de la placa 2 de 0,2
- La temperatura inicial de la placa 1 es de 500 K
- La temperatura inicial de la placa 2 es de 300 K

Comenzando por la interpretación teórica, el flujo de calor para la placa 1 será:

$$q_{\text{Total}_1} = q_{\text{Gain}_1} - q_{\text{Lost}_1} \quad (75)$$

Teniendo en cuenta la pérdida y ganancia de calor

$$\begin{aligned} q_{\text{Gain}_1} &= F_{1,1} \alpha_1 \alpha_1 \sigma A_1 T_1^4 + F_{2,1} \alpha_1 \alpha_2 \sigma A_2 T_2^4 \\ q_{\text{Lost}_1} &= \alpha_1 \sigma A_1 T_1^4 \end{aligned} \quad (76)$$

Luego, la variación de temperatura es obtenida a partir de la siguiente ecuación:

$$\begin{aligned} \Delta T_1 &= \frac{q_{\text{Total}_1} \Delta t}{\rho V c} \\ &= \frac{\sigma A (F_{1,1} \alpha_1 \alpha_1 T_1^4 + F_{2,1} \alpha_2 \alpha_1 T_2^4 - \alpha_1 T_1^4) \Delta t}{\rho A G c} \\ &= \frac{\sigma (F_{1,1} \alpha_1 \alpha_1 T_1^4 + F_{2,1} \alpha_2 \alpha_1 T_2^4 - \alpha_1 T_1^4) \Delta t}{\rho G c} \end{aligned} \quad (77)$$

Se realiza un cambio de variables tal que:

$$\begin{aligned} \hat{Y}_{1,1} &= F_{1,1} \alpha_1 \\ \hat{Y}_{1,2} &= F_{1,2} \alpha_2 \end{aligned} \quad (78)$$

El operador $\hat{Y}_{i,j}$ puede ser calculado para ambas placas haciendo uso de las siguientes expresiones:

$$\begin{aligned} \hat{Y}_{1,1} &= \sum_{i=0}^{\infty} \alpha_1 (1 - \alpha_2)^{i+1} (1 - \alpha_1)^i \\ \hat{Y}_{1,2} &= \sum_{i=0}^{\infty} \alpha_2 (1 - \alpha_1)^i (1 - \alpha_2)^i \end{aligned} \quad (79)$$

De forma análoga se puede obtener una expresión para los factores de la segunda placa, llegando a los siguientes valores:

$$\begin{aligned}
\hat{Y_{1,1}} &\approx 0,737 \\
\hat{Y_{1,2}} &\approx 0,263 \\
\hat{Y_{2,2}} &\approx 0,079 \\
\hat{Y_{2,1}} &\approx 0,921
\end{aligned} \tag{80}$$

Teniendo en cuenta estos valores, la diferencia de temperatura para ambas placas tras 10 segundos de simulación será:

$$\begin{aligned}
\Delta T_1 &= \frac{\sigma (0,737 \times 0,7 \times T_1^4 + 0,921 \times 0,2 \times T_2^4 - 0,7 \times T_1^4) \Delta t}{\rho G c} \\
&= -2,33 \text{ K} \\
\Delta T_2 &= \frac{\sigma (0,079 \times 0,2 \times T_2^4 + 0,263 \times 0,7 \times T_1^4 - 0,2 \times T_2^4) \Delta t}{\rho G c} \\
&= 2,33 \text{ K}
\end{aligned} \tag{81}$$

Por lo que las nuevas temperaturas serían:

$$\begin{aligned}
T_1 &= 500 \text{ K} - 2,33 \text{ K} = 497,67 \text{ K} \\
T_2 &= 300 \text{ K} + 2,33 \text{ K} = 302,33 \text{ K}
\end{aligned} \tag{82}$$

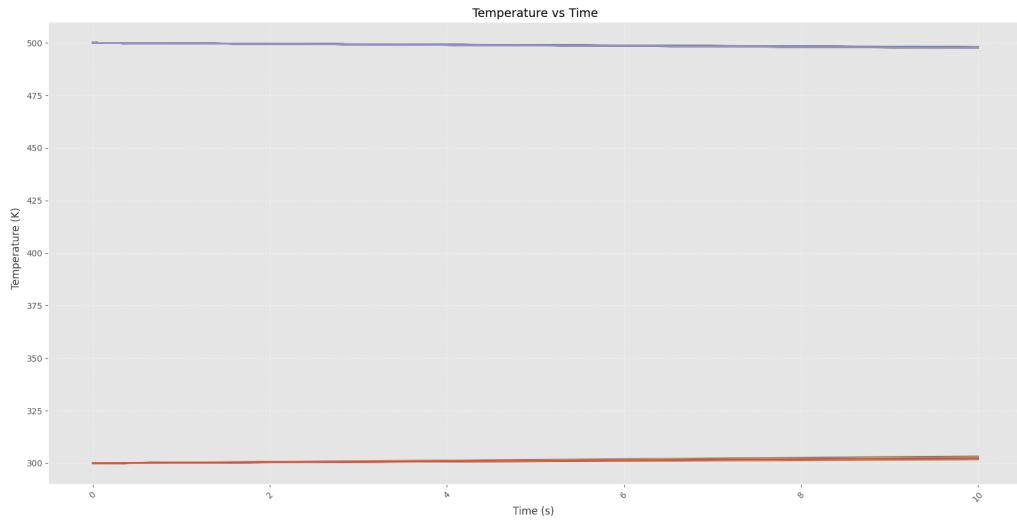


Figura 77: Evolución de la temperatura de los nodos en Agni B3V

Tras 10 segundos de simulación con el solver de Agni B3V (Fig. 77), el plano de temperatura 500 K pasa a una temperatura de 497,80 K, mientras que el plano de 300 K pasa a una temperatura de 302,20 K. Las temperaturas son similares, aunque existe un margen de error fácilmente atribuible a la aproximación por flujos constantes de calor o al hecho de que las placas no son realmente infinitas.

8.1.3. Convergencia

Al punto de comparación antes estudiado se le añaden dos secciones del modelo que se denominarán ‘Pirámide oculta’ (Fig. 78) y ‘Cara interior’ (Fig. 79).

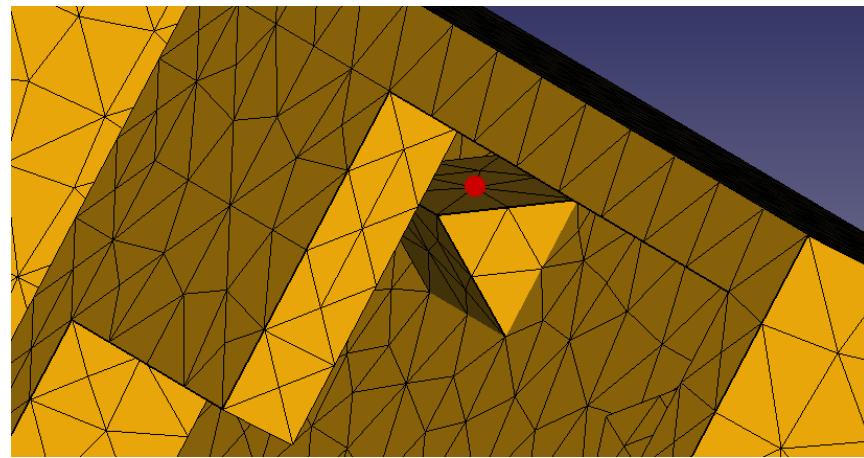


Figura 78: Vista de modelo - Pirámide Oculta

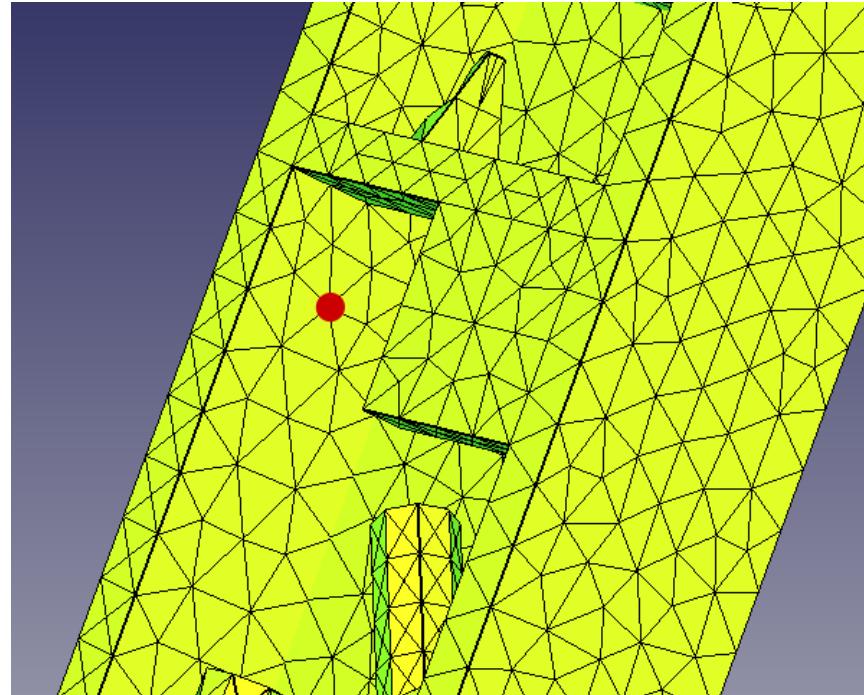


Figura 79: Vista de modelo - Cara Interior

8.1.3.1 Convergencia en Tiempo

No se aprecian diferencias en el ritmo o valor de convergencia en los pasos de tiempo estudiados para ambos casos (Fig. 80) (Fig. 81).

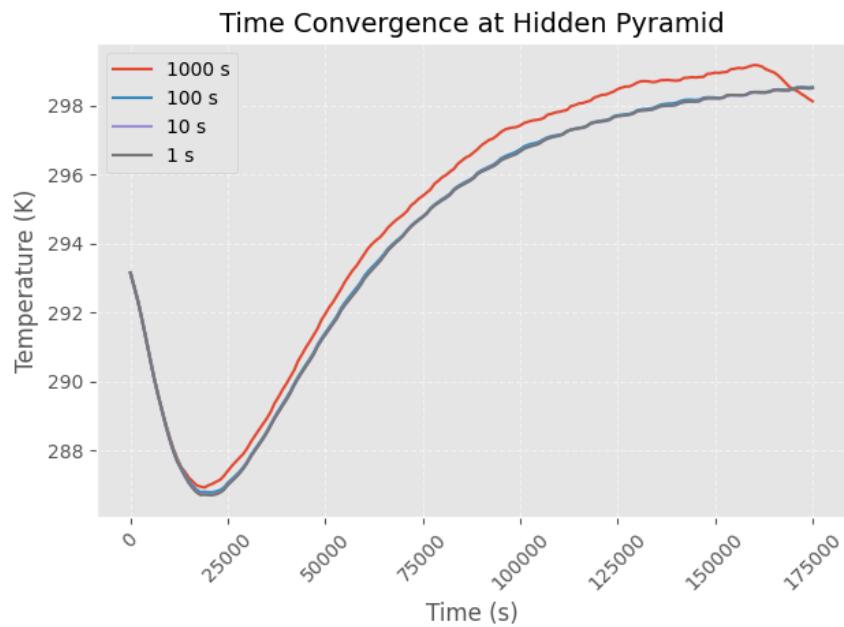


Figura 80: Convergencia en tiempo - Pirámide Oculta

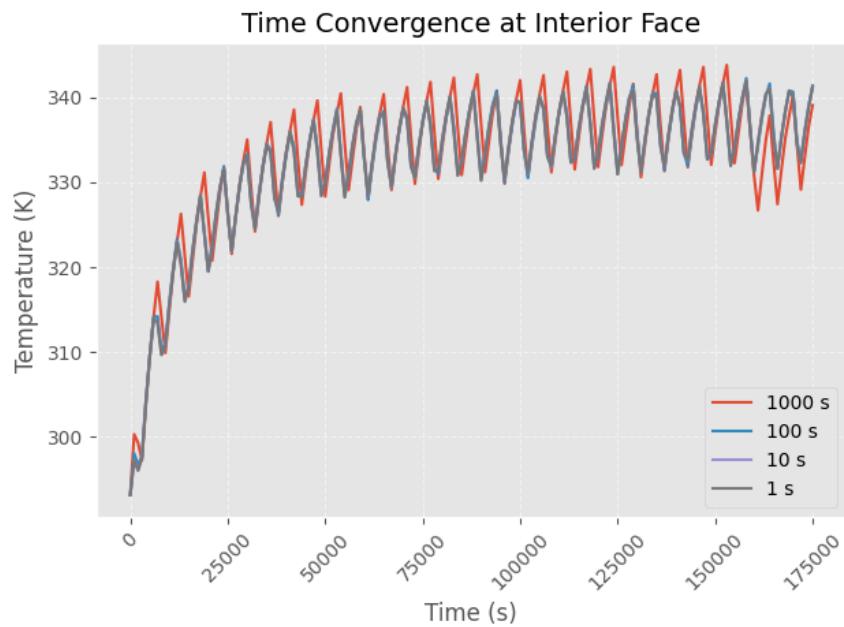


Figura 81: Convergencia en tiempo - Cara Interior

Tomando como referencia el caso con paso de un segundo se calculó el error relativo (Fig. 82) (Fig. 83) y se comprueba que en la cara interior el error relativo solo aumenta al 3 % al incrementar el paso de tiempo a 1000 segundos.

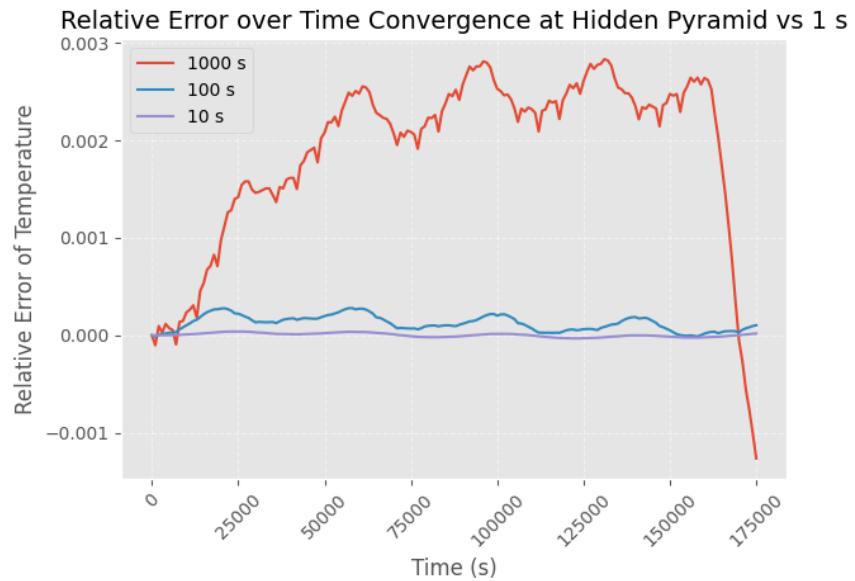


Figura 82: Error relativo - Piramide Oculta

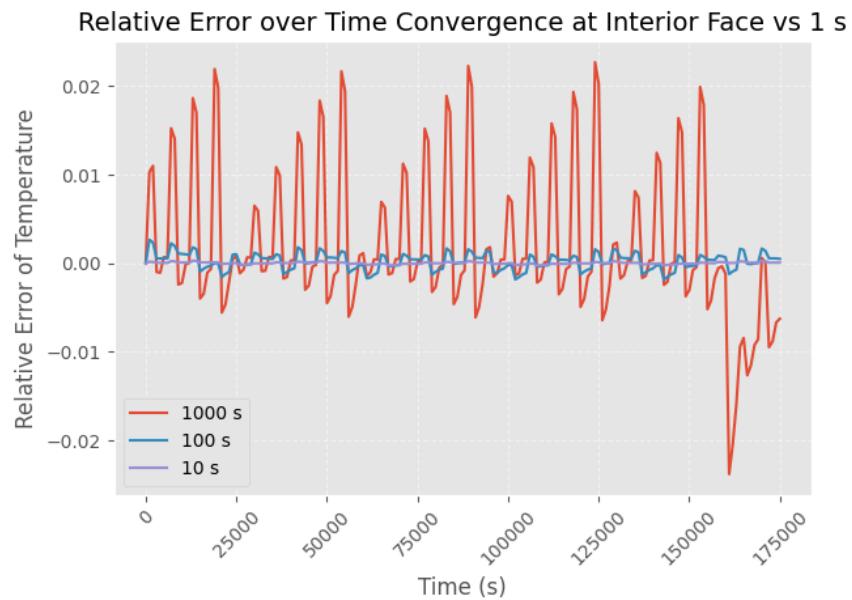


Figura 83: Error relativo - Cara Interior

8.1.3.2 Convergencia en Malla

En la convergencia en malla las diferencias en resolución se notan a simple vista (Fig. 84) (Fig. 85), pero no superan unos pocos kelvins.

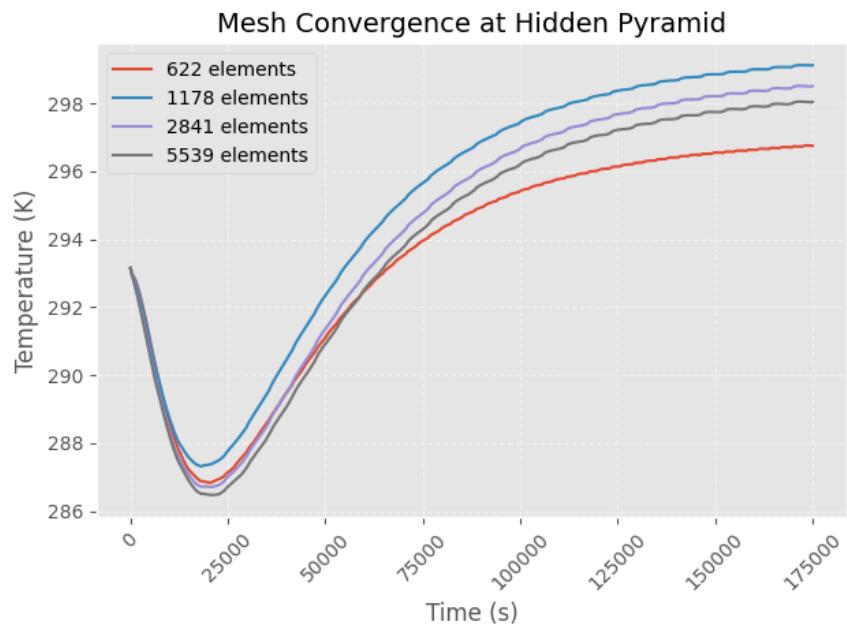


Figura 84: Convergencia en malla - Pirámide Oculta

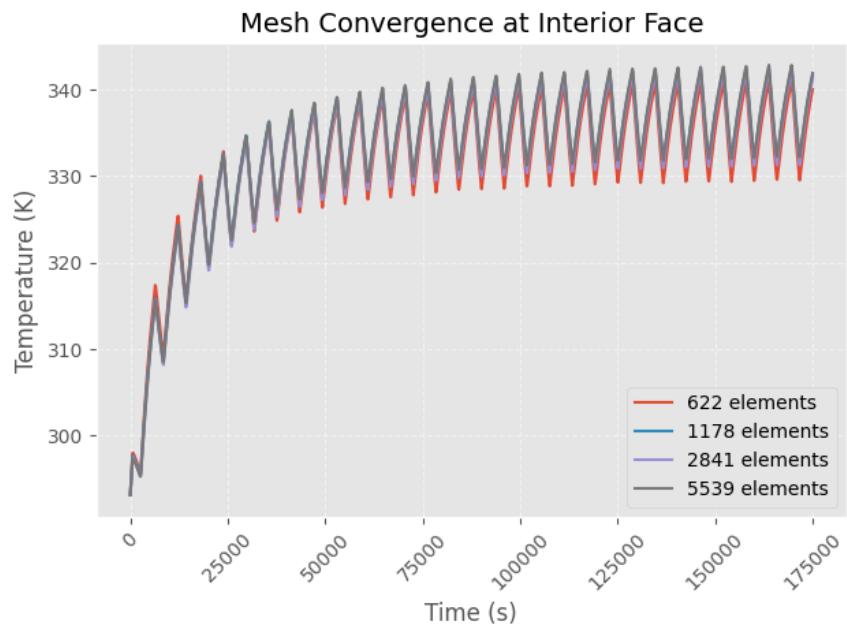


Figura 85: Convergencia en malla - Cara Interior

Se tomó como base el caso con un mallado de 5539 elementos y se calculó el error relativo (Fig. 86) (Fig. 87). Se comprueba que en relación al valor de las temperaturas trabajadas las diferencias en temperatura no son significativas.

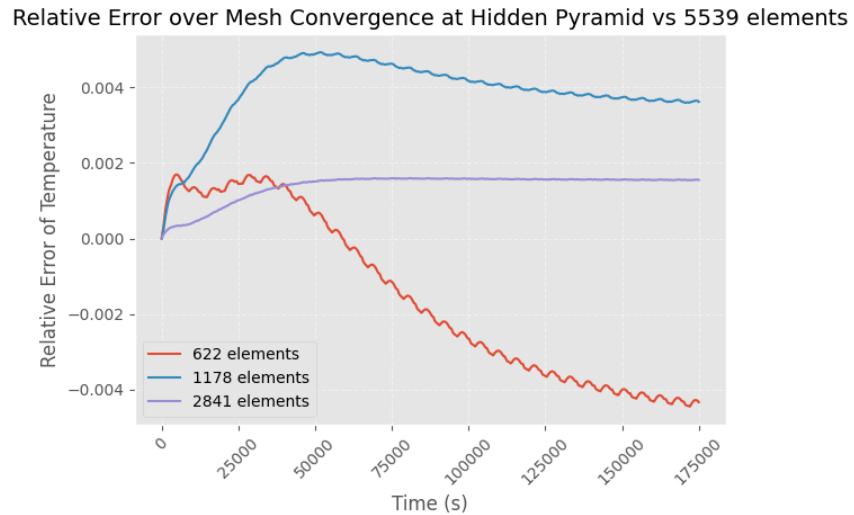


Figura 86: Error relativo - Pirámide Oculta

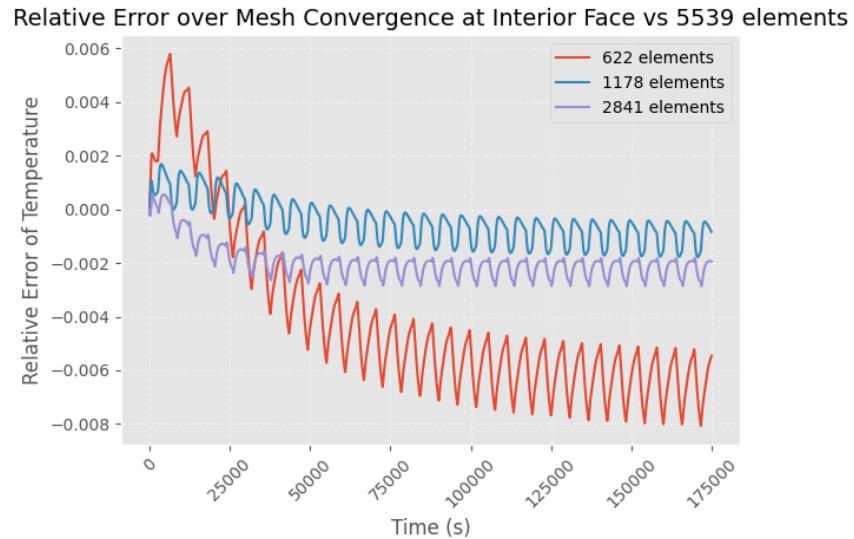


Figura 87: Error relativo - Cara Interior