UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO

FACULTAD DE INGENIERÍA

MODELADO Y SIMULACIÓN TÉRMICA DEL DETECTOR DE MATERIA OSCURA DAMIC 100

TESIS

Que para obtener el título de

Ingeniero Mecánico

PRESENTA

CÉSAR SANTIBÁÑEZ MARTÍNEZ

DIRECTOR DE TESIS

DR. FREDERIC TRILLAUD

Ciudad Universitaria, Cd. Mx., 2016

AGRADECIMIENTOS

A mis padres, Olga Leticia y Miguel, por todo el apoyo incondicional que me han otorgado, éste logro es suyo también. A mi hermano Miguel por ayudarme a tomar las mejores decisiones durante toda la carrera, gracias a ti llegué al proyecto DAMIC 100. A Brenda, Mariana y todos mis amigos que me apoyaron para poder terminar esta tesis.

Especiales agradecimientos al Dr. Frederic Trillaud, por ser tan paciente conmigo y darme la oportunidad de trabajar a su lado. Aprendí muchas cosas con usted, sobre todo lecciones de vida.

A la UNAM, a la Facultad de Ingeniería y al Instituto de Ingeniería por ser mi casa durante estos 5 años, ha sido definitivamente la mejor experiencia de mi vida.

También me gustaría agradecer al Dr. Juan Carlos D'Olivo Saez y al Dr. Alexis Armando Aguilar Arévalo del Departamento de Física de Altas Energías del Instituto de Ciencias Nucleares de la UNAM y a toda la Colaboración Internacional DAMIC.

ÍNDICE GENERAL

ÍNDICE DE FIGURAS Y TABLAS 5

1. Introducción 6
- **1.1 Planteamiento del problema** 8
- **1.2 Objetivo** 8
- **1.3 Justificación** 9

2. Detectores de partículas 11
- **2.1 Curvas de exclusión para detectores de partículas.** 11
- **2.2 Principales detectores de partículas** 12
- **2.3 Modulación anual** 13

3. Salome-Meca 15
- **3.1 Módulo de Geometría (*Geometry Module*)** 16
- **3.2 Módulo de Malla (*Mesh Module*)** 19
- **3.3 Módulo Aster** 21
- **3.4 Módulo ParaViS** 23
- **3.5 Código Python de DAMIC 100: Interfaz textual** 23

4. Diseño mecánico de DAMIC 100 25
- **4.1 Diseño mecánico original** 25
- **4.2 Simplificaciones del diseño original** 27

5. Análisis térmico 30
- **5.1 Modelado matemático de la ecuación de calor** 30
  - 5.1.1 Conducción térmica 30
  - 5.1.2 Radiación térmica 31
  - 5.1.3 Ecuación de calor para DAMIC 100 31
  - 5.1.4 Formulación débil de la ecuación de calor 31
  - 5.1.5 Algoritmo de Newton-Raphson 33
- **5.2 Código Code_Aster para la simulación térmica** 34

6. Resultados 39
- **6.1 Convergencia numérica** 39
- **6.2 Optimización de la temperatura** 40
- **6.3 Dependencia de las propiedades mecánicas de los materiales en el resultado final** 41

7. Conclusiones 44

APÉNDICE A: Código Python principal 46

Bibliografía 50

ÍNDICE DE FIGURAS Y TABLAS

Figura 1.1: Principio de detección de una partícula WIMP. 9
Figura 1.2: Trazas registradas por distintos tipos de partículas. 10
Figura 2.1: Experimentos recientes de detección de materia oscura. 12
Figura 2.2: Flujo de partículas WIMP de acuerdo a la posición de la Tierra. 14
Figura 3.1: Módulos de Salome utilizados en DAMIC 100. 15
Figura 3.2: Geometría del detector DAMIC 100 a estudiar. 16
Figura 3.3: Visualización de un arreglo CCD en Salome. 19
Figura 3.4: Visualización de la Figura 3.2 mallada con el algoritmo NETGEN 1D-2D-3D. 20
Figura 3.5: Figura 3.3 mallada con el algoritmo NETGEN 1D-2D-3D 21
Figura 3.6: Principales herramientas del Módulo Aster. 21
Figura 3.7: Pantalla principal de Eficas. 22
Figura 4.1: Bloque de plomo que cubre al contenedor de vacío. 25
Figura 4.2: Dedo frío y caja de CCDs que se encuentran dentro del bloque de Plomo. 25
Figura 4.3: 18 sensores CCD con sus respectivos cables cubiertos de Kapton. 26
Figura 4.4: Dedo frío de cobre. 27
Figura 4.5: Caja de cobre que cubre a los 18 CCDs. 28
Figura 4.6: Base de cobre del arreglo CCD. 28
Figura 4.7: Cable de Kapton. 28
Figura 4.8: Placa que cubre a la base de Silicio y barra que sujeta al cable de Kapton con la base del arreglo. 29
Figura 6.1: Resultados de la primer simulación térmica. 39
Figura 6.2: Resultado final en el arreglo CCD 1. 40
Figura 6.3: Resultado final en el arreglo CCD 18. 41
Figura 6.4: Gráfica de los errores de temperatura contra la variación de conductividad térmica. 42
Figura 6.5: Gráfica de los errores de temperatura contra los valores de emisividad. 43

Tabla 1: Errores de temperatura obtenidos por la variación de la conductividad térmica. 42
Tabla 2: Errores de temperatura obtenidos por la variación de emisividad térmica. 42

1. Introducción

El estudio de las partículas elementales para conocer sus propiedades, así como el intento por descubrir nuevas partículas, ha llevado a la física moderna a desarrollar nuevas tecnologías y dispositivos que les permitan cumplir dicho objetivo. Todo este equipamiento debe trabajar bajo ciertas condiciones de operación para alcanzar la sensibilidad requerida.

A partir del estudio de cuerpos celestes, los científicos se dieron cuenta de que existía una fuerza gravitacional que influía en el movimiento de dichos cuerpos. Al estudiar galaxias espirales, los científicos esperaban observar que los cuerpos de la periferia de la galaxia, como nubes de gases, se movían con mayor lentitud que los cuerpos del centro. Para su sorpresa, los cuerpos se movían a la misma velocidad, lo que sugirió que existía materia que no se puede ver pero que existe y que tiene efectos gravitacionales sobre la materia bariónica, a la que llamaron "Materia Oscura" [24].

La detección de la materia oscura es difícil pero no imposible. Actualmente ya se está trabajando en experimentos que puedan ayudar a detectarla. Sin duda los avances tecnológicos permitirán desarrollar las herramientas o equipamiento necesario para recabar más información sobre la materia oscura en un futuro muy cercano.

De los experimentos que existen actualmente para estudiar la materia oscura, el más económico y que aún se encuentra en desarrollo, es DAMIC (*Dark Matter in CCDs,* en inglés), que intenta detectar materia oscura con el uso de Dispositivos de Carga Acoplada (*Charge Coupled Devices,* en inglés) de calidad científica, hechos principalmente de Silicio puro. Las primeras pruebas de DAMIC se realizaron en el 2010 en Fermilab y se observó que esa tecnología tenía mucho potencial, por lo que se empezaron a desarrollar nuevas tecnologías con la ayuda de universidades e instituciones de diferentes países del mundo [12].

Actualmente se trabaja en DAMIC 100, donde se espera ocupar 18 sensores CCD de 16 MPixel, de 675 μm de espesor y de una masa de 5.2 g cada uno, lo que hace ~100 g de masa de Silicio puro, cuando anteriormente se manejaba una masa de ~10 g. Los sensores CCD que se utilizan en DAMIC son diseñados y fabricados en el laboratorio LBNL (*Lawrence Berkeley National Laboratory*), California, Estados Unidos [10].

La mayoría de la infraestructura de DAMIC se encuentra ubicado actualmente en el laboratorio SNOLAB, Sudbury, Ontario, Canadá. DAMIC es una colaboración internacional en el que participan diferentes universidades e institutos como: *Fermi National Accelerator Laboratory* (Estados Unidos), *Kavli Institute for Cosmological Physics* and *The Enrico Fermi Institute* (Universidad de Chicago, Estados Unidos), *University of Michigan* (Estados Unidos), Centro Atómico

Bariloche (Argentina), *University of Zurich* (Suiza), Universidad Nacional de Asunción (Paraguay) y la Universidad Nacional Autónoma de México (México) [1].

Los experimentos en busca de materia oscura se desarrollan en laboratorios subterráneos profundos, con el fin de evitar interacción con rayos cósmicos que se encuentran de manera abundante en la superficie de la Tierra. El laboratorio SNOLAB se encuentra a una profundidad de ~2 km [25].

Para poder realizar los experimentos, los científicos físicos requieren de los conocimientos y del uso de herramientas especializadas en la ingeniería. Una herramienta importante en la Ingeniería Mecánica es el uso de software CAD/CAE y herramientas especializadas de análisis para poder realizar el modelado del objeto en estudio, además de la simulación de las condiciones bajo las que se va a encontrar el objeto. Todo esto es de vital importancia para poder visualizar los efectos, internos y externos, que sufrirá nuestro objeto de estudio. De tal manera que si no se obtienen los resultados deseados, se busque una solución que satisfaga las necesidades del estudio, ya sea modificando el diseño original o modificando las condiciones de operación del dispositivo.

Si se logra entender qué es la materia oscura, podremos entender el tamaño, la forma y el futuro del universo. Se podrá comprender si el universo se seguirá expandiendo, se expandirá hasta un punto y luego colapsará, o se expandirá hasta lograr un equilibrio y posteriormente dejará de expandirse. Además, podremos entender la forma en la que las galaxias se forman y evolucionan. Aún nos falta mucho camino por recorrer y muchas cosas por aprender del universo en el que vivimos.

1.1 Planteamiento del problema

El uso de sensores CCD requiere de ciertas condiciones de trabajo para su óptimo funcionamiento, logrando disminuir el ruido provocado por el exterior así como por diferentes dispositivos que estén en contacto con el CCD. En DAMIC no se puede saber la temperatura exacta a la que trabajan los CCD, ya que si se introducen sensores que midan dicha temperatura, al mismo tiempo estarían provocando ruido que afecta a las mediciones. Ese es el motivo por el que se tiene que simular térmicamente todo el sistema para poder obtener un rango de temperatura aproximado al que se pudieran encontrar los CCD. En el último experimento realizado de DAMIC en el 2014, se trabajó a una temperatura en un rango de 130 K – 150 K, de tal manera que se logró reducir drásticamente el ruido generado en el CCD a $2e^-$ RMS [10].

Los electrones generados por el calor producido por el sistema (ruido térmico) no son deseables al momento de realizar mediciones, ya que reducen la calidad de la imagen. El ruido térmico existe aún cuando la luz no está en contacto con la superficie del detector y es conocido como “Corriente Oscura”. Con el fin de reducir en lo mayor posible el ruido generado por la corriente oscura, se debe enfriar el sensor a temperaturas criogénicas [23].

Ahora con DAMIC 100 al aumentar la cantidad a usar de sensores CCD, se busca tener un mayor número de eventos con mayor sensibilidad y con la menor señal de ruido posible. Por lo tanto surge la siguiente pregunta: ¿Con el diseño actual, se puede garantizar el correcto funcionamiento de todos los 18 sensores CCD, con el menor nivel de ruido posible alcanzando una temperatura de trabajo de 130 K – 150 K?

1.2 Objetivo

Con la ayuda de software libre especializado, modelar todos los componentes principales de DAMIC 100 que participan en la transferencia de calor desde los CCD al sistema de enfriamiento, de tal manera que posteriormente se pueda realizar un análisis térmico con el propósito de verificar que todo el sistema funcione óptimamente.

1.3 Justificación

Las WIMP (*Weakly Interacting Massive Particles*, en inglés) son partículas que son un fuerte candidato a ser materia oscura. Se les denomina así a cierto tipo de partículas estables, eléctricamente neutras y que experimentan algún tipo de interacción débil [13]. Su detección en los CCD consiste en que al momento de impactar una partícula WIMP con un átomo de Silicio, el Silicio absorberá energía de la partícula a través de sus electrones, dejando a su vez un hueco o "pozo" de electrones, como se puede observar en la Figura 1.1. Después, los electrones serán almacenados en los capacitores MOS (*Metal-Oxide Semiconductor*, en inglés). Los MOS son acomodados muy cercanos uno a otro para que de esta forma se pueda formar un arreglo de pixeles. La carga almacenada en los pixeles pasará después a un circuito amplificador, donde la carga será extraída y cuantificada [18].

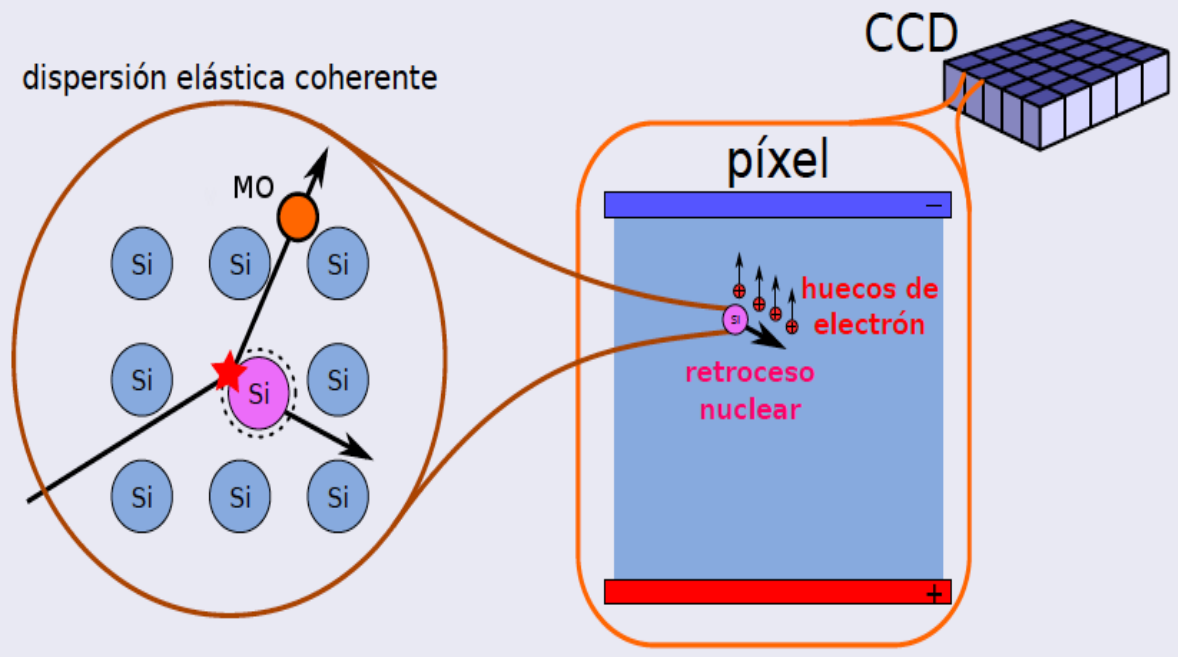


Figura 1.1: Principio de detección de una partícula WIMP [1].

Durante los experimentos suceden diversos tipos de eventos donde dependiendo del tipo de partícula que interactúa con el CCD, se registran diferentes tipos de trazas. En DAMIC 100 nos interesan los eventos localizados, eventos donde se deposita poca energía y en una sola interacción. Por ejemplo, los muones, que son muy penetrantes, registran un movimiento en forma de línea recta y los electrones registran un movimiento en forma de "zigzag". La traza que nos interesa como posible detección de WIMP es una traza de difusión limitada (Figura 1.2).

Los neutrones son el ruido o "*background*" más complicado de suprimir, ya que la señal producida por un neutrón es muy parecida a la de una WIMP. A pesar de que el experimento se encuentra a una gran profundidad y protegido por distintos materiales, los neutrones se generan por la radiación natural de las rocas y del mismo material utilizado en el experimento.

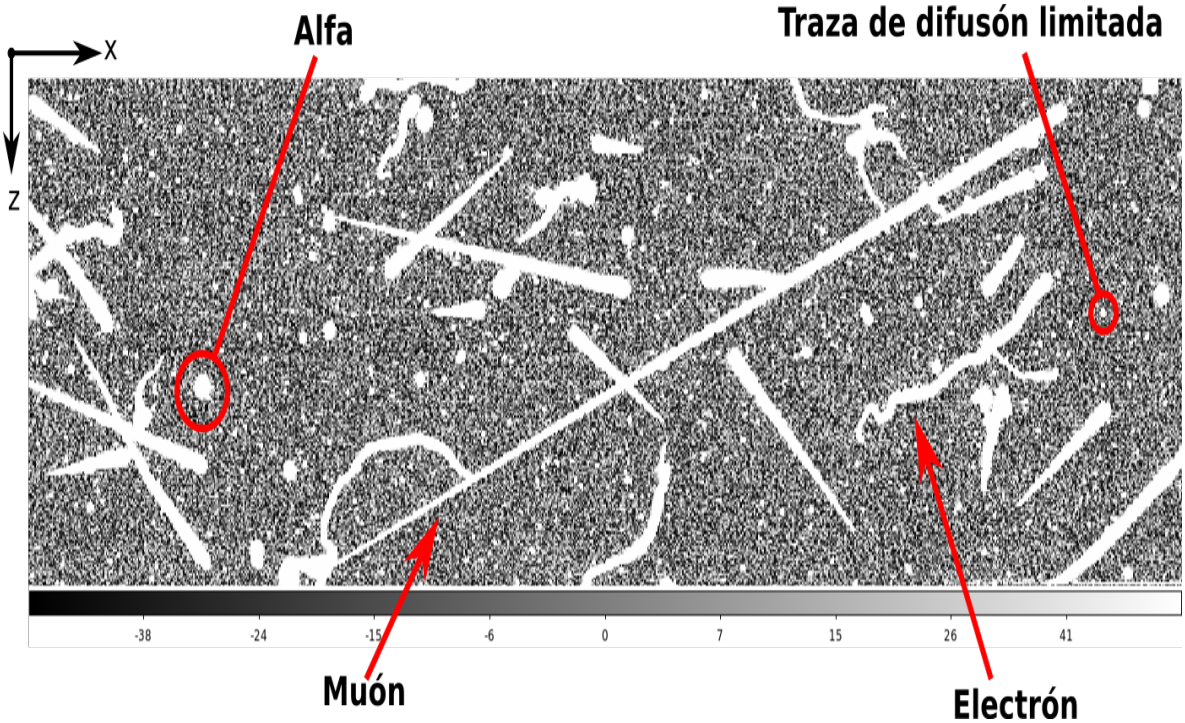


Figura 1.2: Trazas registradas por distintos tipos de partículas [1].

El ruido intrínseco generado por efectos térmicos en las CCD es otro problema que se encuentra en este experimento, es por eso que todo el sistema debe trabajar a ~140 K para evitar que las partículas vibren y se genere ruido. Los productores de sensores CCD miden y reportan el ruido en los sensores como "Electrones RMS" (*Root Mean Square,* en inglés). En el experimento anterior DAMIC se logró obtener un nivel máximo de ruido de $2e^-$ RMS, lo cual ha permanecido hasta la fecha.

Para comprobar que todos los CCDs trabajen a la temperatura deseada, lo ideal sería colocar sensores de temperatura en cada una de ellas. Esto no se puede hacer ya que los sensores generan ruido, el principal problema del experimento. Es por eso la gran importancia de realizar una simulación térmica del experimento, ya que podemos estimar la temperatura a la que se encontrarán los CCD sin la necesidad de utilizar instrumentos que puedan generar más ruido al sistema.

2. Detectores de partículas

2.1 Curvas de exclusión para detectores de partículas.

El interés por detectar partículas surgió a mediados del siglo 19, específicamente para el estudio de cuestiones meteorológicas. Así comenzó el estudio de la condensación del vapor de agua en los laboratorios, al mismo tiempo motivado por el incremento en el uso de los motores de vapor. Con el paso del tiempo, el estudio de partículas continuó desarrollándose hasta llegar al estudio de la física de partículas [15].

Como se había explicado anteriormente, se requiere que los experimentos se realicen en laboratorios que se encuentren a cierta profundidad donde no se tengan efectos por rayos cósmicos y a baja temperatura para lograr reducir el ruido y aumentar la sensibilidad del detector. En cuanto a la detección de las WIMP, actualmente existen muchos experimentos, como se muestra en la Figura 2.1 y que puede ser interpretada de la siguiente manera:

- Las áreas sombreadas con forma de circunferencia, son las zonas donde los experimentos piensan haber detectado posibles candidatos de materia oscura, bajo un cierto nivel de confianza.

- A partir de las líneas curvas y hacia la derecha de la gráfica, son las zonas donde distintos experimentos no detectaron algún posible candidato de materia oscura. El área de la izquierda a partir del trazo de las líneas curvas, son áreas donde se espera poder detectar a posibles candidatos de materia oscura.

- La zona sombreada color amarillo debajo de la línea punteada color naranja, significa que es una zona donde no se podría detectar candidatos de materia oscura, mejor conocida como zona de exclusión (límite teórico).

Algunos de estos experimentos utilizan distintos materiales para facilitar la detección de WIMPs, como gases nobles o sensores de alta pureza. Varían mucho en cuanto al tamaño y la masa activa de detección, pero coincide en que todos estos experimentos se realizan en laboratorios subterráneos para evitar la detección de rayos cósmicos que pudieran alterar los resultados. Los experimentos a parte de DAMIC que han arrojado mejores resultados y que han sido más discutidos son: CDMS II Ge (2009), ZEPLIN-III (2012) y XENON 100 (2012).

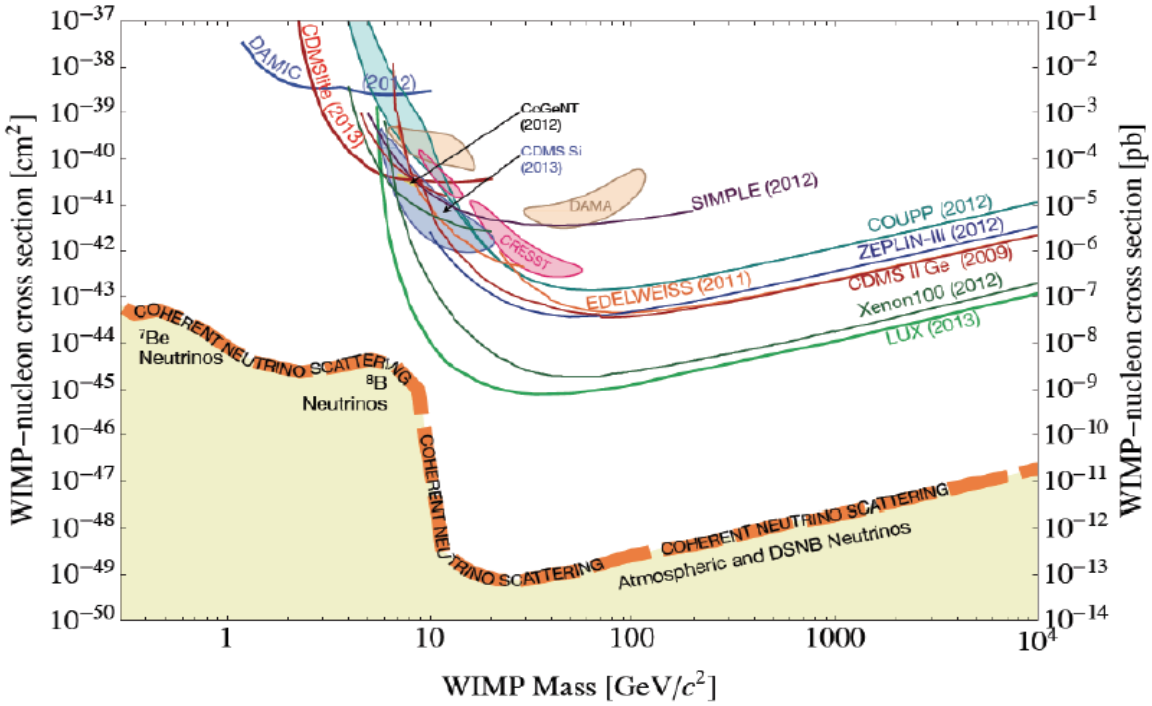


Figura 2.1: Experimentos recientes de detección de materia oscura [10].

2.2 Principales detectores de partículas

El proyecto CDMS (*Cryogenic Dark Matter Search,* en inglés) se localiza en la mina Soudan, Minnesota. Utiliza detectores hechos de Germanio y Silicio con el fin de detectar interacciones con las WIMP. Los detectores son enfriados a temperaturas cercanas al cero absoluto. Su funcionamiento consiste en la interacción de las partículas con los detectores de tal manera que depositen energía en forma de calor y en forma de cargas que se mueven por medio de un campo eléctrico. Sensores especiales detectan estas señales y las amplifican para que puedan ser registradas en una computadora para un futuro estudio [11].

ZEPLIN (*ZonEd Proportional scintillation in LIquid Noble gases,* en inglés) se localiza en la mina Boulby, Inglaterra, a una profundidad de ~1100 m y funciona utilizando Xenón en dos fases (líquido y gas). El experimento consiste en 12 kg de Xenón líquido frío cubierto por una capa delgada de Xenón en forma de vapor. Dentro del líquido se encuentra sumergido un arreglo de 31 tubos fotomultiplicadores. El detector trabaja en campos eléctricos de gran magnitud, lo que le permite obtener una alta precisión del punto donde interactúan las partículas en tres dimensiones con la ayuda de la señal de centelleo del Xenón líquido [16].

Xenón 100 se ubica en el Laboratorio Nacional de Gran Sasso, Italia. El experimento es muy parecido a ZEPLIN y consiste en un volumen activo que contiene 62 kg de Xenón líquido ultra puro (LXe) protegido por un centelleador activo de LXe que contiene 99 kg de Xenón de la misma calidad. Un total de 242 tubos fotomultiplicadores son utilizados para obtener lecturas en los dos volúmenes [4].

2.3 Modulación anual

Se cree que la ubicación de la Tierra en su órbita alrededor del sol es de vital importancia para la detección de partículas WIMP, por lo que la temporada del año en la que se realicen los experimentos puede ser un factor decisivo en cuanto a la detección de estas partículas. El flujo de partículas WIMP se encuentra en su máxima velocidad a principios de Junio, cuando la velocidad tangencial de la Tierra se encuentra en la misma dirección del movimiento del Sistema Solar. A principios de Diciembre la velocidad de las partículas que impactan la tierra se encuentra en la velocidad mínima, esto es debido a que la velocidad tangencial de la Tierra está en sentido contrario a la dirección del movimiento del Sistema Solar [13]. A este fenómeno se le conoce como “Modulación Anual” y se puede visualizar mejor en la Figura 2.2.

Por más de 10 años, el experimento DAMA (*DArk MAtter,* en inglés) ha estado recolectando información para comprobar que el impacto de partículas WIMP con la tierra depende completamente de la ubicación de la misma. Este experimento trabaja con 250 kg de cristales de Yoduro de Sodio (NaI) y se encuentra ubicado en el Laboratorio Nacional de Gran Sasso, Italia. La cantidad de información recolectada es enorme y el significado estadístico del resultado es irrefutable. Tiempo después fue sustituido por el experimento DAMA/LIBRA que terminó por confirmar los resultados obtenidos anteriormente en DAMA [13]. Actualmente existen muchos experimentos que han tratado de confirmar los datos del experimento DAMA pero, hasta la fecha, ninguno ha sido capaz de hacerlo.

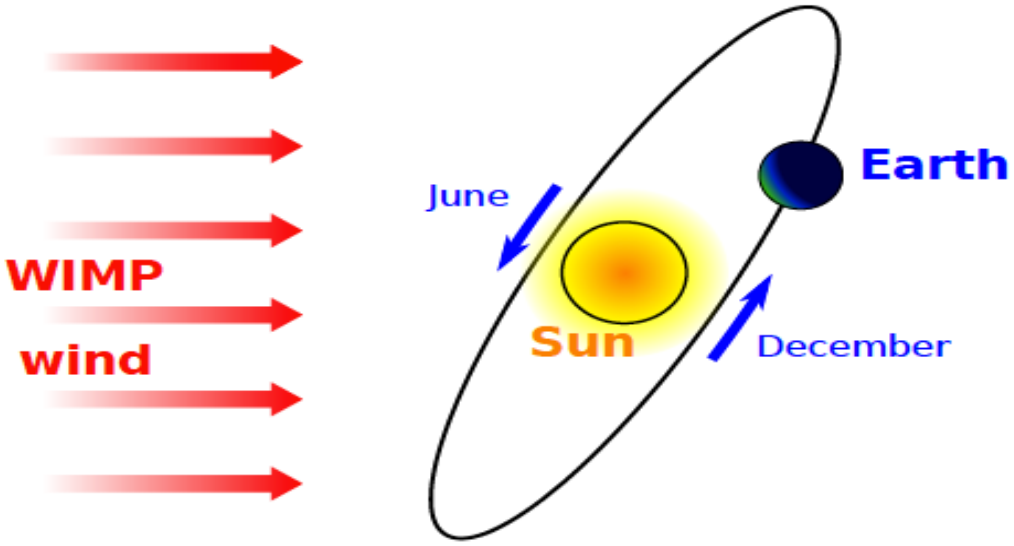


Figura 2.2: Flujo de partículas WIMP de acuerdo a la posición de la Tierra [2].

El experimento DM-Ice (Materia Oscura-Hielo, por sus siglas en inglés) trata de confirmar los datos del experimento DAMA, utilizando la misma estrategia de análisis. El experimento se realiza en el hemisferio sur, lo que invertiría los efectos estacionales sin afectar las señales de materia oscura. DM-Ice17, la primera etapa del programa experimental DM-Ice, fue desarrollado en el Polo Sur en diciembre del 2010. Dos cristales de Yoduro de Sodio de 8.47 kg, fueron colocados en el fondo del detector “Cubo de Hielo”, ubicado a 2457 m de profundidad del hielo Antártico. Después de 3.6 años de recolección de datos (16 Junio 2011 – 28 Enero 2015), se comprobó la confiabilidad del Polo Sur como un sitio para experimentos futuros. Además, no se encontró evidencia de Modulación Anual, aunque los sensores utilizados no fueron lo suficientemente sensibles para confirmar o negar los resultados obtenidos en DAMA, por lo que la Modulación Anual aún es un tema que no está completamente confirmado [6].

3. Salome-Meca

Salome-Meca es un software de origen francés que pertenece a una plataforma libre llamada Open Cascade y que empezó a desarrollarse desde el 2001, con la finalidad de facilitar y mejorar las capacidades de simulación en un software libre. Actualmente puede ser aplicado en la mecánica estructural, termo-hidráulica, ciencia de materiales, geología y electromagnetismo principalmente [9].

Salome es distribuido como un software libre bajo los términos de la licencia GNU LGPL (*Lesser General Public License* en inglés), lo que le da la ventaja de que el programa puede ser modificado, mejorado o adaptado a las necesidades del usuario. Salome posee dos diferentes modos de interacción con los componentes del software:

- Interfaz GUI (*Graphical User Interface,* en inglés) con interacción 3D (Qt4, VTK).
- Interfaz TUI (*Text User Interface,* en inglés) basada en el lenguaje Python.

Salome puede ser utilizado como una aplicación para la generación de modelos en 1D, 2D y 3D, su preparación para cálculos numéricos, así como para el post-procesamiento de los resultados del cálculo. También puede ser utilizado como una plataforma que integre códigos numéricos externos para formar una nueva aplicación, mejor conocido como solucionador [26]. Dentro de Salome existen diferentes módulos que nos permitirán realizar todo lo mencionado anteriormente. En la Figura 3.1 podemos observar todos los módulos disponibles para utilizar en Salome. Para DAMIC 100 sólo se utilizarán los siguientes módulos en el siguiente orden:

1. Módulo de Geometría (*Geometry Module*).
2. Módulo de Malla (*Mesh Module*).
3. Módulo Aster.
4. Módulo ParaViS.

Figura 3.1: Módulos de Salome utilizados en DAMIC 100.

3.1 Módulo de Geometría (*Geometry Module*)

El módulo de geometría contiene una gama completa de comandos para poder crear, editar, importar, exportar o modificar un modelo 1D, 2D ó 3D. La pantalla principal de este módulo así como las diferentes herramientas disponibles se puede visualizar en la Figura 3.2:

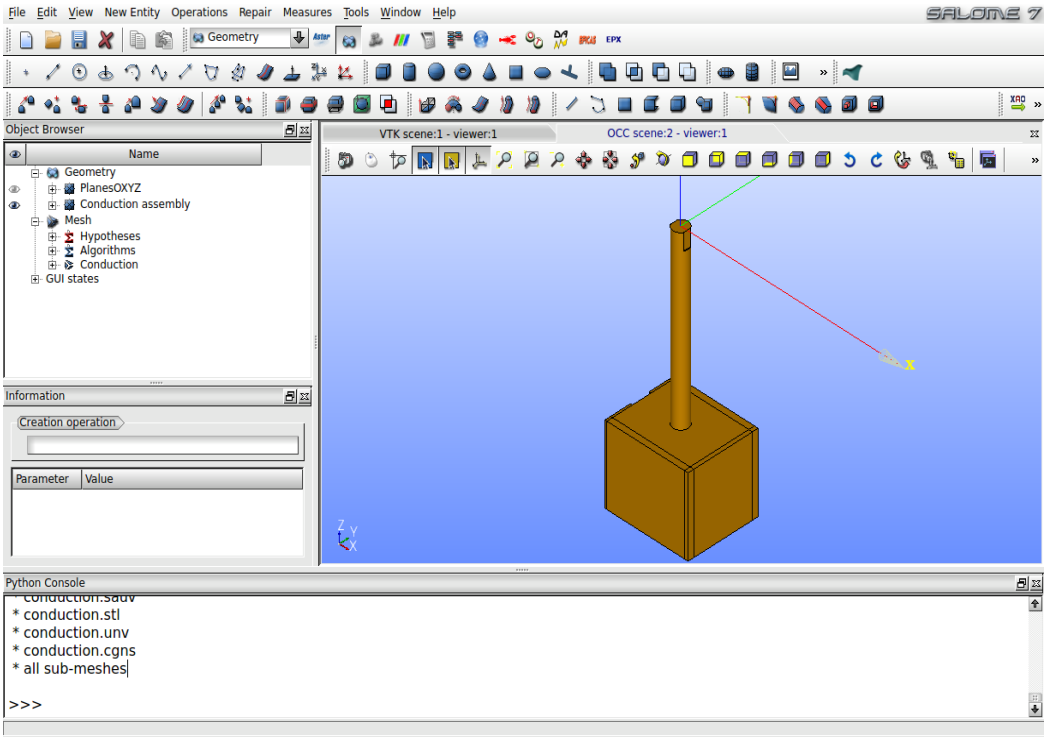


Figura 3.2: Geometría del detector DAMIC 100 a estudiar, donde se puede observar el dedo frío unido a la caja protectora del arreglo de CCDs.

En este módulo se pueden realizar las siguientes opciones [26]:

- Importar y exportar modelos geométricos en los formatos: IGES, BREP, STL, XAO y VTK.
- Construcción de diferentes objetos geométricos desde puntos y líneas, hasta cubos, esferas, etc.
- Visualización de los objetos geométricos a través del Visualizador OCC (*Open CasCade*).
- Transformación de objetos geométricos como lo es: Traslación, rotación, espejo y operaciones booleanas como lo es corte, fusión, etc.
- Reparación de objetos geométricos como lo es la supresión de caras y líneas, cerrar contornos, pegar caras y ejes, remover caras internas y externas, unir caras, etc.
- Herramientas de medición como lo son: propiedades básicas, coordenadas, centro de masa, inercia, tolerancias, etc.

Casi todas las funcionalidades que se pueden utilizar en la interfaz GUI, pueden ser igualmente utilizadas en la interfaz TUI. El paquete GEOM Python esencialmente contiene:

- Interfaz geomBuilder.py para importar o exportar, crear y transformar objetos geométricos, uso de herramientas de medición, etc.
- Funciones de utilidad dentro del módulo Python geomtools.py para manejar los objetos GEOM dentro del estudio de Salome como: agregar o quitar una forma, mostrar u ocultar una forma en el visualizador o borrar por completo una forma.

A continuación se mostrará el código Python para poder modelar el primer arreglo de sensor CCD de un arreglo de dieciocho, visualizando el resultado final con el visualizador OCC del Módulo Geometría (Figura 3.3):

Para la placa base de cobre:

```
# Creación de los vértices de la figura #
point1_solid012 = geompy.MakeVertex((0.5)*lengthOfSolid012,
(0.5)*widthOfSolid012, (-1)*fingerLength+(-
1)*depthOfSolid012+(1)*thicknessOfSolid012)
point2_solid012 = geompy.MakeVertex((-0.5)*lengthOfSolid012, (-
0.5)*widthOfSolid012, (-1)*fingerLength+(-1)*depthOfSolid012)

# Creación del volumen de la figura #
solid012_1 = geompy.MakeBoxTwoPnt(point1_solid012, point2_solid012)
solid012 = geompy.MakeCut(solid012_1, tool012)
```

Para la base de silicio:

```
# Creación de los vértices de la figura #
point1_solid013 =
geompy.MakeVertex((0.5)*lengthOfSolid013+(1)*separationFromFlex,
(0.5)*widthOfSolid013, (-1)*fingerLength+(-
1)*depthOfSolid012+(1)*thicknessOfSolid012+(-1)*thicknessOfTool012)
point2_solid013 = geompy.MakeVertex((-
0.5)*lengthOfSolid013+(1)*separationFromFlex, (-
0.5)*widthOfSolid013, (-1)*fingerLength+(-
1)*depthOfSolid012+(1)*thicknessOfSolid012+(-
1)*thicknessOfTool012+(1)*thicknessOfSolid013)

# Creación del volumen de la figura #
solid013 = geompy.MakeBoxTwoPnt(point1_solid013, point2_solid013)
```

Para el sensor CCD:

```
# Creación de los vértices de la figura #
point1_solid014 =
geompy.MakeVertex((0.5)*lengthOfSolid014+(1)*separationFromFlex,
(0.5)*widthOfSolid014, (-1)*fingerLength+(-
1)*depthOfSolid012+(1)*thicknessOfSolid012+(-
1)*thicknessOfTool012+(1)*thicknessOfSolid013+(1)*thicknessOfSolid0
14)
```

```
point2_solid014 = geompy.MakeVertex((-
0.5)*lengthOfSolid014+(1)*separationFromFlex, (-
0.5)*widthOfSolid014, (-1)*fingerLength+(-
1)*depthOfSolid012+(1)*thicknessOfSolid012+(-
1)*thicknessOfTool012+(1)*thicknessOfSolid013)

# Creación del volumen de la figura #
solid014 = geompy.MakeBoxTwoPnt(point1_solid014, point2_solid014)
```

Para el cable de Kapton:

```
# Creación de los vértices de la figura #
point1_solid015 = geompy.MakeVertex((0.5)*lengthOfSolid015,
(0.5)*widthOfSolid015, (-1)*fingerLength+(-
1)*depthOfSolid012+(1)*thicknessOfSolid012+(1)*thicknessOfSolid015)
point2_solid015 = geompy.MakeVertex((-0.5)*lengthOfSolid015, (-
0.5)*widthOfSolid015, (-1)*fingerLength+(-
1)*depthOfSolid012+(1)*thicknessOfSolid012)

point1_solid015_3 = geompy.MakeVertex((-0.5)*lengthOfSolid015+(-
1)*lengthOfCable001, (0.5)*widthOfCable001, (-1)*fingerLength+(-
1)*depthOfSolid012+(1)*thicknessOfSolid012+(1)*thicknessOfSolid015)
point2_solid015_3 = geompy.MakeVertex((-
0.5)*lengthOfSolid015+(1)*separationFromCable, (-
0.5)*widthOfCable001, (-1)*fingerLength+(-
1)*depthOfSolid012+(1)*thicknessOfSolid012)

# Creación del volumen de la figura #
solid015_1 = geompy.MakeBoxTwoPnt(point1_solid015, point2_solid015)
solid015_2 = geompy.MakeCut(solid015_1, tool015)
solid015_3 = geompy.MakeBoxTwoPnt(point1_solid015_3,
point2_solid015_3)
solid015 = geompy.MakeFuse(solid015_3, solid015_2)
```

Para la cubierta de cobre:

```
# Creación de los vértices de la figura #
point1_solid016 =
geompy.MakeVertex((0.5)*lengthOfSolid016+(1)*separationFromCable,
(0.5)*widthOfSolid012, (-1)*fingerLength+(-
1)*depthOfSolid012+(1)*thicknessOfSolid012+(1)*thicknessOfSolid016)
point2_solid016 = geompy.MakeVertex((-
0.5)*lengthOfSolid008+(1)*separationFromCable+(1)*separationFromFle
x, (-0.5)*widthOfSolid012,(-1)*fingerLength+(-
1)*depthOfSolid012+(1)*thicknessOfSolid012)

# Creación del volumen de la figura #
solid016_1 = geompy.MakeBoxTwoPnt(point1_solid016, point2_solid016)
solid016 = geompy.MakeCut(solid016_1, tool016)
```

Para la barra de montaje de cobre:

```
# Creación de los vértices de la figura #
point1_solid017 = geompy.MakeVertex((-0.5)*lengthOfSolid012,
(0.5)*widthOfSolid017, (-1)*fingerLength+(-
1)*depthOfSolid012+(1)*thicknessOfSolid012+(1)*thicknessOfSolid015)
```

```
point2_solid017 = geompy.MakeVertex((-
0.5)*lengthOfSolid012+(1)*lengthOfSolid017, (-
0.5)*widthOfSolid017,(-1)*fingerLength+(-
1)*depthOfSolid012+(1)*thicknessOfSolid012+(1)*thicknessOfSolid015+
(1)*thicknessOfSolid017)

# Creación del volumen de la figura #
solid017 = geompy.MakeBoxTwoPnt(point1_solid017, point2_solid017)
```

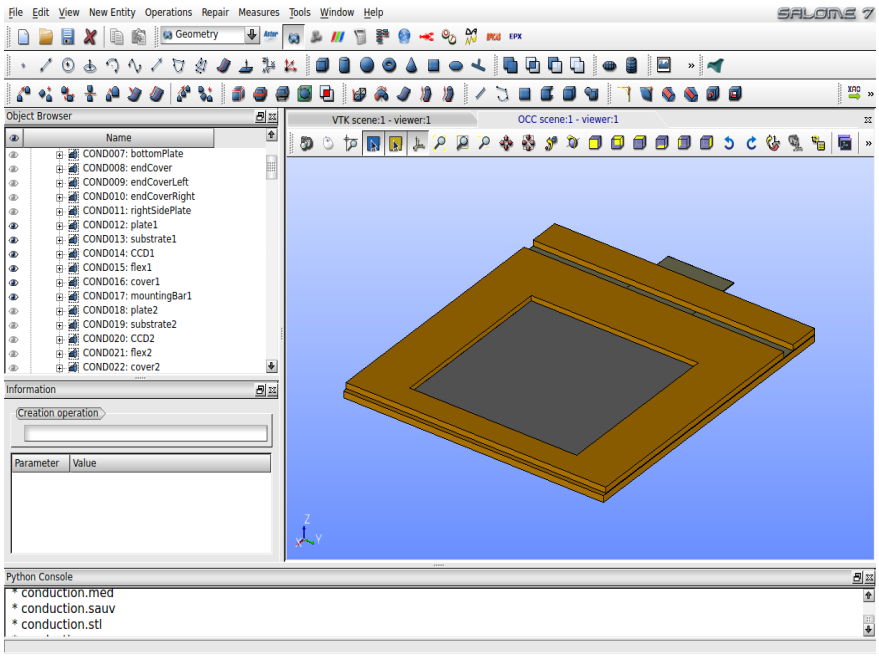


Figura 3.3: Visualización de un arreglo CCD en Salome. De color naranja tenemos la base, cubierta y barra de montaje de cobre. De color gris, el sensor CCD con su respectiva base de silicio. De color verde, el cable transmisor de datos de Kapton.

3.2 Módulo de Malla (*Mesh Module*)

En este módulo se pueden mallar los modelos 1D, 2D o 3D, obtenidos en el Módulo Geometría. Los formatos disponibles para importar o exportar son los siguientes: MED, UNV, DAT, STL, CGNS, GMF y SAUV. Se pueden crear o editar los datos del mallado que incluye una gran variedad de algoritmos libres y comerciales que pueden ser utilizados dependiendo del gusto del usuario o de la geometría de nuestro modelo a mallar. Al igual que el Módulo Geometría, el Módulo Malla cuenta con distintas herramientas que facilitan la creación y edición de mallas complejas. Las mallas pueden ser organizadas por grupos con el fin de facilitar la visualización de cada una de las partes del modelo y ayudar a la definición inicial de las condiciones de frontera. Todas las herramientas de mallado pueden ser programadas con la interfaz TUI [8].

La estructura de una malla en Salome consiste en nodos y en los elementos que se basaron en dichos nodos. Una malla puede incluir los siguientes elementos [27]:

- Nodo: Es un punto de una malla con coordenadas (x, y ,z).
- Eje: Elemento 1D que une dos nodos.
- Cara: Elemento 2D de una malla que representa una parte de la superficie unida por enlaces entre los nodos de dicha cara. Una cara puede ser un triangulo, rectángulo o un trapezoide.
- Volumen: Elemento 3D de la malla que representa una parte del espacio 3D unido por un conjunto de caras. Un volumen puede ser un tetraedro, hexaedro, pentaedro, pirámide, prisma hexagonal o poliedro.
- Elemento 0D: Elemento de una malla definido por un nodo.

La pantalla principal del Módulo Malla puede visualizarse como en la Figura 3.4. La visualización del resultado final del mallado se puede visualizar mediante el visualizador VTK (*Visualization ToolKit* en inglés).

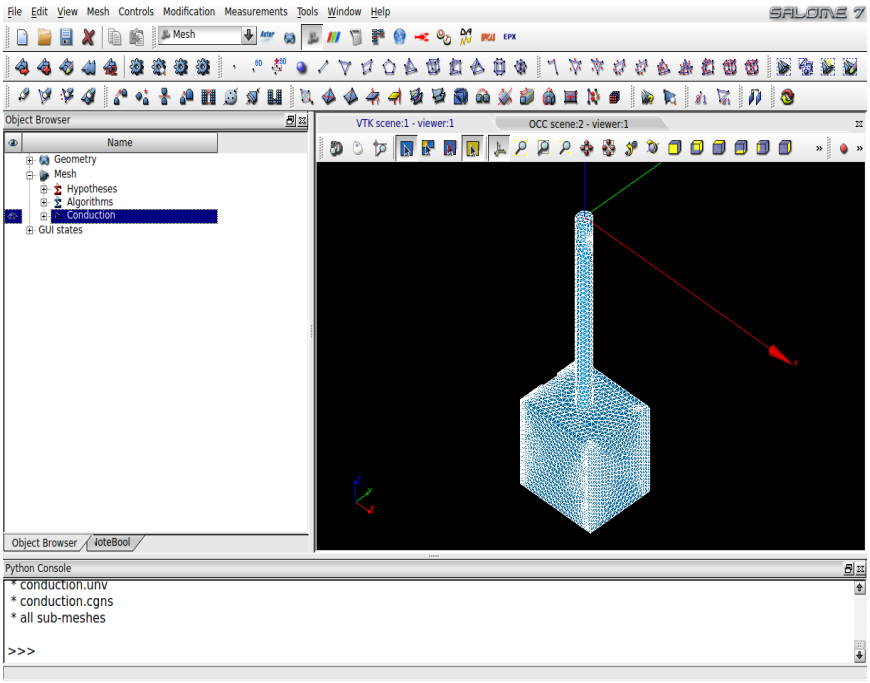


Figura 3.4: Visualización de la Figura 3.2 mallada con el algoritmo NETGEN 1D-2D-3D [3].

Dependiendo de la dimensión en la que se quiera trabajar (1D, 2D, 3D), existen distintos algoritmos que se puede aplicar. En este trabajo únicamente se aplicará el algoritmo NETGEN 1D-2D para los modelos 2D y el algoritmo NETGEN 1D-2D-3D para los modelos en 3D.

Aplicando el algoritmo NETGEN 1D-2D-3D para mallar al sensor CCD dibujado en el Módulo Geometría, se obtiene el siguiente resultado:

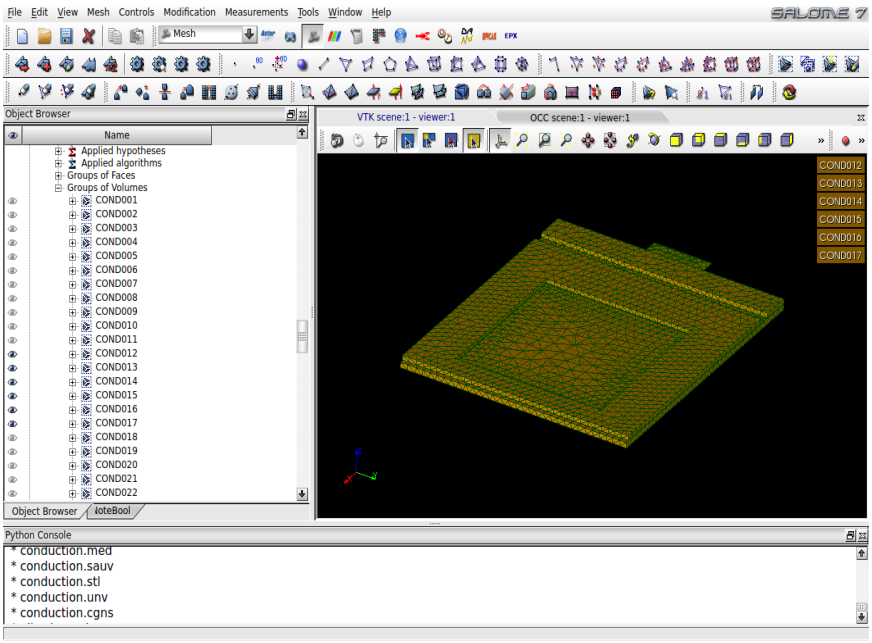


Figura 3.5: Figura 3.3 mallada con el algoritmo NETGEN 1D-2D-3D.

3.3 Módulo Aster

El módulo Aster es un módulo que utiliza como solucionador a Code_Aster, en él se pueden crear estudios de análisis lineales o no lineales de casos térmicos y/o mecánicos, ya sea desde Salome a través de la interfaz GUI, o simplemente ejecutando el archivo que contenga al código con extensión *.comm, escrito previamente, al modelo mallado en estudio. En DAMIC 100 se utilizan únicamente dos herramientas como se muestra en la Figura 3.6. La herramienta número 1 nos sirve para crear un nuevo caso de estudio donde podemos utilizar el archivo *.comm escrito previamente y la herramienta 2 nos sirve para ejecutar el estudio que nos dará un resultado que podremos visualizar posteriormente en el módulo ParaViS.

Figura 3.6: Principales herramientas del Módulo Aster.

Code_Aster, acrónimo de Análisis de Estructuras y Termomecánica para Estudios e Investigación (*Analysis of Structures and Thermomechanics for Studies and Research),* es un software de origen francés de Análisis por Elemento Finito creado por el departamento de investigación y desarrollo de la EDF, *Électricité De France*. Empezó a desarrollarse en 1989 y en el 2001 EDF tomó la decisión de hacer de este software un software libre bajo la licencia GNU LGPL [5].

Code_Aster ofrece toda una gama de análisis multifísicos y métodos de modelado que van mas allá de las funciones estándar de un código de cálculos termo-mecánicos, desde análisis sísmicos hasta análisis de medios porosos a través de la acústica, fatiga, entre otras [9].

El archivo Code_Aster de extensión *.comm (*Command File)* que será utilizado para el estudio, puede ser generado de 3 maneras distintas: desde Salome con los distintos asistentes para casos elásticos y térmicos, desde el editor gráfico llamado *Eficas*, y por último, desde cualquier editor de texto. En este trabajo se utilizará la herramienta *Eficas* debido a que facilita en gran medida la escritura del código a usuarios principiantes, ya que el software cuenta con una lista completa de todos los comandos que se pueden utilizar con sus respectivas opciones. Otra ventaja del uso de *Eficas* es que reduce la posibilidad de errores de sintaxis, el código es correcto cuando los cuadros de todos los comandos está de color verde como se puede observar en la Figura 3.7.

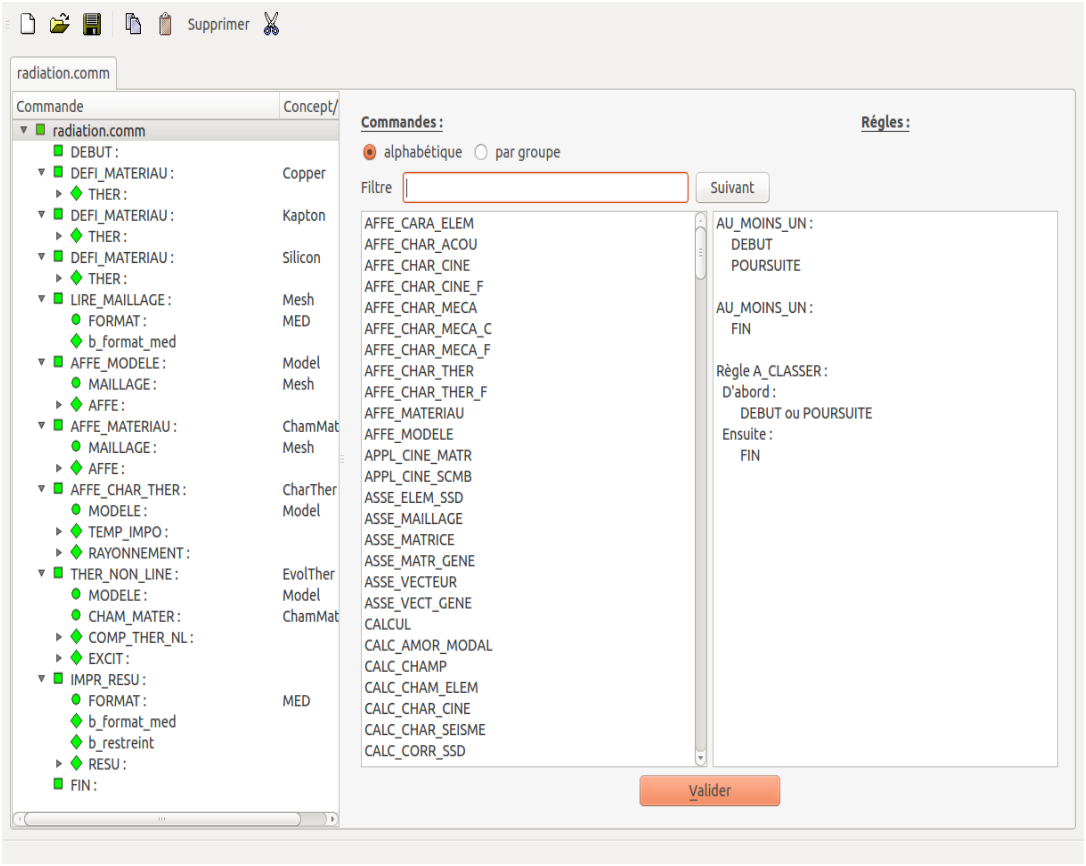


Figura 3.7: Pantalla principal de Eficas.

3.4 Módulo ParaViS

ParaViS es el módulo de post-procesamiento de datos basado en ParaView (*Parallel Visualization Application*). ParaView es una multiplataforma de análisis de datos y aplicación de visualización de código abierto bajo la licencia GNU LGPL. Su desarrollo comenzó en el 2000 en una colaboración entre Kitware Inc. y el Laboratorio Nacional de Los Alamos. La primera versión pública, ParaView 0.6, fue anunciada en Octubre del 2002 [22]. En esta tesis, este módulo sólo será utilizado para poder visualizar el resultado final en 3D, así como para poder realizar gráficas de los datos obtenidos.

Una vez ejecutado con éxito el archivo *.comm en el caso de estudio, se creará automáticamente un archivo con la extensión *.rmed, el cual se deberá abrir manualmente en el módulo ParaViS para poder visualizar el resultado final. Adicionalmente si se requiere, se pueden crear animaciones para poder visualizar los resultados en nuestro estudio a lo largo de cierto tiempo definido por el usuario. En el capítulo 5 podremos observar la visualización del módulo ParaVis con los resultados finales.

3.5 Código Python de DAMIC 100: Interfaz textual

Con ayuda del lenguaje de programación Python, que también es un lenguaje de código abierto o libre, se realizó el modelado de DAMIC 100. Modelar y mallar un objeto puede realizarse con la interfaz GUI, pero para este trabajo, el modelado y mallado se hará a través de la interfaz TUI para tener toda la información ordenada y para que la edición sea más rápida y sencilla, ya que son muchos sólidos los que serán modelados y mallados. El cerebro de todo el código que genera a nuestro modelo es el archivo *main.py*, ya que es el que ejecuta todas las bibliotecas y archivos que modelan y mallan a DAMIC 100. El código se puede consultar en el Apéndice A [29].

Primero, en la sección INCLUDE se “llama” a todas las bibliotecas de Salome que nos permitirán utilizar todas las herramientas de los módulos geometría y malla. También se ejecutarán las bibliotecas que nos permitirán utilizar Python. En la sección PATHS se escriben las distintas rutas donde se organizarán todos los archivos del código según su formato en distintas carpetas. La carpeta principal es “SalomeFiles” y dentro de ella se encuentra la carpeta “PythonFiles”. En esta carpeta se encuentran todos los archivos que nos permitirán modelar cada una de las piezas de DAMIC 100, ensamblarlos, organizarlos por grupos geométricos, mallarlos e igualmente organizar las mallas por grupos, lo que facilitará la simulación térmica posteriormente. Cada pieza del

modelo está escrita en un archivo que tendrá el siguiente formato: *solid0xx_NameOfTheSolid.py*

Todos los parámetros utilizados en las distintas piezas del modelo se declaran en el archivo *parameters.py* como lo es el tipo de material (cobre, kapton o silicio), el color que tendrá cada pieza según su material, así como longitudes, radios, etc., que formarán la geometría de cada sólido. Estas magnitudes están multiplicadas por un factor que nos permitirá cambiar fácilmente de un sistema de unidades a otro si se desea llamado "*unitConversion".* Actualmente el modelo se encuentra en el Sistema Internacional de unidades por lo que el valor de *unitConversion* es igual a uno.

Otro aspecto importante que se encuentra dentro de *parameters.py* es que se puede escoger lo que se desea ver en el visualizador, sea las distintas piezas hechas antes de ensamblar o las herramientas utilizadas para dibujar dichas piezas, lo que facilita visualizar el trabajo hecho pieza por pieza. Similar a *parameters.py*, los parámetros de todas las herramientas utilizadas para formar un sólido como lo son cortes, fusiones, espejos, etc., son declaradas en el archivo *tools.py.*

Posteriormente está la sección OCC VIEWER, en donde se inicializa Salome y los parámetros visuales. Igualmente se establecen las propiedades visuales del visualizador OCC para el módulo de geometría y del visualizador VTK para el módulo de malla. En la sección FILES, el usuario puede elegir si desea importar archivos externos, usar las herramientas, crear las geometrías, ensamblar las geometrías, visualizar las geometrías o mallar las geometrías. Dependiendo de las preferencias del usuario, el programa irá ejecutando los archivos correspondientes.

4. Diseño mecánico de DAMIC 100

4.1 Diseño mecánico original

El diseño original hecho por Greg Derylo, de Fermilab, se puede visualizar de la siguiente forma:

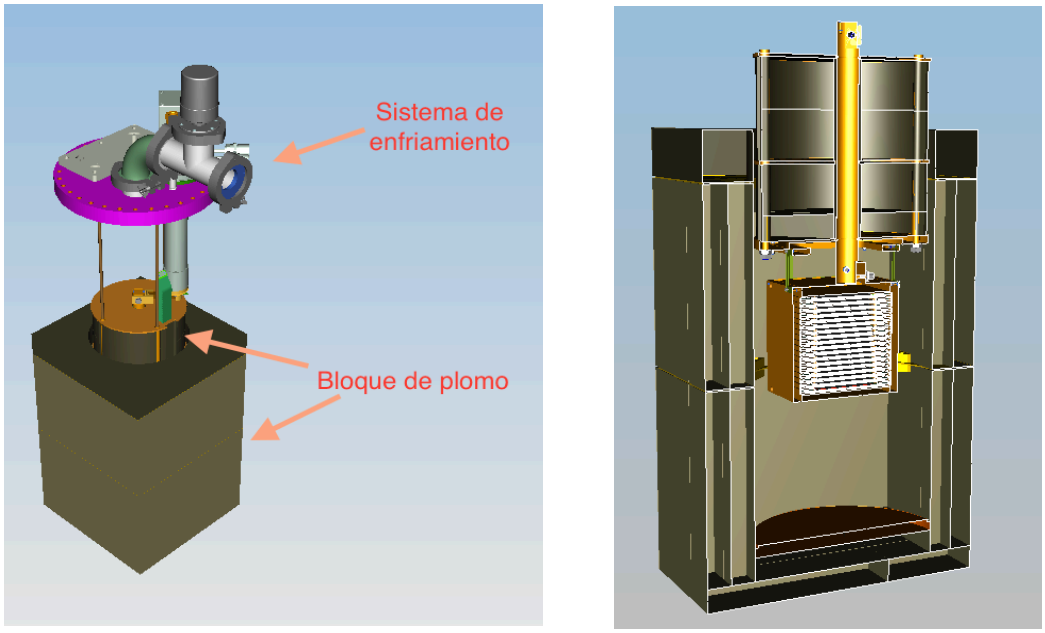


Figura 4.1: Bloque de plomo que cubre al experimento.

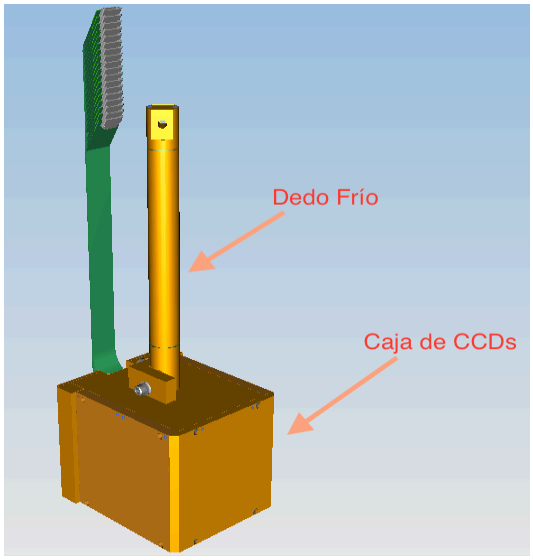


Figura 4.2: Dedo frío y caja de CCDs que se encuentran dentro del bloque de plomo.

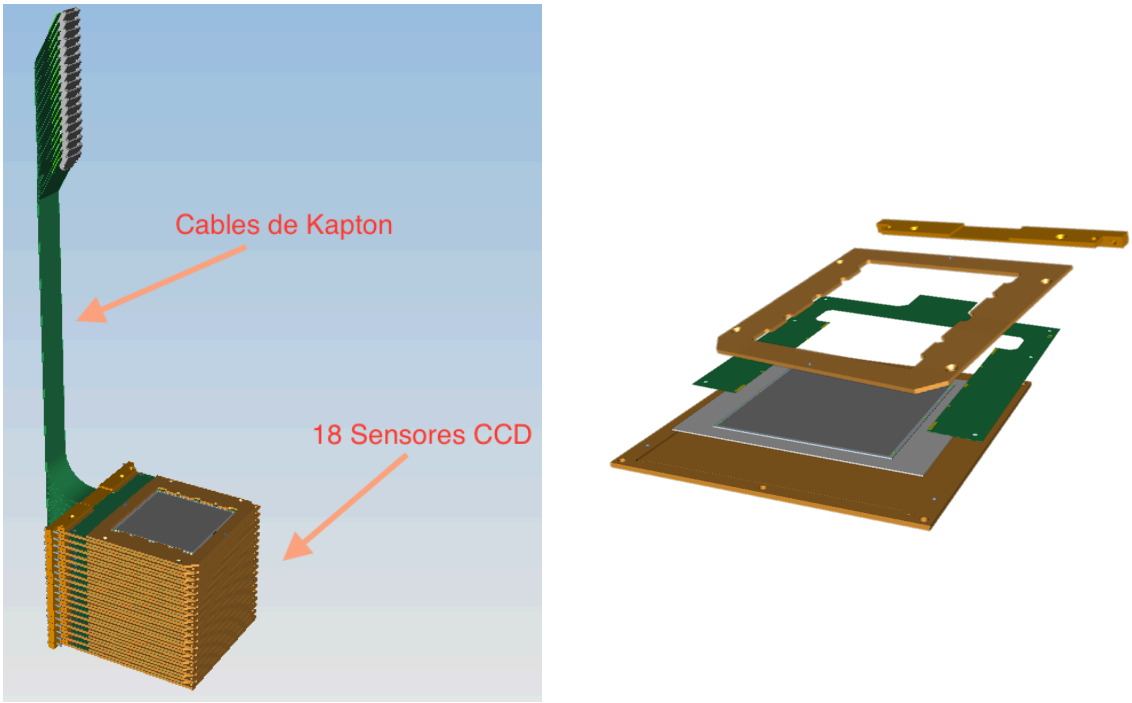


Figura 4.3: 18 arreglos de sensores CCD con sus respectivos cables cubiertos de Kapton.

DAMIC 100 consiste en 18 CCDs diseñados y fabricados en el LBNL, de 16 Mpixel, de 650 μm de espesor que hacen una masa total de 100 g, como se puede observar en la Figura 4.3; utilizará el mismo recipiente de vacío y los mismos materiales protectores de DAMIC, que se encuentran actualmente en SNOLAB. Los CCDs serán instalados en soportes de Silicio de alta pureza que a su vez serán cubiertos por una placa de cobre OFHC (*Oxygen Free High Conductivity,* en inglés) [2].

La caja de cobre, que se puede observar en la Figura 4.2 y que contiene a las CCDs, se coloca en otro contenedor de cobre que se encontrará a ~10^{-7} torr de vacío. Dentro del contenedor de vacío y encima de la caja de CCDs, se colocará un bloque de plomo de 21 cm de espesor que funciona como un escudo de la radiación provocada por la tarjeta de interfaz de vacío o VIB (*Vacuum-Interface Board,* en inglés) que se encuentra en un costado de la parte superior del escudo de plomo. Un "dedo" frío de cobre atraviesa este bloque de plomo, el cual se encarga de mantener la caja de CCDs operando a una temperatura de ~140 K. Para transmitir la señal de la caja de CCDs a la VIB, son utilizados cables delgados cubiertos de Kapton. El contenedor de vacío también se encuentra rodeado por otro bloque de plomo, como se observa en la Figura 4.1, que lo aislará de rayos gamma del exterior. A su vez, este bloque de plomo será rodeado por 42 cm de polietileno de alta densidad que aislará a la caja de CCDs, de neutrones del exterior.

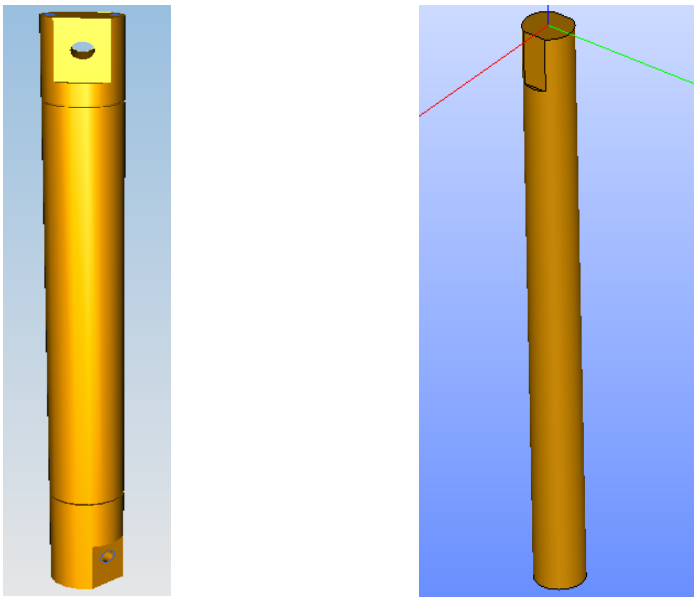
En total, todo el sistema a modelar y estudiar se compone de lo siguiente:

- "Dedo frío" de cobre

- 6 placas de cobre que forman la caja donde se encuentran los 18 CCD.
- 18 arreglos o módulos que consisten en lo siguiente: una base de cobre, un soporte de Silicio, un sensor CCD igualmente de Silicio, el cable de Kapton, una cubierta de cobre y una barra de cobre que sujetará al cable con la base.

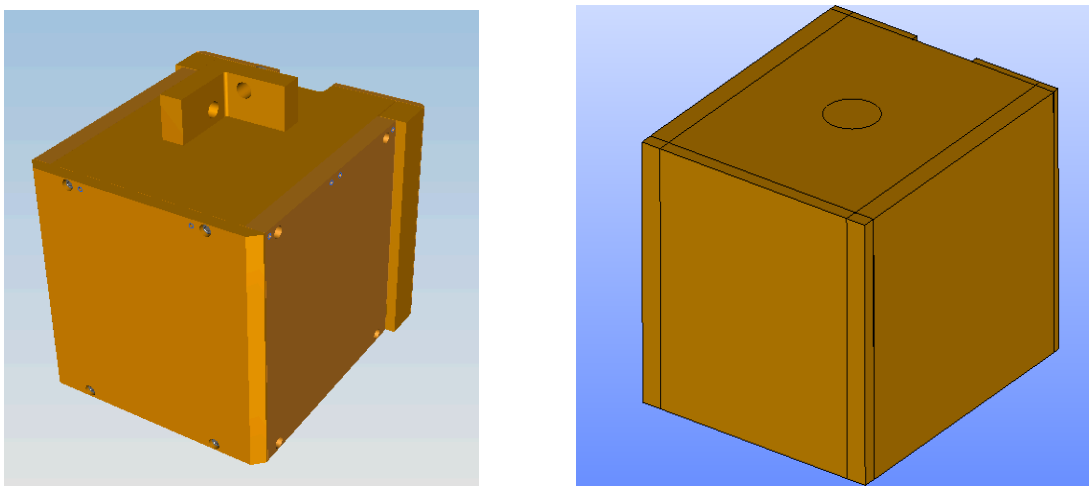
4.2 Simplificaciones del diseño original

Para facilitar el modelado de DAMIC 100 en Salome, se decidió hacer un diseño más robusto del diseño original, simplificando geometrías complejas utilizando geometrías básicas y eliminando detalles como perforaciones, chaflanes, tornillos, entre otros. Todo esto no tiene efectos significativos en los resultados de la simulación térmica. En las siguientes figuras se muestran las comparaciones entre los diseños originales hechos por Greg Derylo de Fermilab y los diseños simplificados propuestos modelados en Salome.



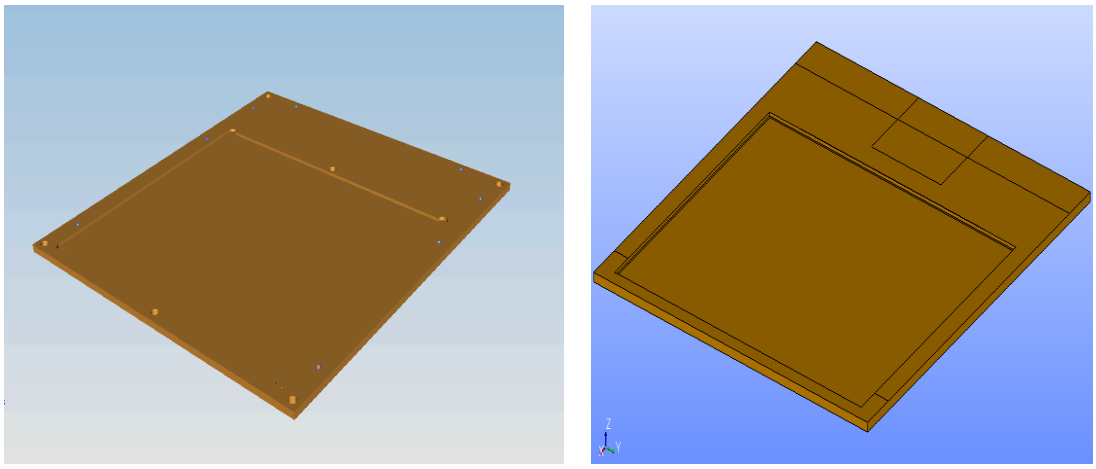
a) b)

Figura 4.4: Dedo frío de cobre. a) Diseño original, b) Diseño simplificado propuesto.



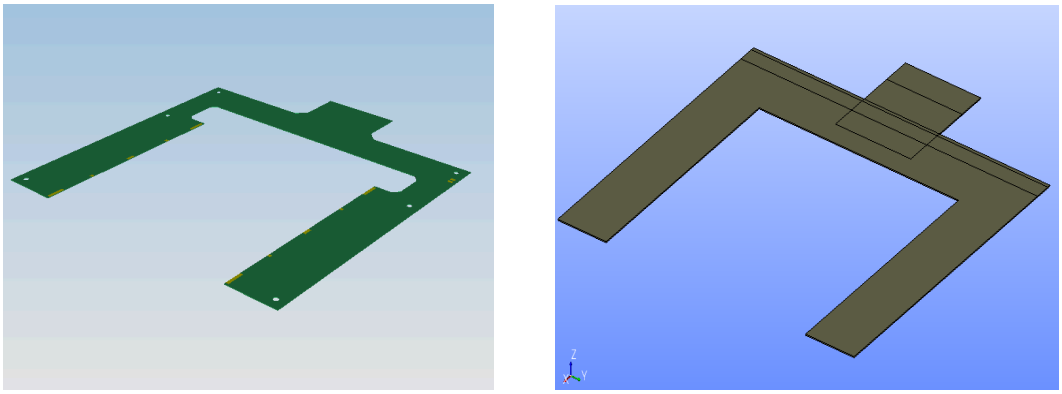
a) b)

Figura 4.5: Caja de cobre que cubre a los 18 CCDs. a) Diseño original. b) Diseño simplificado propuesto.



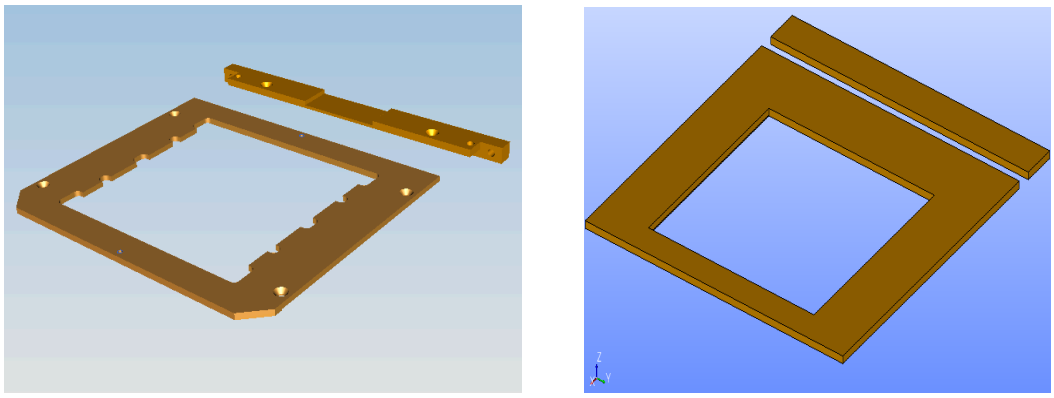
a) b)

Figura 4.6: Base de cobre del arreglo CCD. a) Diseño original. b) Diseño simplificado propuesto.



a) b)

Figura 4.7: Cable de Kapton. a) Diseño original. b) Diseño simplificado propuesto.



a) b)

Figura 4.8: Placa que cubre a la base de Silicio y barra que sujeta al cable de Kapton con la base del arreglo. Ambas de cobre. a) Diseño original. b) Diseño propuesto.

5. Análisis térmico

El objetivo de este análisis es determinar el campo de temperaturas a través del modelo de DAMIC 100 debido a las condiciones impuestas en sus fronteras. Esto quiere decir que se desea saber cómo varía la temperatura respecto a la posición. La parte superior del dedo frío estará en contacto con el sistema de enfriamiento. A partir de ahí y por conducción térmica, se enfriará el resto del sistema hasta llegar a la temperatura deseada en los sensores CCD (~140 K). Las 6 caras de la caja que cubre a los 18 sensores CCD, estarán recibiendo del infinito radiación térmica a temperatura ambiente (~293 K) con una dirección ortogonal a las caras. Cabe mencionar que el experimento se encuentra en vacío, por lo que se despreciará la radiación en los espacios entre los 18 arreglos CCD.

5.1 Modelado matemático de la ecuación de calor

Se considerará un medio homogéneo en el cual existe un gradiente de temperatura y la distribución de temperatura $T(x, y, z)$ está expresada en coordenadas cartesianas. Se considerará que en el experimento no hay generación de energía. También se considerará que todos los contactos térmicos son perfectos y se despreciará la resistencia térmica de los materiales a utilizar en este experimento. Las unidades de las ecuaciones son del Sistema Internacional (S.I.).

5.1.1 Conducción térmica

Por Ley de Fourier sabemos:

$$\vec{q} = -k\nabla(\mathrm{T})$$

donde:

$$k = conductividad\ térmica\ del\ material\ (considerada\ constante)\ [{}^{W}/_{m \cdot K}]$$
$$T = Temperatura\ [°\mathrm{C}]$$
$$\vec{q} = vector\ de\ flujo\ de\ calor\ en\ las\ direcciones\ x, y, z\ \ [{}^{W}/_{m^2}]$$

5.1.2 Radiación térmica

Sabemos que la ley de Stefan-Boltzmann se expresa de la siguiente forma:

$$q_n = \varepsilon\sigma(T^4 - T_\infty^4)$$

donde:

$$\sigma = constante\ de\ Stefan - Boltzmann.$$
$$\sigma = 5.67 \times 10^{-8}\ [^W/_{m^2 \cdot K^4}]$$
$$\varepsilon = emisividad\ del\ material\ (considerada\ constante).$$
$$T_\infty = temperatura\ del\ exterior\ considerada\ constante\ a\ 20\ ºC.$$
$$q_n = flujo\ de\ calor\ en\ las\ direcciones\ x, y, z\ \ [^W/_{m^2}]$$

5.1.3 Ecuación de calor para DAMIC 100

La ecuación diferencial del calor en estado transitorio está dada por:

$$\nabla \cdot [-k\nabla(\mathrm{T})] + Q = \rho c_p \frac{\partial T}{\partial t}$$

donde: *Q* es la generación de calor por unidad de volumen por unidad de tiempo; ρ es la densidad; c_p es el calor específico; y t es el tiempo. La ecuación abarca todos los puntos en el dominio de interés Ω.

Se considera para este análisis que en el experimento no hay generación de calor además de que el sistema se encuentra en estado estacionario, esto quiere decir que la temperatura no varía en el tiempo. Por lo tanto la ecuación de calor para DAMIC 100 se reduce a lo siguiente:

$$\frac{\partial}{\partial x}\left(-k\frac{\partial T}{\partial x}\right) + \frac{\partial}{\partial y}\left(-k\frac{\partial T}{\partial y}\right) + \frac{\partial}{\partial z}\left(-k\frac{\partial T}{\partial z}\right) = 0$$

5.1.4 Formulación débil de la ecuación de calor

La formulación débil consiste en escribir ecuaciones diferenciales en forma de ecuaciones integrales. La formulación débil de la ecuación diferencial a ser

resueltas por el Método de Elementos Finitos es construida considerando los siguiente 4 pasos [28]:

1. Multiplicar la ecuación diferencial por una función arbitraria, la cual contrae las ecuaciones a una ecuación escalar.
2. Integrar el resultado del paso 1 sobre el dominio de consideración.
3. Integrar utilizando una de las fórmulas del teorema de Green para reducir el orden de las derivadas (integración por partes).
4. Sustituir las condiciones de frontera.

A continuación se dará un resumen de todo el proceso a seguir para obtener la formulación débil de la ecuación de calor, tomado de la versión 8.4 del Manual Teórico "*FEAP - - A Finite Element Analysis Program*", de Robert L. Taylor de la Universidad de Berkeley, California [28].

En el paso 1, se multiplica la ecuación general de calor por una función arbitraria que llamaremos $W(x)$, la cual transforma la ecuación diferencial en una función escalar. Entonces,

$$W(x)(\nabla \cdot \vec{q}) = 0$$

En el paso 2, se integra sobre el dominio Ω,

$$\int_{\Omega} W(x)(\nabla \cdot \vec{q})\, d\Omega = 0$$

En el paso 3, integramos por partes utilizando una de las fórmulas de Green, dada por:

$$\int_{\Omega} \phi\, d\Omega = \int_{\Gamma} \phi n\, d\Gamma$$

donde ϕ es el producto de dos funciones $[W(x)(\nabla \cdot q)]$, Γ es la frontera del flujo de calor y n son los cosenos directores del vector normal a esta frontera. Por lo tanto, desarrollando la integración por partes y aplicando el teorema de Green, la ecuación de calor queda de la siguiente manera:

$$\int_{\Gamma} W q_n\, d\Gamma - \int_{\Omega} (\nabla \cdot W)\vec{q}\, d\Omega = 0$$

donde la frontera del flujo de calor Γ, se puede expresar de la siguiente forma:

$$\int_{\Gamma} W q_n\, d\Gamma = \int_{\Gamma_T} W q_n\, d\Gamma + \int_{\Gamma_q} W q_n\, d\Gamma$$

$$\Gamma_T = T_0(x_i) = 140\ K$$

$$\Gamma_q = q_n = \varepsilon\sigma(T^4 - T_\infty^4)$$

Paso 4: Sustituir las condiciones de frontera en la ecuación de calor.

$$\int_{\Gamma_q} W\varepsilon\sigma(T^4 - T_\infty^4)\, d\Gamma + \int_\Omega [\nabla \cdot W][k\nabla(T)] d\Omega = 0$$

De esta manera, sólo tenemos derivadas de primer orden, en comparación con la ecuación original que tiene derivadas de segundo orden.

5.1.5 Algoritmo de Newton-Raphson

Code_Aster resuelve las ecuaciones no lineales automáticamente mediante el uso de un algoritmo basado en el algoritmo de Newton-Raphson. Este método es un método iterativo que nos permite aproximar la solución de una ecuación del tipo $f(x) = 0$. Para aplicar el algoritmo de Newton-Raphson, seguimos los siguientes pasos [21]:

1. Expresamos la ecuación en la forma $f(x_j + \Delta x) = 0$ e identificamos la función f.
2. Calculamos la derivada de dicha función. $\Delta x = -\frac{f(x_j)}{f\prime(x_j)}$
3. Construimos una sucesión de aproximaciones de forma recurrente.

$$x_{j+1} = x_j - \frac{f(x_j)}{f'(x_j)}$$

4. Tomamos una estimación inicial de la solución y sustituimos en la fórmula anterior hasta obtener un resultado que entre en nuestra tolerancia.

$$\left|f(x_{j+1}) - f(x_j)\right| < tolerancia$$

5. El algoritmo de Code_Aster aproxima el resultado a 6 decimales [14], por lo tanto la convergencia en este algoritmo se da cuando el residuo del resultado permanece igual en los 6 primeros decimales.

5.2 Código Code_Aster para la simulación térmica

Para poder escribir el código que dará solución a nuestro problema, es necesario conocer los valores de la conductividad y emisividad de los materiales a utilizar, además de establecer las condiciones de frontera que rigen a nuestro sistema [17].

Para el Cobre puro:

- $k = 450\ [{}^{W}/_{m \cdot K}]$
- $\varepsilon = 0.03$

Para el Silicio puro:

- $k = 600\ [{}^{W}/_{m \cdot K}]$

Para el Kapton:

- $k = 150\ [{}^{W}/_{m \cdot K}]$

Code_Aster no tiene un sistema de unidades pre-establecido, por lo que el usuario puede elegir en que unidades trabajar siempre y cuando mantenga ese sistema de unidades en todo el código, a excepción de algunas funciones como RAYONNEMENT (Radiación en español), que le da a la temperatura las unidades de grados Celsius o Centígrados en automático. Como primer aproximación, propondremos que la temperatura que llegará al dedo frío por el sistema de enfriamiento sea de -133 ºC (140 K) y que la temperatura exterior sea de 20 ºC.

A continuación se dará una breve descripción del código utilizado para la simulación térmica:

```
#Inicio del código#

DEBUT();

#LIRE_MAILLAGE permite recuperar una malla producida por otro
código capaz de crear un archivo con el formato MED
(Modelización e intercambio de datos) o ASTER#

Mesh=LIRE_MAILLAGE(FORMAT='MED',);

#AFFE_MODELE Asigna el tipo de fenómeno a estudiar, sea
mecánico o térmico en 3D o 2D#

Model=AFFE_MODELE(MAILLAGE=Mesh,
```

```
                    AFFE=_F(TOUT='OUI',
                            PHENOMENE='THERMIQUE',
                            MODELISATION='3D',),);

#Declaración del valor de la conductividad térmica para el
cobre en un rango de temperaturas de -273 ºC a 1200 ºC#

LAM_CO=DEFI_FONCTION(
                      NOM_PARA='TEMP',
                      VALE=(-273,450,
                            1200,450,
                           ),);

#Declaración del valor de conductividad térmica para el
silicio en un rango de temperaturas de -273 ºC a 1200 ºC#

LAM_SI=DEFI_FONCTION(
                      NOM_PARA='TEMP',
                      VALE=(-273,600,
                            1200,600,
                           ),);

#Declaración del valor de conductividad térmica para el
kapton en un rango de temperaturas de -273 ºC a 1200 ºC#

LAM_KA=DEFI_FONCTION(
                      NOM_PARA='TEMP',
                      VALE=(-273,150,
                            1200,150,
                           ),);

#Declaración del valor de la densidad multiplicada por el
calor específico para el cobre en un rango de temperaturas de
-273 ºC a 1200 ºC. Ya que el sistema se encuentra en estado
estacionario, el valor de la densidad y el calor específico
se pueden considerar como cero#

RHOCP_CO=DEFI_FONCTION(
                    NOM_PARA='TEMP',
                    VALE=(-273,0,
                          1200,0,
                         ),);

#Declaración del valor de la densidad multiplicada por el
calor específico para el cobre en un rango de temperaturas de
-273 ºC a 1200 ºC. Ya que el sistema se encuentra en estado
estacionario, el valor de la densidad y el calor específico
se pueden considerar como cero#
```

```
RHOCP_SI=DEFI_FONCTION(
                  NOM_PARA='TEMP',
                  VALE=(-273,0,
                        1200,0,
                       ),);
```

#Declaración del valor de la densidad multiplicada por el calor específico para el cobre en un rango de temperaturas de -273 ºC a 1200 ºC. Ya que el sistema se encuentra en estado estacionario, el valor de la densidad y el calor específico se pueden considerar como cero#

```
RHOCP_KA=DEFI_FONCTION(
                  NOM_PARA='TEMP',
                  VALE=(-273,0,
                        1200,0,
                       ),);
```

#Declaración del material cobre con sus respectivas propiedades#

```
Copper=DEFI_MATERIAU(THER_NL=_F(LAMBDA=LAM_CO,
                    RHO_CP=RHOCP_CO),);
```

#Declaración del material silicio con sus respectivas propiedades#

```
Silicon=DEFI_MATERIAU(THER_NL=_F(LAMBDA=LAM_SI,
                    RHO_CP=RHOCP_SI),);
```

#Declaración del material kapton con sus respectivas propiedades#

```
Kapton=DEFI_MATERIAU(THER_NL=_F(LAMBDA=LAM_KA,
                    RHO_CP=RHOCP_KA),);
```

#AFFE_MATERIAU asigna el material, ya sea a un grupo de mallas o a todos los elementos. El grupo MAT002 corresponde a todas las piezas que son de silicio. El grupo MAT003 corresponde a los cables cubiertos de kapton. El resto de las piezas serán de cobre#

```
ChamMatr=AFFE_MATERIAU(MAILLAGE=Mesh,
                       AFFE=(_F(TOUT='OUI',
                                MATER=Copper,),
                             _F(GROUP_MA='MAT002',
                                MATER=Silicon,),
                             _F(GROUP_MA='MAT003',
                                MATER=Kapton,),),);
```

#AFFE_CHAR_THER asigna las condiciones de frontera. El grupo de mallas FACD001 corresponde a las dos caras que están en contacto con el sistema de enfriamiento. Estas caras tendrán una temperatura de -133 ºC (140 K). La radiación se asignará a las 6 caras de la caja de cobre que cubren a los 18 CCDs, estas caras corresponden desde el grupo FACD002 hasta el grupo FACD007. La temperatura exterior será de 20 ºC#

```
CharTher=AFFE_CHAR_THER(MODELE=Model,
                        TEMP_IMPO=_F(GROUP_MA='FACD001',
                                     TEMP=-133),
                       RAYONNEMENT=(_F(GROUP_MA='FACD002',
                                     SIGMA=5.67e-8,
                                     EPSILON=0.03,
                                     TEMP_EXT=20,),
                                    _F(GROUP_MA='FACD003',
                                     SIGMA=5.67e-8,
                                     EPSILON=0.03,
                                     TEMP_EXT=20,),
                                    _F(GROUP_MA='FACD004',
                                     SIGMA=5.67e-8,
                                     EPSILON=0.03,
                                     TEMP_EXT=20,),
                                    _F(GROUP_MA='FACD005',
                                     SIGMA=5.67e-8,
                                     EPSILON=0.5,
                                     TEMP_EXT=20,),
                                    _F(GROUP_MA='FACD006',
                                     SIGMA=5.67e-8,
                                     EPSILON=0.03,
                                     TEMP_EXT=20,),
                                    _F(GROUP_MA='FACD007',
                                     SIGMA=5.67e-8,
                                     EPSILON=0.03,
                                     TEMP_EXT=20,),),);
```

#Definición del número de iteraciones a realizar para el método de Newton-Raphson. Como primer simulación, realizaremos 100,000 iteraciones con un paso de 2,500 iteraciones#

```
list_t = DEFI_LIST_REEL(DEBUT = 0., INTERVALLE = _F (JUSQU_A
= 100000., PAS = 2500.,),);
```

#THER_NON_LINE resuelve fenómenos térmicos no lineales asignando todas las características mencionadas previamente al modelo en estudio#

```
T_KELVIN = THER_NON_LINE(MODELE = Model, CHAM_MATER =
ChamMatr, EXCIT = _F(CHARGE = CharTher), ETAT_INIT = _F (VALE
= 0.0), INCREMENT = _F (LIST_INST = list_t,),);

#Impresión del resultado final con el nombre T_KELVIN#

IMPR_RESU(FORMAT='MED',
          RESU=_F(MAILLAGE=Mesh,
                  RESULTAT=T_KELVIN,),);

#FIN DEL CÓDIGO#

FIN();
```

6. Resultados

6.1 Convergencia numérica

Una vez ejecutado el código mencionado anteriormente y graficando los datos obtenidos, se pudo observar que la convergencia numérica para la solución de nuestra ecuación de calor se alcanza por debajo de las 1,000 iteraciones, cuando la primer suposición fueron 100,000 iteraciones. Reduciendo el rango máximo de iteraciones en la gráfica a 500, se puede ver que la convergencia numérica se alcanza aproximadamente a partir de la iteración 250 como se muestra en la Figura 6.1 a):

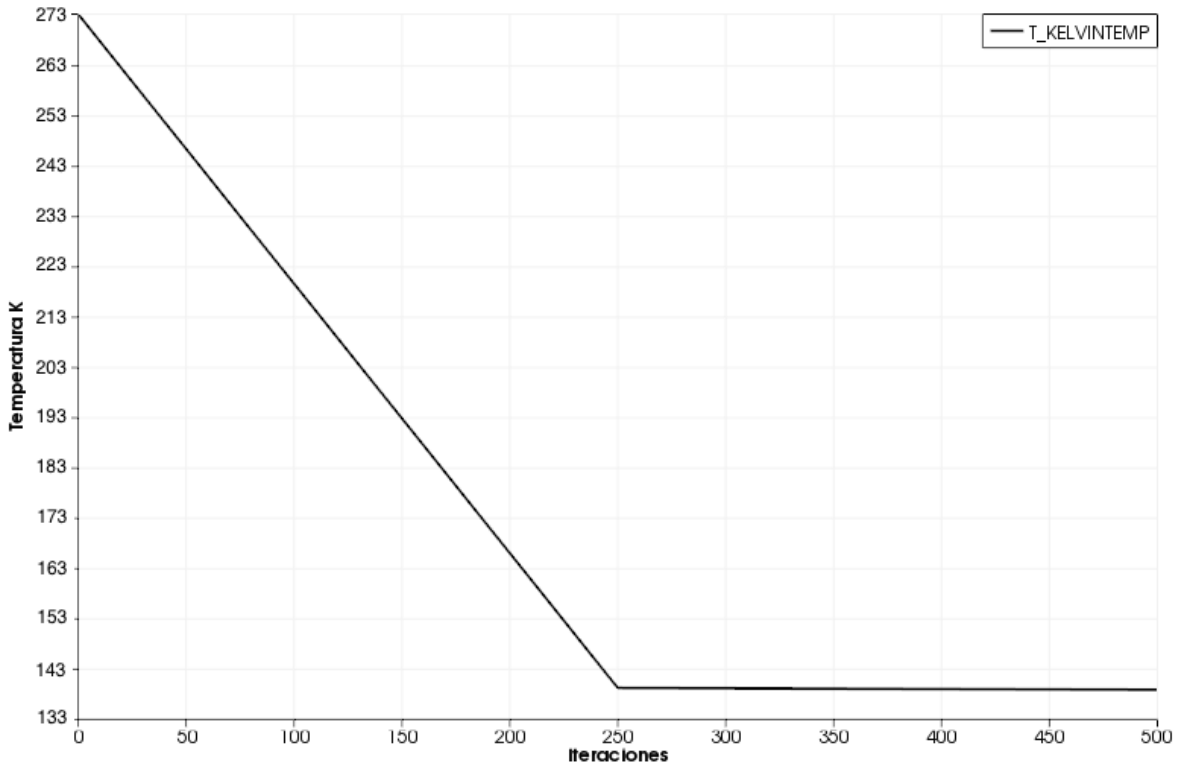


Figura 6.1: Resultados de la primer simulación térmica. a): Gráfica de la convergencia numérica de los datos en la primer simulación térmica.

En la Figura 6.1 b) podemos observar el gradiente de temperatura obtenido de todo el sistema. Los sensores CCD se encontrarían a ~141 K, por lo que será necesario realizar otra simulación reduciendo el número de iteraciones y la temperatura inicial del dedo frío.

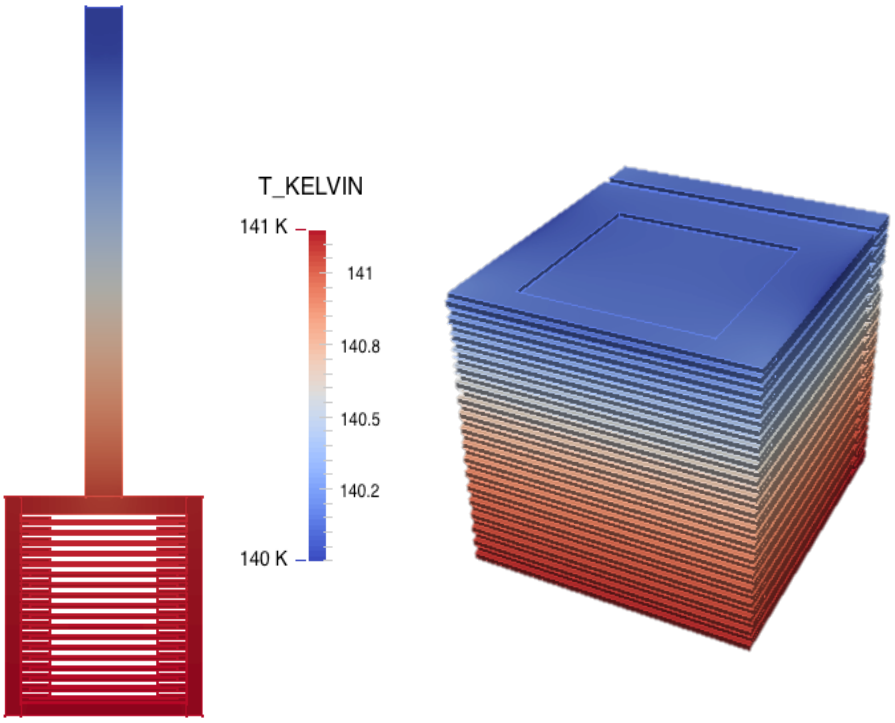


Figura 6.1 b): Gradiente de temperatura resultante de la primer simulación térmica.

6.2 Optimización de la temperatura

Reduciendo la temperatura de condición de frontera del dedo frío a 138 K y reduciendo el número de iteraciones a 500 con un paso de 20 iteraciones, se obtiene el siguiente resultado:

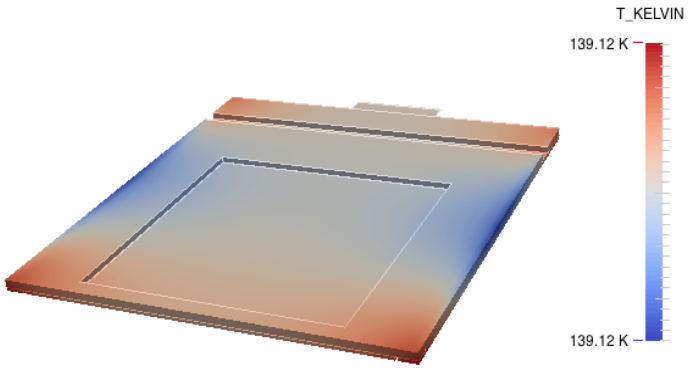


Figura 6.2: Resultado final en el arreglo CCD 1.

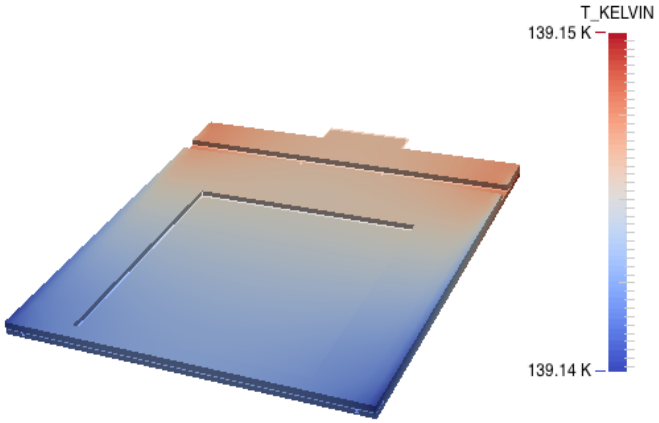


Figura 6.3: Resultado final en el arreglo CCD 18.

En las Figuras 6.2 y 6.3 podemos observar que la temperatura a la que se encuentran los 18 CCDs es en un rango de temperatura de 139.12 K – 139.15 K. Si los contactos térmicos entre todas las piezas del detector son perfectos, la temperatura de operación de los sensores CCD estará dentro del rango admisible. Sin embargo, como trabajo futuro, es necesario la realización de un análisis más detallado que considere contactos térmicos reales, por lo cual son necesarios los resultados experimentales.

6.3 Dependencia de las propiedades mecánicas de los materiales en el resultado final

Para poder observar la influencia de los parámetros utilizados en este análisis térmico, como la conductividad térmica y emisividad, se realizarán simulaciones térmicas con valores para el cobre mayores y menores de un 20% al valor original de la conductividad y se utilizarán valores de emisividad de 0.01 y 0.06. Los resultados obtenidos se muestran a continuación:

$$\%\, Desviación = \frac{Valor\ real - Valor\ calculado}{Valor\ real} \cdot 100$$

Temperatura real: 139.1 [K].

Tabla 1: Desviaciones de temperatura obtenidos por la variación de la conductividad térmica.

Variación de conductividad %	Temperatura [K]	Desviación de temperatura %
-20	139.4	0.216

-10	139.3	0.144
-5	139.2	0.072
5	139.1	0
10	139	-0.072
20	139	-0.072

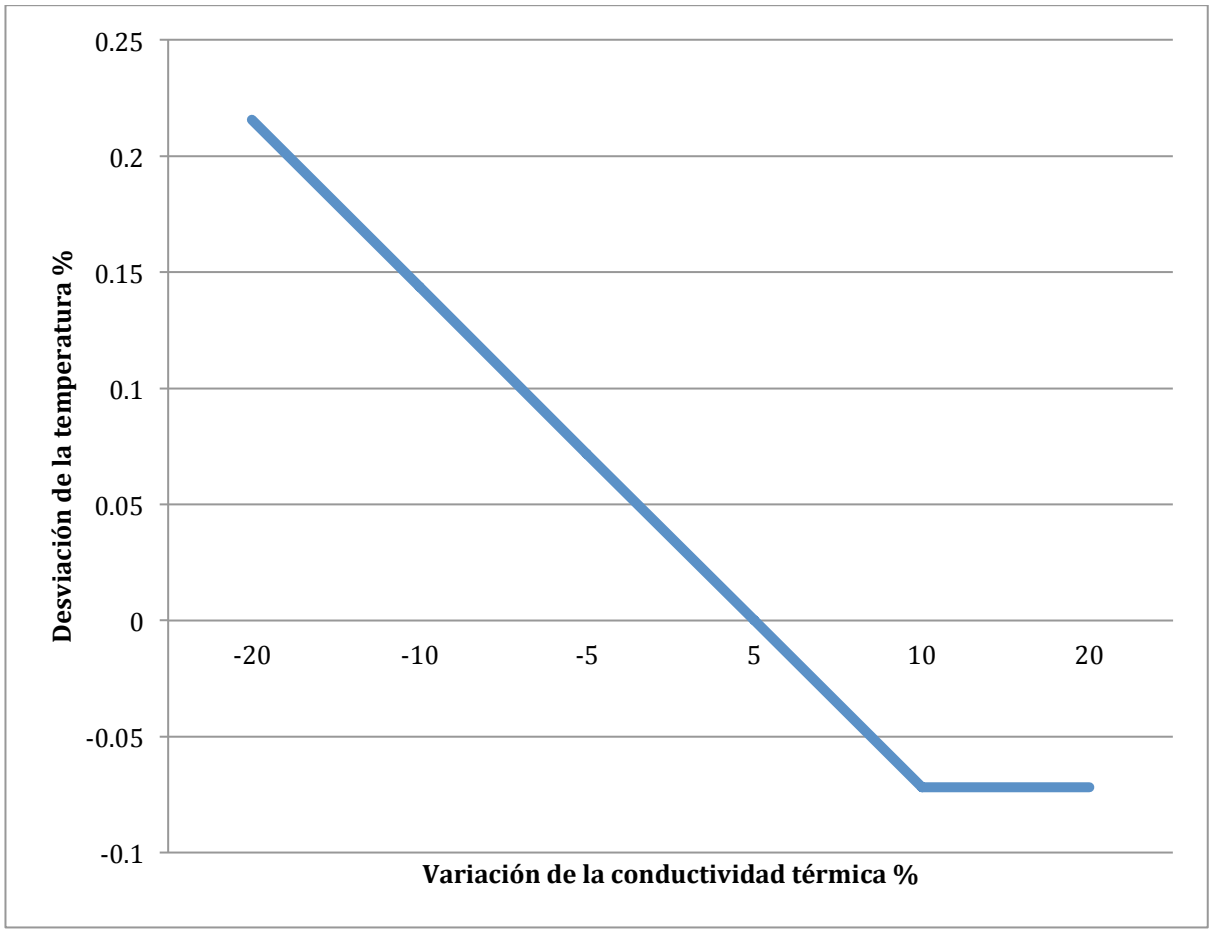


Figura 6.4: Gráfica de las desviaciones de temperatura contra la variación de conductividad térmica.

Tabla 2: Desviaciones de temperatura obtenidos por la variación de emisividad.

Valores de emisividad	Temperatura [K]	Desviación de temperatura %
0.01	138.4	-0.5032
0.06	140.2	0.7908

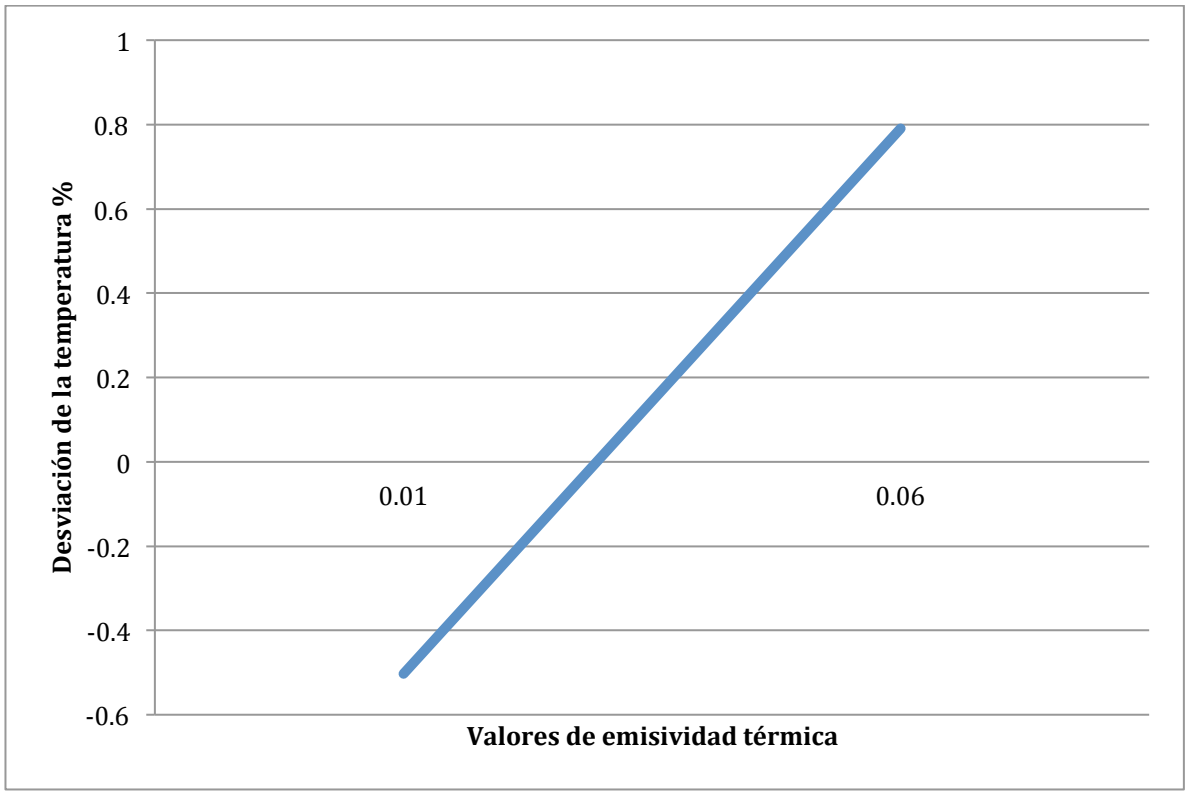


Figura 6.5: Gráfica de las desviaciones de temperatura contra los valores de emisividad.

Analizando la gráfica de la variación de los valores de conductividad térmica para el cobre, podemos observar que aumentando o disminuyendo un 20% su valor, se obtiene un error máximo del 0.21%. Esto quiere decir que si el valor de conductividad del cobre varía dentro de ese rango, el resultado final no tendría un impacto significativo.

En cuanto a la gráfica de la variación en los valores de emisividad térmica del cobre, podemos observar que, aún cuando se aumenta al doble el valor de la emisividad del cobre, el error en la temperatura final es de tan sólo 0.7%. También podemos observar que debido a la baja emisividad térmica del cobre, no hay mucha pérdida de energía. Si la emisividad fuera de 0.5, la pérdida de energía sería mucho mayor, lo que representaría que se tendría que disminuir la temperatura del crio-refrigerador.

En este experimento se utiliza el cobre como material debido a que es el material con las mejores propiedades térmicas. Además, debido a su alta pureza, no generará demasiado ruido a las lecturas de los sensores.

7. Conclusiones

En este trabajo de tesis, la principal aportación fue la implementación de las nuevas geometrías propuestas. Se modificó el código Python realizado por el Dr. Frederic Trillaud para DAMIC, se implementó en las nuevas geometrías de DAMIC 100 para lograr su modelado y se logró realizar la simulación térmica con Code_Aster.

Con el fin de facilitar el modelado, se modificó el diseño original realizado por Greg Derylo de Fermilab y se propusieron simplificaciones de geometrías complejas, respetando a las geometrías relevantes para la simulación térmica y verificando las mejores condiciones para establecer contactos térmicos ideales y de esta manera, garantizar el correcto funcionamiento del sistema. La herramienta utilizada para modelar y mallar a DAMIC 100 es Salome, la cual es una herramienta excelente para el uso en la ingeniería, ya que es un software que es libre, por lo tanto no tiene costo alguno, no tiene restricciones de uso y entrega resultados de la misma calidad que cualquier otro software comercial.

En cuanto al análisis térmico, se logró desarrollar el código en Code_Aster en lugar de la herramienta Cast3M, herramienta que se utilizó anteriormente en DAMIC. Code_Aster es una herramienta importante que nos ayudó a resolver los problemas relacionados a los fenómenos de conducción y radiación de calor que afectan al experimento, lo cual es de vital importancia para garantizar las condiciones de temperatura que requieren los 18 sensores CCD para su correcto funcionamiento debido a su alta sensibilidad. A pesar de ser un código difícil de desarrollar, en internet existe una extensa gama de manuales para cada uno de los comandos de Code_Aster, lo cual ayuda a determinar qué comando es el que más conviene dependiendo del estudio que quieras realizar.

Cabe recalcar que la herramienta *Eficas* es una herramienta de enorme ayuda para la escritura del código. Es una herramienta que facilita el aprendizaje de Code_Aster a usuarios principiantes, ya que cuenta con una lista completa de todos los comandos disponibles en Code_Aster así como todas las opciones de cada uno de dichos comandos. Otro aspecto importante de *Eficas* es que te ayuda a escribir correctamente el código sin errores de sintaxis; se sabe que todo el código tiene toda la información necesaria cuando todos los cuadros de cada comando están de color verde, como se muestra en la Figura 2.8.

Para poder garantizar que los arreglos de sensores CCDs funcionen correctamente con el menor nivel de ruido posible, deben encontrarse a una temperatura de 140 K, para esto, necesitamos que el dedo frio se encuentre a 138 K, de acuerdo a los resultados obtenidos en el análisis térmico. Por lo tanto, la temperatura a la que debe trabajar el crio-refrigerador es entre 135 K – 138 K, lo cual se puede obtener de una manera sencilla controlando el regulador del crio-

refrigerador. Cabe recalcar que esto se cumpliría sólo si los contactos térmicos son perfectos.

Cabe mencionar que para poder obtener los resultados de esta tesis, se hicieron muchas suposiciones y no se tomaron en cuenta algunos factores. No se toma en cuenta el comportamiento del crio-refrigerador, todo el análisis es hecho a partir de la zona de contacto entre el dedo frío y el crio-refrigerador. Se supone que la radiación es totalmente ortogonal a cada una de las caras de la caja de cobre que encierra a los CCDs. Se considera que los contactos térmicos son idealmente perfectos y no se toman en cuenta la resistencia térmica de cada uno de los materiales utilizados en este experimento

Como futuro trabajo quedará realizar las simulaciones térmicas pero ahora sin las suposiciones tomadas para este trabajo. Se debe estudiar el contacto térmico entre las diferentes piezas del experimento para tener una mejor idea del funcionamiento real de todo el sistema.

APÉNDICE A: Código Python principal

```
###############
### INCLUDE ###
###############

isLatestSalomeWanted = 1

### Salome:

import salome
import GEOM
import SMESH

if isLatestSalomeWanted:

  from salome.smesh import smeshBuilder
  smesh = smeshBuilder.New(salome.myStudy)
  from salome.geom import geomBuilder
  geompy = geomBuilder.New(salome.myStudy)

else:

  import smesh
  import geompy
  import visu_gui
  import VISU
  import StdMeshers
  import NETGENPlugin
  import SALOMEDS
  import iparameters
  import salome_notebook
  from medutilities import sauv2med
  from medutilities import med2sauv

### Python:

import os
import sys
import re
import shutil
import subprocess
import numpy
import scipy
from math import *
from datetime import date


#############
### PATHS ###
#############

basePathDirectoryName = '/home/sanmarces/Documents/CAD/SalomeFiles/'
pathDirectoryNameOfPythonFiles = basePathDirectoryName+'PythonFiles/'
pathDirectoryNameOfStepFiles = basePathDirectoryName+'StepFiles/'
pathDirectoryNameOfBrepFiles = basePathDirectoryName+'BrepFiles/'
```

```
pathDirectoryNameOfIgesFiles = basePathDirectoryName+'IgesFiles/'
pathDirectoryNameOfVtkFiles = basePathDirectoryName+'VtkFiles/'
pathDirectoryNameOfStlFiles = basePathDirectoryName+'StlFiles/'
pathDirectoryNameOfUnvFiles = basePathDirectoryName+'UnvFiles/'
pathDirectoryNameOfCgnsFiles = basePathDirectoryName+'CgnsFiles/'
pathDirectoryNameOfExternalFiles = basePathDirectoryName+'ExternalFiles/'
pathDirectoryNameOfMedFiles = basePathDirectoryName+'MedFiles/'
pathDirectoryNameOfSauvFiles = basePathDirectoryName+'SauvFiles/'
pathDirectoryNameOfHdfFiles = basePathDirectoryName+'HdfFiles/'
pathDirectoryNameOfDataFiles = basePathDirectoryName+'DataFiles/'
os.chdir(pathDirectoryNameOfPythonFiles)
GlobalFileName = 'damic100'


##################
### OCC Viewer ###
##################

### Initialize Salome ###
salome.salome_init(0)

### Initialization of the tree view ###
gg = salome.ImportComponentGUI("GEOM")

### Visual parameters ###
ipar =
iparameters.IParameters(salome.myStudy.GetCommonParameters("Interface
Applicative", 1), True)


### Visual properties ###

### VTKViewer:

ipar.setProperty("AP_ACTIVE_VIEW", "VTKViewer_0_0")
ipar.append("AP_MODULES_LIST", "Mesh")

### OCC viewer:

ipar.setProperty("AP_ACTIVE_VIEW", "OCCViewer_0_0")
ipar.setProperty("AP_ACTIVE_MODULE", "Geometry")

### State of the GUI interface:

ipar.setProperty("AP_SAVEPOINT_NAME", "State: 1")

### Upload salome profile ###

if salome.sg.hasDesktop():
  salome.sg.updateObjBrowser(1)
  iparameters.getSession().restoreVisualState(1)
#############
### FILES ###
#############

### USER DEFINED RUNS ###

### Importing external files ###
```

```
runImporting = 0

### Using tools ###

runTools = 1

### Create the various geometries ###

runGeometries = 1

### Assemble the geometries ###

runAssembly = 1

### VIsualization of the geometries ###

runVisualization = 1

### Meshing the geometries ###

runMeshing = 1


### MAIN ###

print " "
print "### Initialization of parameters ###"
execfile(pathDirectoryNameOfPythonFiles+"parameters.py")
print "### Automatic data ###"
execfile(pathDirectoryNameOfPythonFiles+"automatic.py")

print " "

if runImporting:
  print "### Importing external geometric models ###"
  execfile(pathDirectoryNameOfPythonFiles+"importGeometries.py")
  print " "

if runTools:

  print "### Tools ###"
  execfile(pathDirectoryNameOfPythonFiles+"tools.py")

  print " "

if runGeometries:

  print "### Geometrical models ###"
  for i, value in enumerate(listOfConductiveSolidFileNames):

execfile(pathDirectoryNameOfPythonFiles+"solid"+"{0:03d}".format(i+1)+"_"
+value+".py")
  execfile(pathDirectoryNameOfPythonFiles+"copiedGeometries.py")
  for i, value in enumerate(listOfConductiveSolidAdditionalFileNames):
execfile(pathDirectoryNameOfPythonFiles+"solid"+"{0:03d}".format(i+initia
lCountForAdditionalSolids)+"_"+value+".py")
```

```
  print " "

if runAssembly:

  execfile(pathDirectoryNameOfPythonFiles+"assembly.py")  #  Only  if  more
than one geometry
execfile(pathDirectoryNameOfPythonFiles+"properties.py")

  print " "

if runVisualization:
  print "### Geometrical groups ###"
  execfile(pathDirectoryNameOfPythonFiles+"geometricGroups.py")
  print "### Visualization ###"
  execfile(pathDirectoryNameOfPythonFiles+"printing.py")
  print "### Exporting Geometrical models ###"
  execfile(pathDirectoryNameOfPythonFiles+"exportGeometries.py")
  print " "

if runMeshing:

  print "### Meshes ###"
  execfile(pathDirectoryNameOfPythonFiles+"meshes.py")
  print "### Mesh groups ###"
  execfile(pathDirectoryNameOfPythonFiles+"meshGroups.py")
  print "### Mesh Separation ###"
  execfile(pathDirectoryNameOfPythonFiles+"meshSeparation.py")
  print "### Exporting mesh files ###"
  execfile(pathDirectoryNameOfPythonFiles+"exportMeshes.py")
  print " "

############
### SAVE ###
############

currentStudy = salome.myStudy
salome.myStudyManager.SaveAs(pathDirectoryNameOfHdfFiles+GlobalFileName+'
.hdf', currentStudy, False)
geompy.init_geom(currentStudy)

pass
```

Bibliografía

[1] Castañeda Vázquez, A., & Hernández Torres, K. P. (2015). Herramienta gráfica para la automatización de la caracterización de sensores CCD de uso científico. *Tesis de Ingeniería Eléctrica y Electrónica* . Facultad de Ingeniería, UNAM, D.F., México.

[2] Chavarria, Alvaro E. et al. (2015). Damic at Snolab. *Physics Procedia, Volume 61, p. 21-33* .

[3] 3D hp-Finite Elements: Fast Solvers and Adaptivity. (s.f.). *NETGEN - automatic mesh generator*. Obtenido de hpFEM: http://www.hpfem.jku.at/netgen/

[4] Aprile, E. et al. (XENON100 Collaboration). (30 de Julio de 2013). Response of the XENON100 dark matter detector to nuclear recoils. *Phys. Rev. D 88, 012006* .

[5] Aubry, J.-P. (2013). *Beginning with Code_Aster.* (F. (. Framabook), Ed.)

[6] Barbosa de Souza, E. et. al. (24 de FEBRERO de 2016). First Search for a Dark Matter Annual Modulation Signal with NaI(TI) in the Southern Hemisphere by DM-Ice17 - DM-Ice Collaboration. *Phys.Rev.Lett. arXiv: 1602.05939 [physics.ins-det]* .

[7] EDF RECHERCHE ET DÉVELOPPEMENT. (s.f.). *Code_Aster, Salome-Meca course material*. Recuperado el 24 de Julio de 2015, de code-aster.org: http://www.code-aster.org/V2/UPLOAD/DOC/Formations/02-salome.pdf

[8] EDF RECHERCHE ET DÉVELOPPEMENT. (s.f.). *Analysis of Structures and Thermomechanics for Studies & Research*. Recuperado el 21 de Agosto de 2015, de Code_Aster: http://researchers.edf.com/fichiers/fckeditor/Commun/Innovation/logiciels/plaquetteasteren.pdf

[9] EDF RECHERCHE ET DÉVELOPPEMENT. (Noviembre de 2011). *SALOME 6 THE OPEN SOURCE INTEGRATION PLATFORM FOR NUMERICAL SIMULATION*. Recuperado el 22 de Julio de 2015, de CODE-ASTER: http://www.code-aster.org/V2/UPLOAD/DOC/Presentation/SALOME6_brochure-3.pdf

[10] Estrada, J. (18 de Diciembre de 2014). *DAMIC - Direct search for low mass DM with CCDs*. Recuperado el 03 de Julio de 2015, de Facultad de Ingeniería, UNA: http://www.ing.una.py/pdf2014/damic_EOP_estrada.pdf

[11] Fermilab Center for Particle Astrophysics. (s.f.). *Projects: Cryogenic Dark Matter Search (CDMS)*. Recuperado el 08 de Julio de 2015, de Fermilab: http://astro.fnal.gov/projects/DarkMatter/cdms.html

[12] Fermi National Accelerator Laboratory. (Octubre de 2010). *Experiment Profile: DAMIC*. Recuperado el 02 de Julio de 2015, de Fermilab: http://www.fnal.gov/pub/presspass/factsheets/pdfs/experiment_DAMIC.pdf

[13] Freese, K., Lisanti, M., & Savage, C. (01 de Noviembre de 2013). Colloquium: Annual Modulation of Dark Matter: A Review. *Rev. Mod. Phys. 85, 1561* .

[14] Geniaut, S. (2013). *The Councils of use of STAT_NON_LINE.* Recuperado el 14 de Octubre de 2015, de Code_Aster: http://www.code-aster.org/V2/doc/v12/en/man_u/u2/u2.04.01.pdf

[15] Hauschild, M. (27 de Abril de 2009). History of Particle Detectors. 430. WE-HERAEUS-SEMINAR.

[16] Imperial College HEP Research Group. (s.f.). *ZEPLIN-III*. Recuperado el 10 de Julio de 2015, de http://www.hep.ph.ic.ac.uk/ZEPLIN-III-Project/

[17] Incropera. (1990). F. P., & De Witt, D. P. *Introduction to Heat Transfer* (2 ed.). School of Mechanical Engineering, Purdue University: John Wiley & Sons.

[18] Janesick, J. (2001). *Scientific Charge-Coupled Devices.* Washington, Estados Unidos: SPIE Press.

[19] Open Cascade. (s.f.). *Salome Software Suite*. Recuperado el 20 de Julio de 2015 de Open Cascade: http://www.opencascade.com/content/salome-software-suite.

[20] Operator LIRE_MAILLAGE (FORMAT='MED'). (23 de Mayo de 2011). *Code_Aster.* Recuperado el 23 de Octubre de 2015, de

Code_Aster: http://www.code-aster.org/V2/doc/v12/en/man_u/u7/u7.01.21.pdf

[21] Palacios, F. (2008). *Resolución aproximada de ecuaciones: Método de Newton-Raphson.* Escuela Politécnica Superior de Ingeniería de Manresa: Universidad Politécnica de Catalunya.

[22] ParaView. (s.f.). *ParaView Overview*. Recuperado el 24 de Agosto de 2015, de http://www.paraview.org/overview/

[23] Quantum Scientific Imaging, Inc. (2008). *Understanding CCD Read Noise*. Recuperado el 05 de Julio de 2015, de QSI: http://qsimaging.com/ccd_noise.html

[24] Redd, N. T. (01 de Mayo de 2013). *What is Dark Matter?* Recuperado el 02 de Julio de 2015, de SPACE.com: http://www.space.com/20930-dark-matter.html

[25] Salazar Lagunes, C. A. (Abril de 2014). Desarrollo de un sistema de monitoreo y control para el experimento DAMIC. *Tesis de Ingeniería Mecatrónica* . Facultad de Ingeniería, UNAM, D.F, México.

[26] Salome Platform. (s.f.). *Salome Platform Documentation*. Recuperado el 22 de Julio de 2015, de Salome Platform: http://docs.salome-platform.org/7/gui/GEOM/

[27] Salome Platform. (s.f.). *Salome Platform Documentation*. Recuperado el 24 de Julio de 2015, de Salome Platform: http://docs.salome-platform.org/latest/gui/SMESH/index.html

[28] Taylor, R. L. (2011). *A Finite Element Analysis Program* (Version 8.3 Theory Manual ed.). Department of Civil and Enviromental Engineering, University of California at Berkeley, USA.

[29] Trillaud, F. *Código Python DAMIC.* Comunicación Privada.