ESCUELA DE INGENIERÍA INFORMÁTICA

PONTIFICIA UNIVERSIDAD CATÓLICA DE VALPARAÍSO

OPTIMIZACIÓN ROBUSTA EN IMPT

Nicolás Brevis

Profesor Guía: Guillermo Cabrera

Profesor Correferente: Mauricio Moyano

Informe final de Proyecto de Título

Para Optar al Título Profesional de

Ingeniería Civil Informática

Noviembre 2025

Dedico este trabajo
al amor de mi vida Michelle
y a mi querida abuela Carmen
Muchas gracias por apoyarme
en cada paso de esta carrera .

RESUMEN

Este trabajo trata sobre la planificación de tratamientos en la radioterapia de protones, explorando métodos de optimización robusta para enfrentar las incertidumbres a la hora de realizar la entrega de la dosis de energía.

El objetivo principal fue estudiar un programa llamado OpenTPS, el cual es de código abierto y sirve para realizar tratamientos de IMPT. Este software genera una matriz de deposición de dosis (DDM), la cual es una estructura clave para la formulación matemática de los modelos de optimización.

Una vez obtenida la matriz, se utilizará optimización robusta para generar planes de tratamiento.

Palabras clave: Protonterapia; IMPT; Optimización Robusta; OpenTPS; Matriz de Deposición de Dosis (DDM).

ABSTRACT

This work addresses treatment planning in proton therapy, exploring robust optimization methods to mitigate uncertainties during dose delivery.

The primary objective was to study OpenTPS, an open-source software designed for Intensity-Modulated Proton Therapy (IMPT). This software generates a Dose Deposition Matrix (DDM), which is a key structure for the mathematical formulation of optimization models.

Upon obtaining this matrix, robust optimization techniques will be applied to generate treatment plans.

Keywords: Proton Therapy; IMPT; Robust Optimization; OpenTPS; Dose Deposition Matrix (DDM).

ÍNDICE GENERAL

LISTA DE FIGURAS

1. INTRODUCCIÓN

El cáncer es una de las enfermedades más mortales que existe en el mundo moderno, consiste en que algunas células del cuerpo se multiplican sin control y se propagan a distintas partes del cuerpo, lo que genera un tumor que es necesario erradicar. Existen varios caminos para eliminar las células cancerígenas, como la quimioterapia, que es un tratamiento médico que utiliza fármacos que se distribuyen por todo el cuerpo para combatir el cáncer, lo que técnicamente se conoce como un tratamiento sistémico (Gobierno de México, 2023). Otro tratamiento para poder combatir las células cancerígenas es la radioterapia, la cual utiliza radiación ionizante para destruir las células cancerosas o reducir el tamaño de los tumores (instituto nacional del Cáncer (NIH), 2023). Esta técnica se diferencia de la quimioterapia en su forma de aplicarse y funcionamiento: la radioterapia es más focalizada y menos agresiva, mientras que la quimioterapia es más general y puede afectar todo el cuerpo. No existe un tratamiento mejor que el otro; la elección depende de varios factores, como el tipo de cáncer, el estado de la enfermedad y los objetivos del tratamiento.

Es necesario entender cómo funciona la radioterapia de forma más técnica para comprender cuáles son los desafíos a la hora de someterse a este tipo de tratamiento. Cada célula de nuestro cuerpo posee ADN, el cual es utilizado para generar otras células. Es por eso que los tumores no dejan de crecer en la mayoría de los casos, invadiendo tejidos vitales y alterando el funcionamiento normal del cuerpo, lo que finalmente puede provocar la muerte. Los tumores poseen una característica que beneficia al paciente: estas células poseen mecanismos de reparación defectuosos, por lo que no pueden recuperarse luego de una irradiación con fotones (instituto nacional del Cáncer (NIH), 2023). La radioterapia utiliza aceleradores lineales (LINAC) que producen rayos X de alta energía (haz de fotones) que interactúan físicamente con los átomos del cuerpo, lo cual genera una ionización. Esta se produce cuando una partícula transfiere suficiente energía a un átomo o molécula y logra extraerle uno o más electrones. Esto genera electrones libres, los cuales pueden por sí mismos generar más reacciones en las células. Estas interacciones rompen cadenas de ADN del tumor, lo que hace que las células mueran o pierdan su capacidad de multiplicarse.

Aunque la radioterapia pareciera ser una muy buena opción para erradicar el cáncer, tiene un efecto colateral en el cuerpo. Si bien su objetivo es destruir células cancerosas, también afecta tejido sano cercano, ya que este recibe parte de la dosis de radiación. Esto se produce porque los fotones atraviesan todo el cuerpo, por lo que irradian tejido antes, dentro y después del tumor (Gobierno de México, 2023). Aún así, es una de las técnicas más utilizadas y efectivas en la medicina moderna: aproximadamente un 40% de los pacientes curados de cáncer han recibido radioterapia como parte de su tratamiento, y más de un 15% de todas las curaciones se consiguen con tratamientos de radioterapia exclusivamente, como en algunos tumores precoces de laringe o cáncer de próstata (Quirónsalud, 2023).

2. DEFINICIÓN DEL PROBLEMA

El principal desafío a la hora de aplicar IMPT es la presencia de múltiples fuentes de incertidumbre con el movimiento del paciente, como movimientos internos (respiración, latidos del corazón). Estas incertidumbres pueden provocar que el pico de Bragg no se deposite en el tumor, haciendo que el tratamiento no se produzca como se espera (Cao et al., 2012; Neves et al., 2024a). Por esta razón, se han desarrollado enfoques de **optimización robusta**, tanto deterministas (como el método de peor caso) como probabilísticos (simulación Monte Carlo), que permiten generar planes que se mantengan efectivos bajo distintos escenarios posibles (et al., 2022; Kong et al., 2025).

Para abordar estas limitaciones, es muy necesario tener una representación precisa de la forma en la que el haz de protones deposita la dosis en el volumen del paciente. Esta información se encuentra en la **matriz de deposición de dosis** (Dose Deposition Matrix, DDM), la cual es la base matemática para la construcción y optimización de los planes de tratamiento para el paciente. Sin embargo, la creación de la DDM no es tan fácil, ya que depende directamente de las características físicas del haz, las propiedades de los tejidos del paciente y del modelo utilizado.

Por eso, es necesaria una herramienta que permita obtener la matriz y poder utilizarla y manipularla, con la finalidad de luego aplicar técnicas de optimización robusta que consideren explícitamente las fuentes de incertidumbre y entreguen planes más seguros y efectivos (Unkelbach et al., 2009). Actualmente, existen programas comerciales que generan planes para el paciente en base a imágenes **DICOM**, en donde se genera la DDM pero no permiten el acceso directo a esta, sólo la utilizan de manera interna para poder crear el plan; además, no dan flexibilidad para implementar nuevos algoritmos de optimización personalizados (Fredriksson & Bokrantz, 2014).

Frente a esta situación, este trabajo se propone utilizar un software el cual es un sistema de código abierto de planificación y optimización de tratamientos de radioterapia de fotones y radioterapia de protones **OpenTPS** (OpenTPS Project, 2023), el cual, al ser un programa hecho por investigadores para investigadores, permite manipular el nucleo central del programa y acceder a estructuras y procesos internos como la DDM. Además permite trabajar con imágenes médicas en formato DICOM para desarrollar una metodología que permita extraer dicha matriz y utilizarla como basa para los procesos de optimización robusta aplicados a un caso clínico real o simulado. Esto permitirá sentar las bases para futuras investigaciones para la optimización de tratamientos y desarrollo de sistemas de planificación independientes.

3. OBJETIVOS

A continuación, se presentarán los objetivos generales y específicos de este trabajo:

3.1. Objetivo general

Teniendo en cuenta este contexto, este trabajo busca estudiar e investigar un enfoque de optimización robusta para IMPT, considerando explícitamente las incertidumbres de posicionamiento del paciente para generar planes de tratamiento confiables. Esto como base en desarrollar una metodologia computacional basada en el sistema OpenTPS que permita obtener de forma sencilla la matriz de deposición de dosis (DDM) en un formato que sea coherente y ayude a realizar la optimización robusta, a partir de imagenes médicas en fotmato DICOM, para luego generar tratamientos de protonterapia, considerando las ioncertidumbres del proceso y promoviendo una planificación más segura, personalizada y adaptable para la ayuda del paciente.

3.2. Objetivos especíIficos

Para poder lograr el objetivo general de este trabajo, se definen los siguientes objetivos específicos: Primero se plantea como objetivo el comprender en profundidad el funcionamiento del software OpenTPS, leyendo la documentación que se encuentra en línea y asesorandose con los expertos detrás de la creación del mismo. Es necesario prestar atención a su arquitectura modular, flujo de datos y capacidad de acceso a estructuras internas como la matriz de deposición de dosis (Dose Deposition Matrix, DDM). Además se desea explotar por completo los componentes del software relacionados al procesamiento de imagenes CT, scripts de ejemplos que proporcionan y la represenctación computacional del tratamiento.

El objetivo principal de este trabajo es desarrollar un flujo de trabajo sólido que permita obtener de forma precisa y consistente, con un formato entendible, de la Distribución de Dosis Media (DDM) a partir de conjuntos de datos médicos en formato DICOM. Para ello, se tiene en cuenta la carga de una serie de imágenes de tomografía computarizada (CT), las zonas de mayor interes (ROI), la configuración del haz de protones y la realización de los cálculos de deposición de dosis por vóxel utilizando las herramientas de OpenTPS.

Una vez obtenida la DDM, se plantea validarla y analizarla a través de la visualización de matrices, gráficos de dosis y comparaciones entre distintas regiones anatómicas. El propósito de este análisis es comprobar que el modelo físico de deposición utilizado refleja adecuadamente la realidad, y entender cómo varía la distribución de dosis según los parámetros del haz. Estos resultados servirán como base para avanzar hacia la etapa de optimización robusta.

En esa línea, se pretende diseñar una estrategia inicial de optimización robusta basada en la DDM, teniendo en cuenta las principales fuentes de incertidumbre propias de la protonterapia con modulación de intensidad (IMPT), como desplazamientos del paciente o errores en el alcance del haz. La propuesta se inspira en enfoques recientes que utilizan clones estructurales y pseudo-volúmenes para simular escenarios clínicos adversos, con el objetivo de obtener planes de tratamiento que conserven su eficacia incluso ante variaciones geométricas.

Finalmente, se busca documentar cada etapa del proceso de forma clara y replicable, de modo que esta metodología no solo sea útil para futuras investigaciones centradas en la automatización de planes de tratamiento robustos, sino que también sirva como plataforma para incorporar nuevos modelos de optimización en entornos abiertos como OpenTPS.

4. ESTADO DEL ARTE

Dentro del área de la radioterapia como tal, la evolución tecnológica ha permitido desarrollar nuevas técnicas más precisas en la dosis de radiación a los tumores, destacando entre ellas la **Radioterapia de Protones o Protonterapia (IMPT)**. Esta técnica se basa en la utilización de protones para hacer el depósito de radiación, a diferencia de la radioterapia convencional que utiliza fotones para irradiar. Los protones tienen una característica muy particular llamada **pico de Bragg**, que permite depositar la máxima energía justo antes de detenerse en el recorrido, evitando dañar tejidos más allá del tumor (et al., 2022).

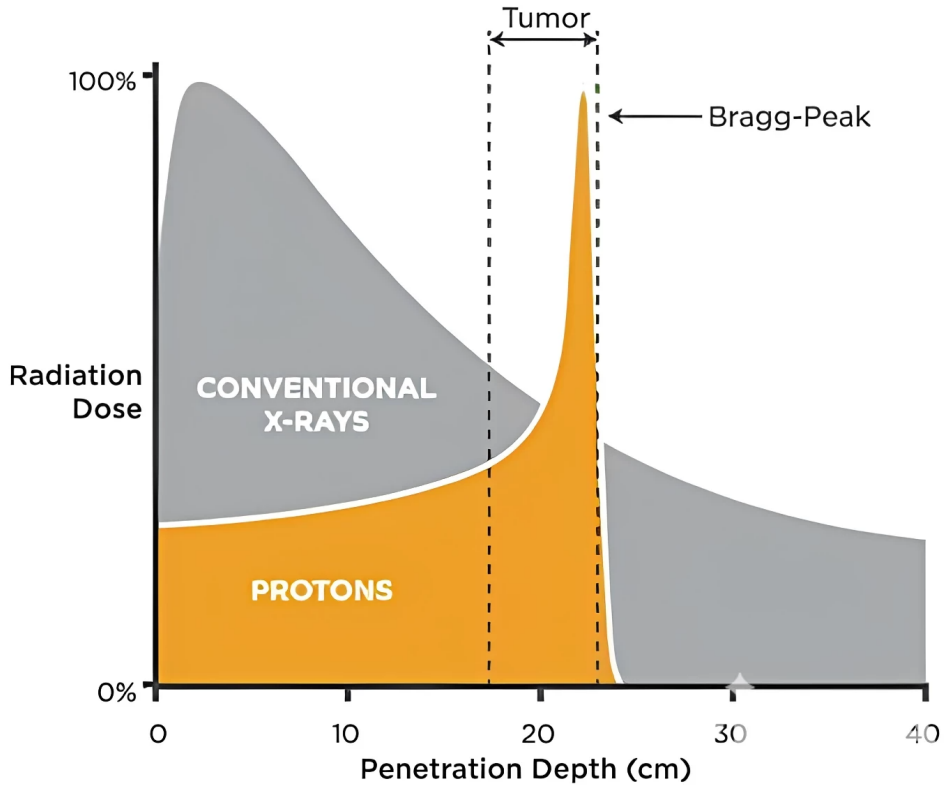


Figure 4.1: Pico de Bragg.(Centro Nacional de Aceleradores (CNA), 2023)

Esta especialidad física otorga al IMPT un mayor potencial para preservar tejidos sanos y reducir efectos secundarios que produce la IMRT (Kong et al., 2025). Sin embargo, esta misma especialidad hace que IMPT sea una técnica altamente sensible a pequeñas variaciones anatómicas, como el posicionamiento del paciente o la estimación del rango de los protones, lo que puede arriesgar la calidad del tratamiento (Cao et al., 2012; Neves et al., 2024a).

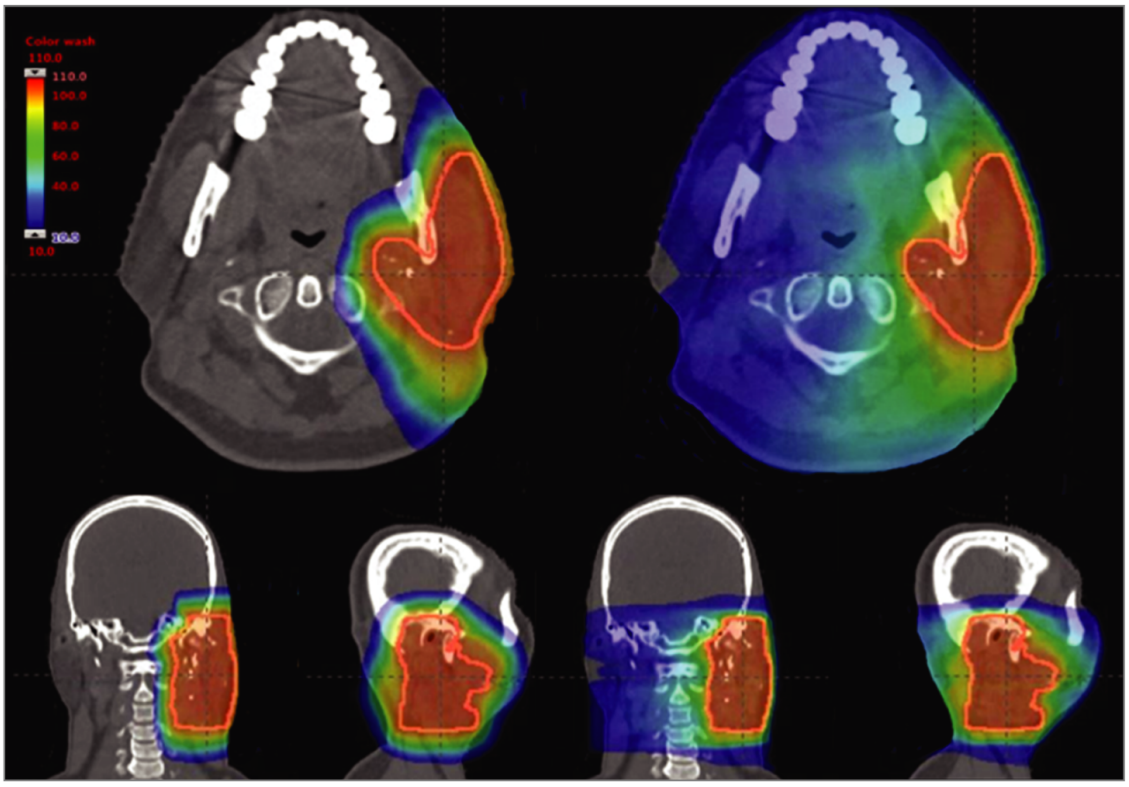


Figure 4.2: IMPT vs IMRT (Giantsoudi et al., 2020)

Estos haces de protones deben irradiarse desde ángulos específicos para lograr que el recorrido sea lo más óptimo posible, por lo que un elemento clave en la planificación de este tratamiento es la selección de los ángulos desde los cuales se dirigirán los haces al tumor. Este problema es conocido como **Beam Angle Optimization (BAO)**. Una mala elección de dicho ángulo puede generar zonas en el tumor que reciban muy poca radiación o una sobreexposición a órganos de riesgo. Además, dado que los ángulos posibles son finitos y combinatoriamente grandes, el BAO representa un problema computacionalmente complejo (Cao et al., 2012; Kong et al., 2025). Sin embargo, la elección de ángulos es una desición que se realiza solamente una vez en la fase inicial de la planificación del tratamiento, y cuya calidad puede mantenerse si se utilizan plantillas robustas y algoritmos automáticos como iCycle-pBAO (Kong et al., 2025).

Por otra parte, la incertidumbre en el posicionamiento del paciente (como movimientos internos, respiración, latidos del corazón) es un problema que afecta en cada fracción del tratamiento, por lo que tiene una mayor probabilidad de ocurrir. Estas incertidumbres pueden provocar que el pico de Bragg no se deposite en el tumor, haciendo que el tratamiento no se produzca como se espera (Cao et al., 2012; Neves et al., 2024a). Además, el impacto que genera la incertidumbre no puede ser resuelto mediante una configuración inicial como lo hace BAO, sino que se requieren estrategias de **optimización robustas** y validarlas bajo múltiples escenarios posibles, por lo que esta problemática es considerada como uno de los principales desafíos clínicos en la planificación del tratamiento con IMPT.

5. METODOLOGÍA DEL TRABAJO

5.1. Comprensión del contexto clínico y técnico

Una de las primeras acciones para comenzar el desarrollo de este trabajo fue necesario comprender y adquirir una comprensión profunda del contexto clínico asociado a la radioterapia, para luego dar paso a la radioterapia con protones, también conocida como **Intensity-Modulated Proton Therapy** (IMPT).

Al estudiar a fondo cómo funcionaba IMPT, se comprendió que una de las principales limitaciones es su elevada sensibilidad a las incertidumbres clínicas, como errores de posicionamiento del paciente, movimientos internos de órganos e incluso variaciones en el rango del haz. Estas incertidumbres pueden hacer que la máxima energía que entrega el haz de protones tenga una desviación significativa entre la dosis planificada y la dosis realmente entregada, afectando la cobertura tumoral y comprometiendo la protección de los órganos adyacentes al haz.

Por lo tanto, es necesario que los planes de tratamiento generados para IMPT sean robustos, lo que significa que deben ser clínicamente aceptables bajo cualquier escenario de error, ya sea geométrico o físico. Por esto, se ha generado la necesidad de desarrollar modelos de optimización robusta para la planificación de tratamientos, los cuales tienen en cuenta los múltiples escenarios de incertidumbre en el proceso de optimización (Unkelbach & Paganetti, 2018; Wahl et al., 2018).

Además, fue completamente necesario familiarizarse con el lenguaje técnico utilizado en el ámbito clínico de la radioterapia. Esta comprensión permitió interpretar correctamente tanto los artículos científicos revisados como los datos estructurales y dosimétricos manipulados en el software de planificación.

5.2. Revisión bibliográfica especializada

La planificación de tratamientos con protones mediante modulación de intensidad (IMPT) tiene beneficios significativos en comparación con la radioterapia convencional de fotones (IMRT), especialmente por su capacidad de entregar dosis de energía en un punto específico gracias al Pico de Bragg. Sin embargo, uno de los principales desafíos de esta técnica es su alta sensibilidad a las

incertidumbres inherentes al proceso clínico. (Mohan & Grosshans, 2017; Unkelbach & Paganetti, 2018).

Por eso, distintas metodologías de *optimización robusta* han sido desarrolladas en los últimos años para mejorar los planes de tratamiento para los pacientes. Estas metodologías se dividen principalmente en dos enfoques: el enfoque probabilístico y el enfoque de peor caso (*worst-case*).

Optimización probabilística. Este enfoque modela las incertidumbres como variables aleatorias con distribuciones conocidas. A partir de múltiples escenarios de tratamiento, se calcula una función objetivo basada en el valor esperado de una métrica dosimétrica sobre todos los escenarios posibles (Wahl et al., 2017, 2018). La formulación típica es:

$$\min_{\mathbf{w}} \; E_s \left[f(\mathbf{D}^{(s)}\mathbf{w}) \right] \tag{5.1}$$

donde $\mathbf{D}^{(s)}$ representa la matriz de deposición de dosis (DDM) en el escenario s, y $\mathbf{w}$ es el vector de intensidades de los *beamlets*. Este tipo de optimización busca una solución que tenga buen desempeño promedio, pero eso no significa que pueda garantizar una cobertura adecuada en escenarios extremos.

Optimización de peor caso (worst-case). Este enfoque busca encontrar el plan que minimice el efecto negativo del peor escenario posible (Fredriksson et al., 2011; Liu et al., 2013). Se define una función objetivo que penaliza la dosis mínima al CTV o la dosis máxima en los OARs bajo las condiciones más adversas:

$$\min_{\mathbf{w}} \; \max_{s \in \mathcal{S}} f(\mathbf{D}^{(s)}\mathbf{w}) \tag{5.2}$$

Esto garantiza que el plan de tratamiento sea clínicamente aceptable incluso en los peores casos simulados. Su gran ventaja es la mejora en robustez, aunque puede generar planes más conservadores.

Formulación cuadrática penalizada. Una de las formulaciones más utilizadas en los 2 enfoques es la basada en funciones objetivo cuadráticas penalizadas por incumplimientos de dosis. En el trabajo de (Neves et al., 2024b), se propone la siguiente función:

$$\min_{\mathbf{w} \geq 0} \sum_{s \in \mathcal{S}} \sum_{i \in s} \left[\lambda_s \cdot \left(L_s - \sum_{j=1}^{N} D_{ij} w_j \right)_+^2 + \left(\sum_{j=1}^{N} D_{ij} w_j - U_s \right)_+^2 \right] \tag{5.3}$$

donde L_s y U_s representan los límites inferior y superior de dosis permitida para la estructura s, λ_s es el peso de penalización, y $[x]_+ = \max(0, x)$ denota la parte positiva. Esta formulación

penaliza tanto la subdosis como la sobredosis, y puede aplicarse en contextos deterministas, probabilísticos o de peor caso.

Representación geométrica de la incertidumbre. En el mismo estudio, los autores introducen dos estrategias automáticas para incorporar la incertidumbre espacial:

- **Estrategia de clones:** Se generan estructuras desplazadas (clones) a partir de cada ROI original, simulando errores de posicionamiento en diferentes direcciones (ejes x, y, z y diagonales). Todas las estructuras clonadas juegan un rol importante en la optimización, dandole validez al plan bajo muchos desplazamientos (Neves et al., 2024b).
- **Estrategia de pseudo-ROI:** Se crea una única estructura ampliada (pseudo-PTV o pseudo-OAR) que resulta de la unión de todos los clones, emulando el enfoque tradicional de márgenes aplicado en clínica.

Ambas estrategias fueron evaluadas por la simulación de Monte Carlo en 100 escenarios con incertidumbres en posicionamiento y rango, confirmando que los planes generados con estructuras clonadas de 6 mm presentaban mayor robustez (menor desviación estándar en D98 del CTV) que los basados en pseudo-ROI o clones de 3 mm.

Comparaciones entre técnicas. Otros trabajos como los de (Zaghian et al., 2017) y (Liu et al., 2012) comparan métodos de programación lineal y no lineal con diferentes formas de representación de la incertidumbre. Si bien los métodos de peor caso son más conservadores, tienden a ofrecer una mayor robustez. Por otro lado, los métodos probabilísticos pueden lograr una mejor cobertura promedio, pero podrían fallar en escenarios extremos si no se modelan correctamente.

5.3. Estudio y exploración del software OpenTPS

Como parte de este trabajo, se llevó a cabo el proceso de estudio del software **OpenTPS** (Open Treatment Planning System), el cual es un sistema de planificación de tratamientos especializado en IMPT, aunque también genera tratamientos para IMRT. OpenTPS es una plataforma de código abierto orientada a la investigación de la radioterapia de fotones y protones. Esta herramienta puede permitir la manipulación de imágenes médicas, estructuras anatómicas, haces de irradiación, y matrices de cálculo de dosis, por lo que resulta especialmente adecuada para el estudio de modelos personalizados de optimización.

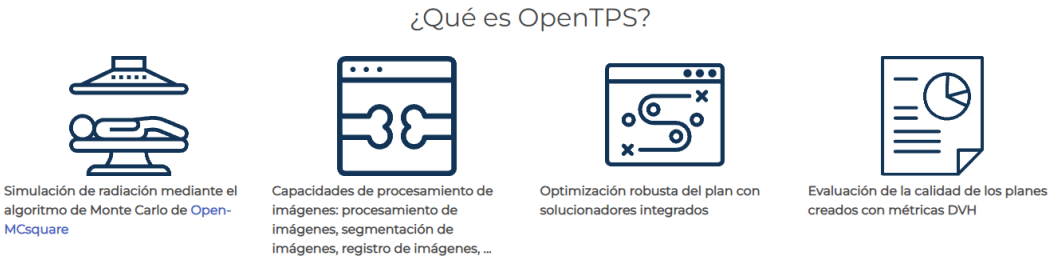


Figure 5.1: Características de OpenTPS

Para poder instalar OpenTPS, es necesario seguir los pasos que están en la página oficial del programa, por lo que se utilizó Anaconda Navigator y Visual Studio Code como herramientas principales. Una vez instalado, se exploró la estructura modular del sistema. OpenTPS organiza su lógica en paquetes como `core.io`, `core.data`, `core.processing` y `core.visualization`.

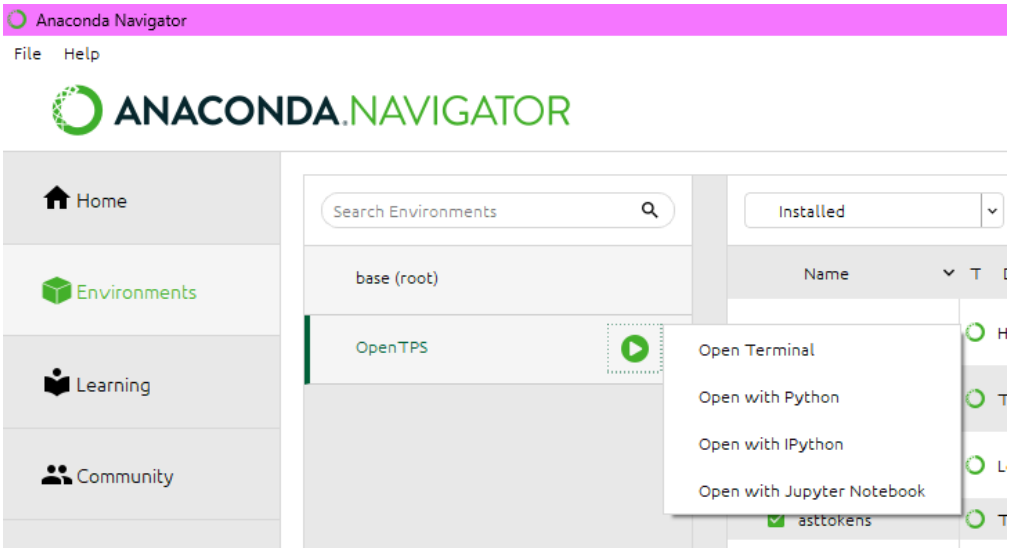


Figure 5.2: OpenTPS instalado

En la documentación oficial de OpenTPS, se destaca que existen distintos módulos clave que componen al programa, como por ejemplo el módulo `dataLoader`, el cual se encarga del proceso de carga y lectura de datos médicos, lo que permite importar imágenes de **tomografía computarizada** (CT) y estructuras anatómicas en formato DICOM RTSTRUCT, representadas internamente como objetos `CTImage` y `ROIMask`, respectivamente.

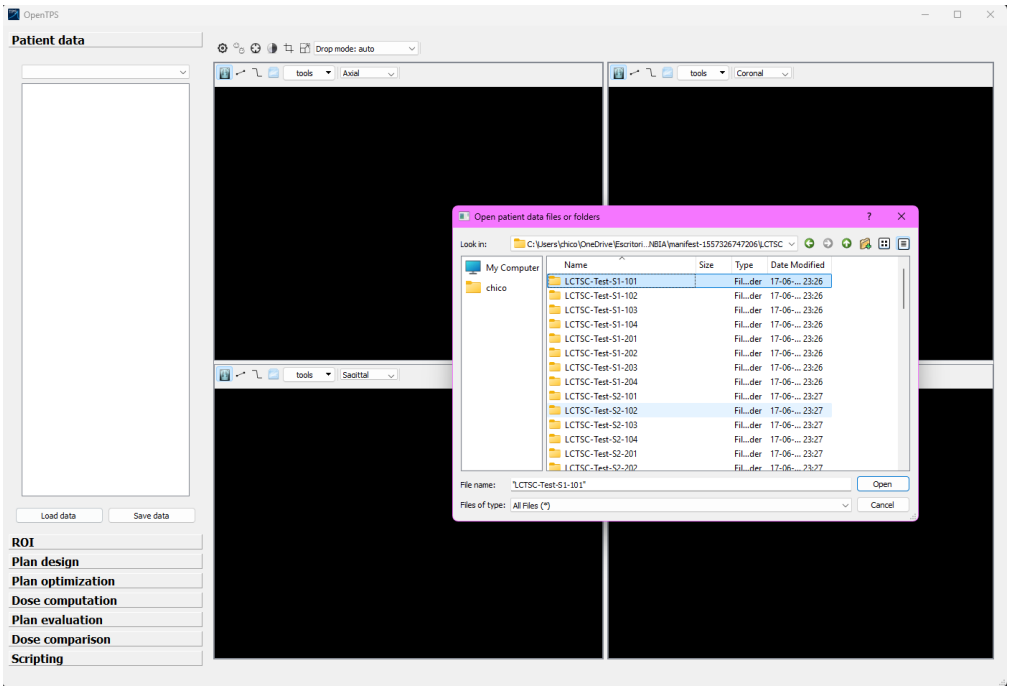


Figure 5.3: Importación de paciente

En la siguiente figura se muesta el modelado 3D que posee OpenTPS para poder tener una mejor visualización del rayo incidente y a la izquierda de muestran las Regiones de Interés (ROI)

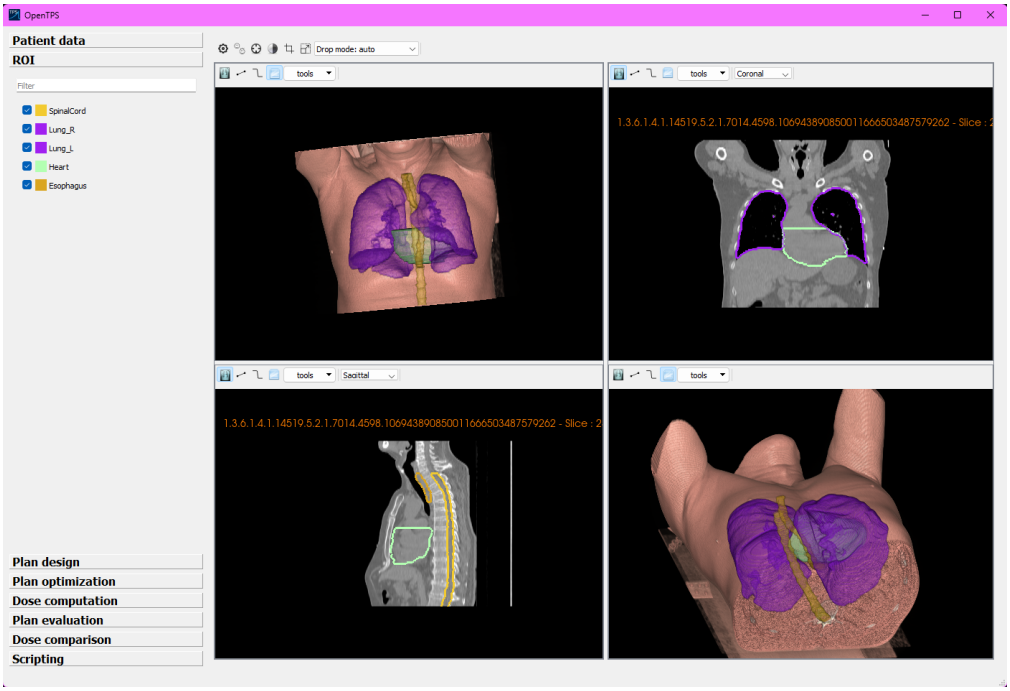


Figure 5.4: Visualizacion de imagenes DICOM con ROI

Luego, los módulos `mcsquareIO` y `doseCalculationConfig` son los encargados de gestionar los archivos para calibrar el escáner, el modelado del haz y los parámetros necesarios para la simulación. Los lanes de tratamiento se hacen gracias a los módulos `ProtonPlanDesign` y `RTPlan`, los cuales permiten definir y organizar los parámetros del tratamiento, definiendo el ángulo y la dosis .

5.3.1. Objetos de Datos Centrales: El Lenguaje de OpenTPS

OpenTPS utiliza una serie de clases de Python para representar las entidades físicas y clínicas del proceso de planificación.

- Representación de Datos del Paciente: Las clases de datos primarias reflejan el estándar DICOM. La clase **CTImage** almacena los datos de la tomografía computarizada, mientras que la clase **RTStruct** contiene el conjunto de estructuras o regiones de interés (ROIs) contorneadas. OpenTPS puede importar estos objetos directamente desde archivos DICOM.
- Representación del Plan de Tratamiento: El objeto **RTPlan** almacena los parámetros de la máquina del tratamiento. Esto incluye los ángulos del gantry y de la camilla, las capas de energía, y las posiciones e intensidades de cada spot de protones.
- Representación de Datos Dosimétricos: La clase **DoseImage** es una matriz 3D que representa la distribución de dosis calculada. Este objeto puede superponerse a la **CTImage** para una evaluación visual.

Esta estructura de clases no es arbitraria; es una abstracción orientada a objetos del flujo de trabajo clínico del mundo real. Un objeto **Patient** puede contener una **CTImage** y un **RTStruct**. Un objeto **PlanDesign** utiliza la **CTImage** y el **RTStruct** para generar un **RTPlan**. Un **DoseCalculator** toma el **RTPlan** y la **CTImage** para producir una **DoseImage**l. Esta relación jerárquica y lógica hace que el código Python sea altamente legible e intuitivo para alguien con experiencia en física médica, permitiendo al investigador pensar en términos de entidades físicas en lugar de matrices de datos de bajo nivel.

5.3.2. Flujo de Trabajo para la Planificación Estándar de un Plan IMPT en la GUI Paso a Paso

1. **Carga de Datos:** En el panel de datos del paciente, se utiliza la opción "Load Data" para seleccionar el directorio que contiene los archivos DICOM. OpenTPS cargará y mostrará automáticamente la TC y los contornos.
2. **Diseño del Plan:** En la pestaña "Plan Design", se pueden añadir haces de forma interactiva. Al seleccionar un haz, se pueden introducir los ángulos de gantry y camilla. En la misma pestaña, se configuran los parámetros de colocación de spots (ROI objetivo, márgenes, espaciado). Al hacer clic en "Design Plan", se genera el plan inicial.
3. **Definición de Objetivos:** Se navega a la pestaña "Objectives". En este panel (mostrado en la Figura 4 de la referencia), se pueden añadir objetivos uno por uno. Para cada objetivo, se selecciona una ROI de una lista desplegable, se elige la métrica (Dmax, Dmin, etc.), se introduce el valor de dosis y se asigna un peso.

4. Optimización y Cálculo de Dosis: En la pestaña "Optimization", se selecciona un solucionador de la lista de opciones disponibles. Al hacer clic en "Optimize Plan", OpenTPS inicia el proceso. Si los beamlets no han sido calculados previamente, el software primero ejecutará el cálculo de la matriz de influencia de dosis (que puede llevar tiempo) y luego procederá con la optimización de los pesos de los spots.
5. **Evaluación:** Una vez finalizada la optimización, la dosis resultante se muestra superpuesta en los visores de imágenes. La GUI tiene una pestaña o herramienta de "Evaluation" dedicada que permite generar y visualizar los HDV para todas las estructuras contorneadas de forma interactiva.

5.3.3. Preparación del Entorno y los Datos

El primer paso en cualquier flujo de trabajo programático es importar las bibliotecas necesarias y cargar los datos del paciente.

- **Importación de Bibliotecas Necesarias:** El script comienza importando las clases esenciales del paquete opentps.core.

```
from opentps.core.data.images import CTImage, DoseImage
from opentps.core.data.patient import Patient
from opentps.core.data.plan import PlanDesign, RTPlan
from opentps.core.io import DICOMIO
from opentps.core.processing.doseCalculation import MCsquareDoseCalculator
from opentps.core.processing.planOptimization import IMPTPlanOptimizer, FidObjective
```

Figure 5.5: Importación de Bibliotecas

- **Carga de Datos DICOM:** Se utiliza la clase DICOMIO para leer un directorio que contiene los archivos DICOM del paciente (imágenes de TC y el archivo RT-STRUCT con los contornos). Este proceso instancia un objeto Patient que contiene automáticamente los objetos CTImage y RTStruct asociados. Los datos de ejemplo suelen estar disponibles en el repositorio del software.

```
# Ruta al directorio con los datos DICOM del paciente
dicom_path = 'path/to/your/dicom_data'

# Cargar los datos y crear un objeto Paciente
patient = Patient(patient_id="TestPatient")
dcm_io = DICOMIO(patient, path=dicom_path)
dcm_io.load_all()

# Acceder a la TC y a las estructuras
ct = patient.get_patient_data_by_type(CTImage)
rtstruct = patient.rtstruct
```

Figure 5.6: Carga de datos

5.3.4. Diseño del Plan de Tratamiento (Objeto PlanDesign)

Una vez cargados los datos del paciente, el siguiente paso es diseñar la geometría del tratamiento.

- **Instanciación de PlanDesign:** Se crea un objeto PlanDesign, que servirá como contenedor para todos los parámetros de diseño del plan. Se le pasan la imagen de TC y el conjunto de estructuras del paciente.

```
plan_design = PlanDesign(ct=ct, rt_struct=rtstruct)
```

Figure 5.7: PlanDesign

- **Definición de la Geometría del Haz:** Se añaden los haces al plan, especificando sus ángulos de gantry y camilla. En este ejemplo, se configura un plan con dos haces opuestos.

```
beam_1_gantry = 90.0
beam_2_gantry = 270.0
couch_angle = 0.0

plan_design.add_beam(gantry_angle=beam_1_gantry, couch_angle=couch_angle)
plan_design.add_beam(gantry_angle=beam_2_gantry, couch_angle=couch_angle)
```

Figure 5.8: Geometría del Haz

- **Configuración de la Colocación de Spots:** Se definen los parámetros que controlan cómo se distribuyen los spots de protones para cubrir el volumen blanco (por ejemplo, 'PTV'). Esto incluye márgenes laterales alrededor del blanco, y el espaciado entre spots y entre capas de energía. Internamente, OpenTPS utiliza un procedimiento de trazado de rayos (ray-tracing) para calcular las longitudes de camino equivalentes en agua (WEPL) mínimas y máximas necesarias para cubrir el volumen blanco, lo que se traduce en el rango de energías requerido

```
plan_design.set_spot_placing(roi_name='PTV', spot_spacing=(3.0, 3.0),
                             layer_spacing=3.0, margin=(5.0, 5.0, 5.0))
```

Figure 5.9: Spots

- **Generación del RTPlan Inicial:** Finalmente, se ejecuta el método design para crear el objeto RTPlan. Este objeto contiene la geometría completa del plan, con las posiciones de todos los spots calculadas, pero con sus intensidades inicializadas a uno.

```
plan_design.design()
initial_rtplan = plan_design.plan
```

Figure 5.10: RTPlan

5.3.5. Generación de la Matriz de Beamlets (El Núcleo Computacional)

Este es el paso más intensivo desde el punto de vista computacional, donde se calcula la matriz de influencia de dosis A.

- **Configuración del Calculador de Dosis:** Se instancia un objeto MCsquareDoseCalculator, especificando el modelo de haz de la máquina y el número de partículas primarias a simular por cada spot. Un mayor número de primarias reduce la incertidumbre estadística del cálculo Monte Carlo a costa de un mayor tiempo de cómputo

```
mc2 = MCsquareDoseCalculator(beam_model_path='path/to/your/beam_model.bdl')
mc2.nb_primaries = 1e5  # Número de protones a simular por spot
```

Figure 5.11: Calculador de Dosis

- **Ejecución del Cálculo:** Se invoca el método para calcular la matriz de beamlets. OpenTPS ejecutará una simulación Monte Carlo para cada uno de los miles de spots en el plan, almacenando la dosis resultante en un formato de matriz dispersa para conservar memoria.

```
plan_design.compute_beamlets(mc2)
beamlet_matrix_A = plan_design.beamlet_matrix
```

Figure 5.12: Cálculo

Este paso representa el principal cuello de botella computacional en el flujo de trabajo de planificación. La necesidad de ejecutar una simulación separada para cada spot significa que cualquier cambio en la geometría del plan (ángulos de haz, márgenes del blanco) requiere un recálculo completo y costoso de la matriz A. Por el contrario, la iteración sobre los objetivos clínicos (descrita en la siguiente sección) es computacionalmente barata, ya que reutiliza la misma matriz A precalculada. Esta comprensión es fundamental para una investigación eficiente, ya que guía al usuario a finalizar la geometría del plan antes de refinar los objetivos dosimétricos.

5.3.6. Formulación de Objetivos Clínicos

En esta fase, las prescripciones clínicas se traducen en objetivos matemáticos que guiarán la optimización.

- **La Clase FidObjective:** Cada objetivo clínico se define utilizando un objeto FidObjective (Objetivo de Fidelidad). Este objeto vincula una meta dosimétrica (por ejemplo, dosis mínima, máxima o media) a una ROI específica.
- **Definición de Objetivos para el Blanco:** Para el volumen blanco ('PTV'), se suelen definir dos objetivos: una dosis mínima para asegurar la cobertura y una dosis máxima para controlar los puntos calientes y mejorar la homogeneidad.

```
# Objetivo de dosis mínima para el PTV (cobertura)
target_min_obj = FidObjective(roi_name='PTV', metric='DoseMin', dose=50.0, weight=100.0)

# Objetivo de dosis máxima para el PTV (homogeneidad)
target_max_obj = FidObjective(roi_name='PTV', metric='DoseMax', dose=52.0, weight=50.0)
```

Figure 5.13: Definición de Objetivos

- **Definición de Objetivos para OARs:** Para los órganos en riesgo, el objetivo es limitar la dosis. Esto se logra comúnmente con un objetivo de DoseMax.

```
# Limitar la dosis máxima en la médula espinal
spinal_cord_max_obj = FidObjective(roi_name='SpinalCord', metric='DoseMax', dose=45.0, weight=500.0)
```

Figure 5.14: OARs

- **Asignación de Pesos:** El parámetro weight es crítico. Determina la importancia relativa de cada objetivo durante la optimización. Un peso más alto en el objetivo de la médula espinal, por ejemplo, le indicará al optimizador que priorice la protección de este órgano, incluso si eso compromete ligeramente la homogeneidad en el blanco. El ajuste de estos pesos es un proceso iterativo clave en la planificación del tratamiento.
- Añadir Objetivos al PlanDesign:

```
plan_design.add_objective(target_min_obj)
plan_design.add_objective(target_max_obj)
plan_design.add_objective(spinal_cord_max_obj)
```

Figure 5.15: Añadir objetivos

5.3.7. Ejecución y Monitorización de la Optimización

Con el plan diseñado, los beamlets calculados y los objetivos definidos, el sistema está listo para optimizar las intensidades de los spots.

- **Instanciación de IMPTPlanOptimizer:** Se crea el objeto optimizador, pasándole el objeto PlanDesign que ahora contiene toda la información necesaria.

```
optimizer = IMPTPlanOptimizer(plan_design=plan_design)
```

Figure 5.16: IMPTPlanOptimizer

- **Selección de un Solucionador:** OpenTPS ofrece varios solucionadores. El algoritmo L-BFGS (Limited-memory Broyden–Fletcher–Goldfarb–Shanno) es una opción robusta y de propósito general para problemas de optimización suaves.

```
optimizer.solver = 'opentps_lbfgs'
```

Figure 5.17: Solucionador

- **Ejecución de la Optimización:** Se invoca el método optimize(). Este algoritmo ajustará iterativamente el vector de pesos de los spots x para minimizar la función objetivo global.

```
result = optimizer.optimize()
```

Figure 5.18: Ejecución

- **Recuperación de Resultados:** El objeto result contiene el plan optimizado (con los pesos de los spots finales) y la distribución de dosis 3D total.

```
optimized_plan = result.plan
optimized_dose = result.dose
```

Figure 5.19: Recuperación

5.4. Búsqueda de la matriz de deposición de dosis (DDM)

Uno de los objetivos principales y más importantes de este trabajo es poder acceder a la **Dose Deposition Matrix** (DDM), la cual representa el aporte de cada **beamlet** de un haz a cada vóxel del volumen del paciente. Para poder realizar las distintas técnicas de optimización robusta, esta matriz es indispensable, ya que se puede realizar una relación lineal entre la intensidad de los haces

de protones y la dosis entregada en cada parte del cuerpo, como se representa en la expresión $\mathbf{d} = \mathbf{D} \cdot \mathbf{w}$, donde $\mathbf{D}$ es la DDM, $\mathbf{w}$ es el vector de pesos de los beamlets, y $\mathbf{d}$ es el vector de dosis resultante.

6. PROPUESTA DE DESARROLLO

El desarrollo de este trabajo se basó principalmente en conocer cómo OpenTPS realizaba la DDM a la hora de crear un plan de tratamiento, con el objetivo de poder utilizarla e integrarla en los modelos de optimización robusta que conocemos, y así poder crear un plan mucho mas específico, cambiando parámetros a nuestra conveniencia. Esta propuesta aborda el problema identificado previamente, la falta de acceso explícito a la DDM en los sistemas comerciales y la necesidad de incorporar incertidumbres clínicas dentro del proceso de planificación.

6.1. Metodología general

Como se dijo en secciones anteriores, la metodología considera ciertas etapas parte desde la adquisición de imágenes médicas hasta la preparación de un modelo matemático de optimización robusta. En primer lugar, se procedió a buscar un dataset público de datos clínicos en formato DICOM, los cuales incluyen imágenes de tomografía computarizada (CT) y estructuras anatómicas definidas mediante archivos RTSTRUCT.

Title	Data Type	Format	Access Points	Subjects	Studies	Series	Images	License	Metadata
Images and Radiation Therapy Structures	RTSTRUCT, CT	DICOM	DOWNLOAD (5.15GB) SEARCH Download requires NBIA Data Retriever	60	60	120	9,593	CC BY 3.0	View

Figure 6.1: Images and Radiation Therapy Structures

Estos datos son importados a través del módulo `dataLoader` de OpenTPS, que los transforma en objetos de tipo `CTImage` y `ROIMask`.

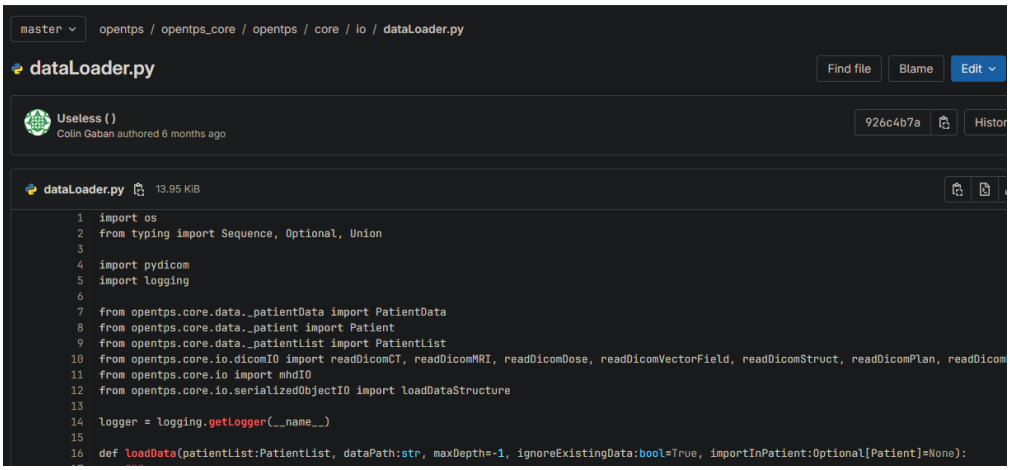


Figure 6.2: DataLoader

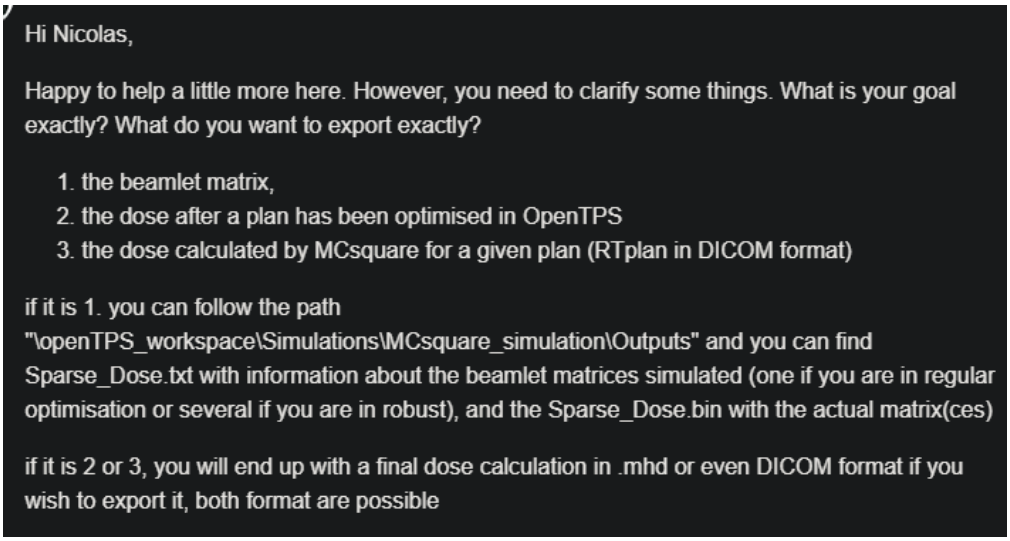
Luego, según la documentación oficial de OpenTPS, se utilizan estructuras como `ProtonPlanDesign` para poder configurar el tratamiento, definiendo los parametros clínicos basicos. En esta etapa, se utiliza el archivo de calibración del escáner, el modelo del haz de protones y los parámetros de simulación necesarios para calcular la dosis depositada.

6.2. Técnica propuesta para acceder a la DDM

Como OpenTPS no expone de forma directa la matriz de deposición de dosis, se exploró el código fuente del módulo `opentos.core`, el cual contiene todos los demás módulos para la creación del plan, en donde también se genera la DDM.

Se consultó directamente en el foro oficial de OpenTPS el cómo se generaba la DDM y cómo se puede obtener, para así tener una fuente confiable y no perder tanto tiempo buscando en el código. En varias discusiones se indica que los cálculos realizados por OpenTPS están basados en modelos de Monte Carlo simplificados y que la DDM se genera en el flujo de cálculo, pero que no se encuentra disponible como un objeto serializable por defecto, por lo que sería necesario modificar el código fuente para exponerla.

Una desarrolladora del programa indicó que la matriz se encontraba en un directorio caché de Opentps, especificando la ruta directa `openTPS_workspace/Simulations/MCsquare_simulation/C` en donde se encontraba un archivo llamado *Sparse_Dose.txt* con información sobre la matriz de beamblets simulada y otro archivo *Sparse_Dose.bin* con la matriz actual.



Hi Nicolas,

Happy to help a little more here. However, you need to clarify some things. What is your goal exactly? What do you want to export exactly?

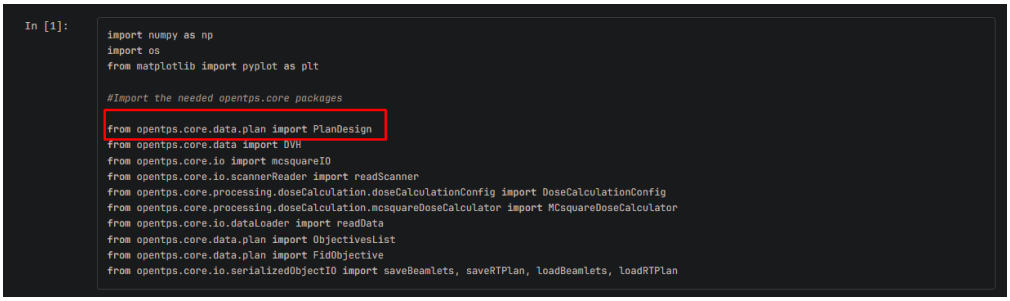
1. the beamlet matrix,
2. the dose after a plan has been optimised in OpenTPS
3. the dose calculated by MCsquare for a given plan (RTplan in DICOM format)

if it is 1. you can follow the path "\openTPS_workspace\Simulations\MCsquare_simulation\Outputs" and you can find Sparse_Dose.txt with information about the beamlet matrices simulated (one if you are in regular optimisation or several if you are in robust), and the Sparse_Dose.bin with the actual matrix(ces)

if it is 2 or 3, you will end up with a final dose calculation in .mhd or even DICOM format if you wish to export it, both format are possible

Figure 6.3: Respuesta OpenTPS

Para poder encontrar esos archivos, primero era necesario generar un plan con calculación de dosis. OpenTPS también cuenta con ejemplos de calculación de dosis para poder conocer más sobre el software como `SimpleRealDoseComupationOptimization.py`—, se observó que muchas de las clases y funciones que deberían estar según la documentación, no se encontraban disponibles en el repositorio principal el cual se instaló para su ejecución, lo que generó errores muy confusos de importación al intentar ejecutar los ejemplos.



In [1]:

```
import numpy as np
import os
from matplotlib import pyplot as plt

#Import the needed opentps.core packages

from opentps.core.data.plan import PlanDesign
from opentps.core.data import DVH
from opentps.core.io import mcsquareIO
from opentps.core.io.scannerReader import readScanner
from opentps.core.processing.doseCalculation.doseCalculationConfig import DoseCalculationConfig
from opentps.core.processing.doseCalculation.mcsquareDoseCalculator import MCsquareDoseCalculator
from opentps.core.io.dataLoader import readData
from opentps.core.data.plan import ObjectivesList
from opentps.core.data.plan import FidObjective
from opentps.core.io.serializedObjectIO import saveBeamlets, saveRTPlan, loadBeamlets, loadRTPlan
```

Figure 6.4: Ejemplo

Como se puede ver en la siguiente figura, en el repositorio que OpenTPS tiene público no existe un módulo llamado PlanDesign.

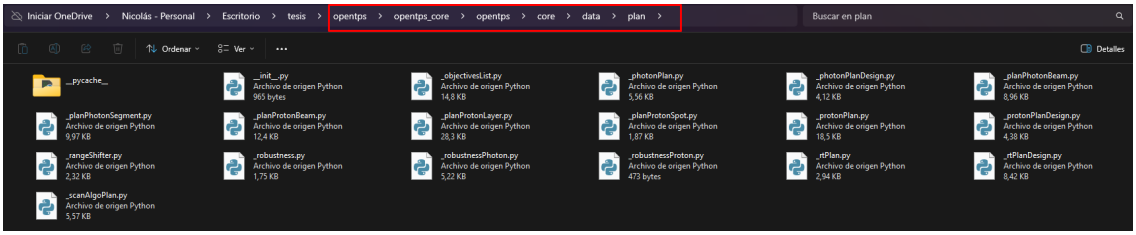
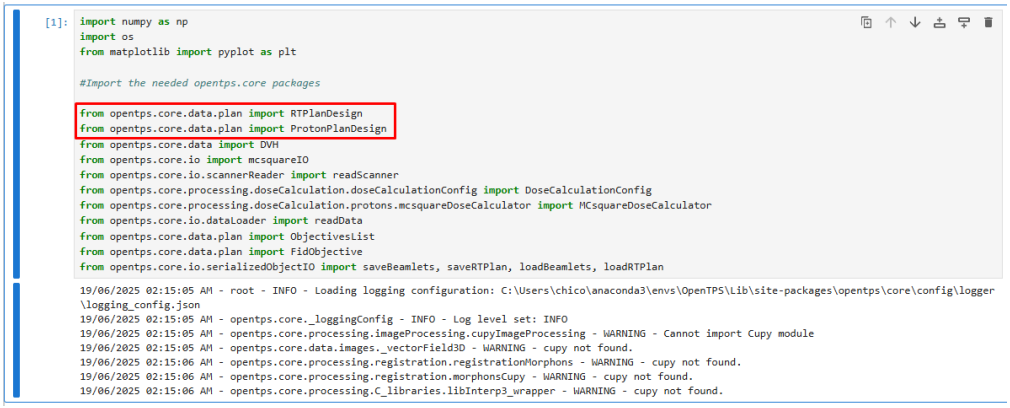


Figure 6.5: Repositorio

Entrando más dentro del código, se encontró con que el archivo habpia cambiado de nombre a *RTPlanDesign.py*, por lo que se cambió en el código para avanzar en el ejemplo



```
[1]: import numpy as np
     import os
     from matplotlib import pyplot as plt

     #Import the needed opentps.core packages

     from opentps.core.data.plan import RTPlanDesign
     from opentps.core.data.plan import ProtonPlanDesign
     from opentps.core.data import DVH
     from opentps.core.io import mcsquareIO
     from opentps.core.io.scannerReader import readScanner
     from opentps.core.processing.doseCalculation.doseCalculationConfig import DoseCalculationConfig
     from opentps.core.processing.doseCalculation.protons.mcsquareDoseCalculator import MCsquareDoseCalculator
     from opentps.core.io.dataLoader import readData
     from opentps.core.data.plan import ObjectivesList
     from opentps.core.data.plan import FidObjective
     from opentps.core.io.serializedObjectIO import saveBeamlets, saveRTPlan, loadBeamlets, loadRTPlan

     19/06/2025 02:15:05 AM - root - INFO - Loading logging configuration: C:\Users\chico\anaconda3\envs\OpenTPS\Lib\site-packages\opentps\core\config\logger
     \logging_config.json
     19/06/2025 02:15:05 AM - opentps.core._loggingConfig - INFO - Log level set: INFO
     19/06/2025 02:15:05 AM - opentps.core.processing.imageProcessing.cupyImageProcessing - WARNING - Cannot import Cupy module
     19/06/2025 02:15:05 AM - opentps.core.data.images._vectorField3D - WARNING - cupy not found.
     19/06/2025 02:15:06 AM - opentps.core.processing.registration.registrationMorphons - WARNING - cupy not found.
     19/06/2025 02:15:06 AM - opentps.core.processing.registration.morphonsCupy - WARNING - cupy not found.
     19/06/2025 02:15:06 AM - opentps.core.processing.C_libraries.libInterp3_wrapper - WARNING - cupy not found.
```

Figure 6.6: Código modificado

Luego, teniendo en cuenta que se deben de cambiar varias llamadas a módulos, se siguió paso a paso el ejemplo, hasta que llegó el momento de diseñar el plan, en donde se percató que la funcion *buildPlan()* estaba vacía.

```
# Design plan
beamNames = ["Beam1","Beam2","Beam3"]
gantryAngles = [0.,45.,315.]
couchAngles = [0.,0.,0.]

# Generate new plan
planDesign = RTPlanDesign()
planDesign.ct = ct
planDesign.gantryAngles = gantryAngles
planDesign.beamNames = beamNames
planDesign.couchAngles = couchAngles
planDesign.calibration = ctCalibration
planDesign.spotSpacing = 5.0
planDesign.layerSpacing = 5.0
planDesign.targetMargin = 5.0
# needs to be called prior to spot placement
planDesign.defineTargetMaskAndPrescription(target = target, targetPrescription = 20.)


plan = ProtonPlanDesign.buildPlan()  # Spot placement
plan.PlanName = "NewPlan"
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
Cell In[10], line 20
     16 # needs to be called prior to spot placement
     17 planDesign.defineTargetMaskAndPrescription(target = target, targetPrescription = 20.)
---> 20 plan = ProtonPlanDesign.buildPlan()  # Spot placement
     21 plan.PlanName = "NewPlan"

TypeError: ProtonPlanDesign.buildPlan() missing 1 required positional argument: 'self'
```

Figure 6.7: Error buildPlan

```
def buildPlan(self):
    """
    Builds a plan from the plan design
    """
    pass

def createBeams(self):
    """
    Creates the beams of the plan

    """
    pass

def initializeBeams(self):
    """
    Initializes the beams of the plan
    """
    pass
```

Figure 6.8: buildPlan vacío

Al momento de redacción de este informe, se ha completado la configuración del entorno, la carga de datos clínicos en OpenTPS y la identificación de los módulos que participan en el cálculo de dosis. Si bien la DDM no ha sido localizada directamente debido a limitaciones en la documentación y a problemas de compatibilidad en el entorno, se ha dejado planteada una estrategia concreta para acceder a ella, ya sea por exportación directa o reconstrucción controlada.

Se espera que una vez superados estos obstáculos, el modelo de optimización robusta pueda ser implementado y evaluado sobre datos reales o sintéticos, como parte del trabajo de continuación o futura investigación.

6.3. Modelo de optimización robusta

Una vez se obtenga obtenida la DDM, se plantea aplicar un modelo de optimización robusta inspirado en el trabajo de (Neves et al., 2024b). El modelo considera múltiples escenarios de incertidumbre, generados mediante desplazamientos geométricos sistemáticos en los volúmenes de interés (clones de estructuras CTV y OARs). La función objetivo penaliza la subdosis en el CTV y la sobredosis en los OARs mediante una formulación cuadrática convexa

6.4. Herramientas y recursos

Para el desarrollo de esta propuesta se consideraron las siguientes herramientas y recursos:

- **OpenTPS**: sistema de planificación de tratamientos con acceso a módulos de cálculo de dosis y manipulación de estructuras clínicas.
- **Python 3.9**: lenguaje base para automatización, análisis de datos y manipulación de matrices.
- **NumPy y SciPy**: bibliotecas utilizadas para implementar la formulación matemática de la optimización.
- **Visual Studio Code + Anaconda**: entorno de desarrollo y administración de dependencias.
- **LCTSC Dataset**: conjunto de datos públicos del Lung CT Segmentation Challenge 2017, publicado por la AAPM y disponible en The Cancer Imaging Archive (TCIA) ((TCIA), 2017), que incluye imágenes CT de planificación y estructuras anatómicas en formato DICOM RTSTRUCT. Este dataset fue utilizado para pruebas de carga de imágenes, estructuras y simulación del flujo de planificación clínica.

7. EXTRACCIÓN Y PROCESAMIENTO DE LA MATRIZ DE DEPOSICIÓN DE DOSIS (DDM)

Dado que la interfaz gráfica de OpenTPS no permite exportar la matriz de influencia (DDM) filtrada por estructuras específicas, se desarrolló un script personalizado en Python para interactuar con el núcleo del software.

7.1. Generación de la DDM Global

Se modificó el codigo fuente de mcsquareDoseCalculator.py para lograr ionterceptar la matriz generada por el simulador de Monte Carlo que utiliza OpenTPS, luego se exporta automáticamente en un formato binario serializado (.blm) antes de su eliminación de la caché.

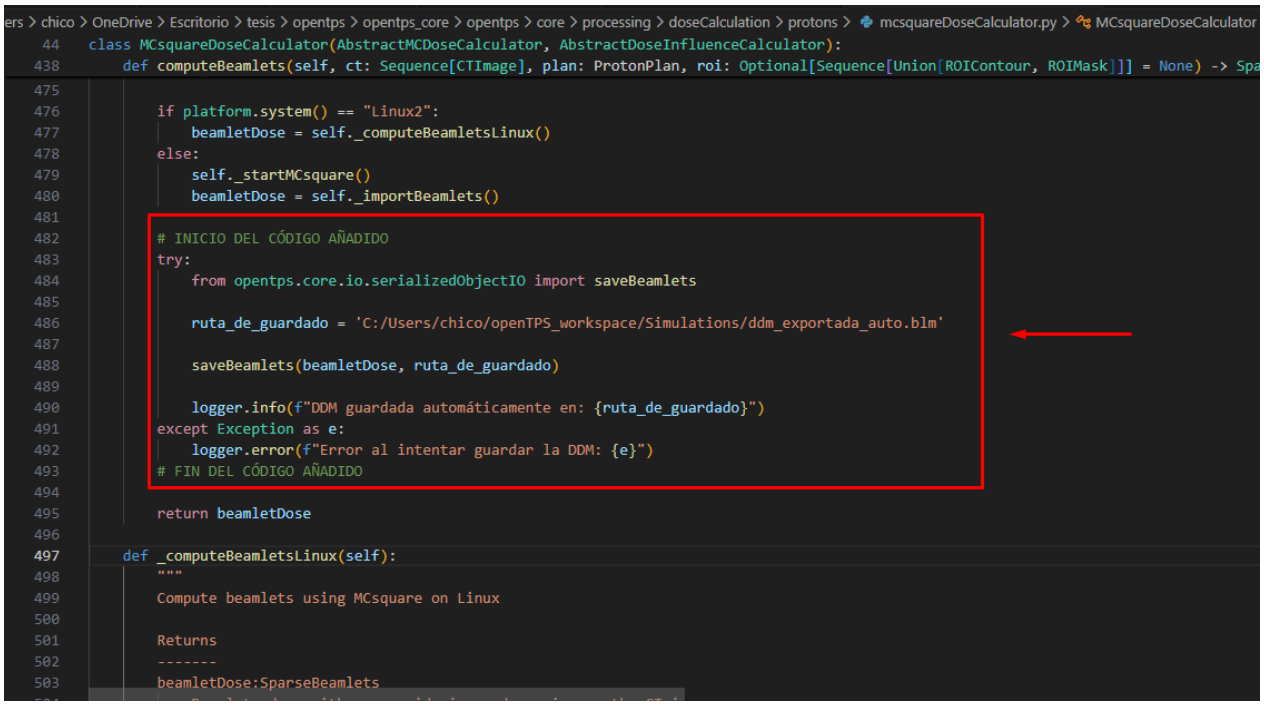


Figure 7.1: Extracción DDM

7.2. Reconstrucción de la Geometría

Junto con el módulo dicomIO, se pudo cargar las imágenes CT y los archivos de estructuras del paciente para visualizarlas mediante la GUI.

7.3. Filtrado y Aplanamiento

La máscara 3D se "aplanó" para generar un vector booleano de longitud N_{voxels}. Este vector se utilizó para filtrar la DDM global ($N_{voxels} \times M_{beamlets}$), extrayendo únicamente las filas correspondientes al volumen del tumor.

El codigo que se desarrolló para filtrar la DDM desde el archivo 'ddm exportada auto.blm' fue:

```
import numpy as np
import scipy.sparse
import os
from opentps.core.io.serializedObjectIO import loadBeamlets

from opentps.core.io.dicomIO import readDicomCT, readDicomStruct
from scipy.sparse import save_npz

print("Librer as importadas correctamente...")
```

```
# Ruta al archivo DDM
ruta_archivo_ddm = 'ddm_exportada_auto.blm'

# Ruta a la CARPETA que contiene los archivos CT
ruta_carpeta_ct = 'bTESTxd'
# Ruta al ARCHIVO .dcm de las estructuras
ruta_archivo_rtstruct = '2.000000-NA-99068/1-1.dcm'

lista_de_archivos_dcm = []

try:
    for nombre_archivo in os.listdir(ruta_carpeta_ct):

        if nombre_archivo.lower().endswith('.dcm'):

            ruta_completa = os.path.join(ruta_carpeta_ct, nombre_archivo)

            lista_de_archivos_dcm.append(ruta_completa)

    # Verificamos si encontramos archivos
    if not lista_de_archivos_dcm:
        print(f"No se encontraron archivos .dcm en: {ruta_carpeta_ct}")
    else:
        print(f"Se encontraron {len(lista_de_archivos_dcm)} archivos .dcm. l

        imagen_ct_resultante = readDicomCT(lista_de_archivos_dcm)

        print(" Proceso  completado exitosamente!")
        # print(imagen_ct_resultante) # Puedes imprimir el resultado

except FileNotFoundError:
    print(f"La carpeta especificada no existe: {ruta_carpeta_ct}")
except NotADirectoryError:
    print(f"La ruta especificada no es una carpeta: {ruta_carpeta_ct}")
except Exception as e:
    print(f"Ocurri  un error inesperado: {e}")

# Cargar los datos
try:
    beamlets_cargados = loadBeamlets(ruta_archivo_ddm)
    matriz_ddm_completa = beamlets_cargados.toSparseMatrix()
    print(f"DDM completa cargada. Dimensiones: {matriz_ddm_completa.shape}")

    # Cargar la imagen CT desde su carpeta
```

```
    ct_image = readDicomCT(lista_de_archivos_dcm)
    print(f"Imagen CT '{ct_image.seriesInstanceUID}' cargada.")

    # Cargar las estructuras (ROIs) desde su archivo
    rt_struct = readDicomStruct(ruta_archivo_rtstruct)
    print(f"Archivo de estructuras cargado.")

except Exception as e:
    print(f"Error cargando archivos: {e}")


# Extraer la ROI del Paciente
if 'rt_struct' in locals():

    nombre_roi_objetivo = 'GTV-1'
    roi_objetivo = None

    for roi in rt_struct.contours:
        if roi.name == nombre_roi_objetivo:
            roi_objetivo = roi
            break

    if roi_objetivo is None:
        print(f"Error: No se encontr  la ROI con el nombre '{nombre_roi_ob
        print("Nombres de ROIs disponibles:")
        for roi in rt_struct.contours:
            print(f"- {roi.name}")
    else:
        print(f"ROI '{roi_objetivo.name}' encontrada.")

        # Implementar la Soluci n de Eliot

        print("Creando m scara 3D usando la CT como plantilla...")
        mascara_3d_objeto = roi_objetivo.getBinaryMask(origin=ct_image.origi
                                                       gridSize=ct_image.gridSize,
                                                       spacing=ct_image.spacing)

        print("Aplanando la m scara...")
        mascara_aplanada = mascara_3d_objeto.imageArray.flatten().astype(bo

        print(f"M scara aplanada creada con {mascara_aplanada.size} elemen

        # Validar Dimensiones y Aplicar el Filtro

        if mascara_aplanada.shape[0] != matriz_ddm_completa.shape[0]:
```

```
        print(f"El tama o de la m scara ({mascara_aplanada.shape[0]})

    else:
        print("Filtrando la DDM principal")
        ddm_roi = matriz_ddm_completa[mascara_aplanada, :]

        print(f"Dimensiones de la DDM extra da para la ROI '{nombre_ro
        # GUARDAR LA MATRIZ EXTRA DA

        ruta_salida_ddm_roi = 'ddm_GTV-1.npz'
        save_npz(ruta_salida_ddm_roi, ddm_roi)

        print(f"\n DDM de la ROI guardada con  xito  en: {ruta_salida_d
```

La ejecución del script de extracción arrojó resultados exitosos, validando la integridad del flujo de datos desde OpenTPS hasta el entorno de Python externo.

```
Objeto de beamlets cargado exitosamente!
<opentps.core.data._sparseBeamlets.SparseBeamlets object at 0x0000022803D93D

Matriz DDM extra da:
  (15350569, 0) 0.032988067716360 09
  (15077192, 0) 0.10006410628557205
  (15076680, 0) 0.01795349456369877
  (15076679, 0) 0.08710777759552002
  (15076167, 0) 0.02229497954249382
  (15076166, 0) 0.016220122575759888
  (14824236, 0) 0.017525549978017807
  (14814022, 0) 0.035117063671350 48
  (14813510, 0) 0.028827639296650887
  (14813509, 0) 0.02759093977510929
  (14813002, 0) 0.02385716885328293
  (14812997, 0) 0.03660918027162552
  (14812996, 0) 0.017894983291625977
  (14812490, 0) 0.007177030202001333
  (14812489, 0) 0.01534968614578247
  (14812484, 0) 0.03973274305462837
  (14812483, 0) 0.008602887392044067
  (14811977, 0) 0.014836743474006653
  (14811976, 0) 0.0038863380905240774
  (14811971, 0) 0.02536487951874733
  (14811464, 0) 0.009122329764068127
  (14549827, 0) 0.013028265908360481
  (14549826, 0) 0.004247927572578192
```

```
  (14549320, 0) 0.008494596928358078
  (14549314, 0) 0.03957909345626831
  :      :
  (14804822, 4588)       0.014273824170231 82
  (14013769, 4588)       0.039109572768211365
  (14013257, 4588)       0.0689416229724884
  (14012745, 4588)       0.04407934471964836
  (14012742, 4588)       0.0844419077038765
  (14012233, 4588)       0.00417380640283227
  (14012230, 4588)       0.0421866700053215
  (14011721, 4588)       0.08552075177431107
  (14011718, 4588)       0.03563966974616051
  (14011209, 4588)       0.03681526705622673
  (14011208, 4588)       0.014376125298440456
  (14011206, 4588)       0.01558396965265274
  (14011205, 4588)       0.013359863311052322
  (14010696, 4588)       0.041200052946805954
  (14010693, 4588)       0.0278193186968565
  (14010181, 4588)       0.02707049995660782
  (14009669, 4588)       0.026517625898122787
  (15835972, 4591)       0.0070866793394088745
  (15835460, 4591)       0.04103884473443031
  (15834948, 4591)       0.03513360023498535
  (13493061, 4592)       0.040896229445934296
  (13492549, 4592)       0.0677938386797905
  (13492037, 4592)       0.004373330622911453
  (13492036, 4592)       0.04292173683643341
  (13491524, 4592)       0.0383729562163353
Dimensiones de la matriz (V xeles x Beamlets): (23855104, 4604 )
```

Este fue el resultado que tenía toda la matriz de deposición de dosis, luego de aplicar el filtrado, las dimensiones se redujeron en un 99.43%

```
Matriz indices: [30298 30030 29941 ... 25584 25512 25467]
Matriz indptr: [     0     93    105 ... 226124 226132 226133]
Matriz format: b'csc'
Matriz shape: [30304  9186]
Matriz data: [8.8190020e-04 1.0555763e-03 1.6270606e-03 ... 7.5561291e-0
```

- Matriz Original (Espacio Total): $23,855,104$ vóxeles (filas) $\times$ $9,186$ beamlets (columnas).
- Matriz Filtrada (Espacio Objetivo): $137,352$ vóxeles $\times$ $9,186$ beamlets.

Esta transformación implica que solo el 0.57% del volumen total simulado corresponde al volumen objetivo (GTV-1). Al aislar estos $137,352$ elementos relevantes, se reduce la carga computacional en más de dos órdenes de magnitud, lo que resulta fundamental para la viabilidad de los algoritmos de optimización robusta subsiguientes.Finalmente, la matriz resultante fue serializada exitosamente en el archivo ddm GTV-1.npz, confirmando que la estructura de datos dispersa se mantuvo intacta durante el proceso.

8. CONCLUSIÓN

Este trabajo comenzó con la idea de entender cómo se planifican tratamientos con protones considerando las incertidumbres del paciente, y cómo herramientas abiertas como OpenTPS pueden ayudar en ese proceso. A lo largo del semestre, ese objetivo fue tomando forma, enfrentando varias dificultades, pero también dando pasos importantes.

Lo primero fue entender bien qué es la radioterapia de protones, qué la hace distinta, y por qué la optimización robusta es necesaria en este tipo de tratamientos. Para eso, se revisó bastante literatura, especialmente papers que proponían modelos matemáticos para enfrentar la incertidumbre. Esos modelos fueron clave para proponer luego una estrategia propia.

Después vino la parte más práctica, que comprendía en trabajar con OpenTPS. Se logró instalar el software, entender su estructura y cargar datos médicos reales en formato DICOM. También se exploraron varios de sus módulos, especialmente aquellos relacionados con el cálculo de dosis. En esa búsqueda se detectó que, aunque la documentación no lo muestra directamente, OpenTPS sí genera internamente la matriz de deposición de dosis (DDM), algo que fue posible comprobar gracias a los archivos generados por el motor MCsquare.

Se logró superar la mayoría de las barreras que OpenTPS tenía, como código desactualizados o falta de documentacion. A travez del desarrollo de scripts personalizados e incrustándolos dentro del corazón del software, se logró obtener la matriz de deposición de dosis (DDM) en un formato matricial (.blm). Esto es indispensable para formular cualquier modelo de optimización matemática del tipo $d = D \cdot w$, lo que hace computacionalmente viable la ejecución futura de algoritmos de optimización robusta, los cuales requieren resolver problemas iterativos complejos que serían inmanejables con la matriz completa

Un aporte significativo fue la implementación de redicción de dimensionalidad de la DDM general. Al filtrarla, se redujo el espacio de busqueda de 23.8 millones de vóxeles a solo 137,000 elementos relevantes.

Ya con la matriz filtrada, el proyecto se encuentra en condiciones inmediatas para iniciar la fase de modelamiento matemático para la optimización y generar un plan acorde a las necesidades que se estimen pertinentes.

Durante la preparación de este trabajo, se ha utilizado ChatGPT (OpenAI) para mejorar la claridad y la redacción del contenido. Después de utilizar esta herramienta, se ha revisado y editado el contenido generado. El autor toma completa responsabilidad por los contenidos en este informe.

BIBLIOGRAPHY

Cao, W., Lim, G., Li, Y., Zhu, X., & Zhang, X. (2012). Uncertainty incorporated beam angle optimization for impt treatment planning. *Medical Physics*, *39*(8), 5248–5258.

Centro Nacional de Aceleradores (CNA). (2023). Experimentos en el cna para la mejora de los tratamientos de protonterapia [Accedido en abril de 2025].

et al., W. C. (2022). Reflections on beam configuration optimization for intensity-modulated proton therapy. *Physics in Medicine and Biology*, *67*, 13TR01.

Fredriksson, A., & Bokrantz, R. (2014). A critical review of optimization approaches for robust impt planning. *Medical physics*, *41*(8), 080901.

Fredriksson, A., Forsgren, A., & Hardemark, B. (2011). Minimax optimization for handling range and setup uncertainties in proton therapy. *Medical Physics*, *38*(3), 1672–1684.

Giantsoudi, D., Grassberger, C., Craft, D., Niemierko, A., Trofimov, A., Paganetti, H., Shusharina, N., & Efstathiou, J. A. (2020). Linear energy transfer–guided optimization in intensity modulated proton therapy reduces high–linear energy transfer radiation to organs at risk. *JAMA Oncology*, *6*(3), 469–476. https://doi.org/10.1001/jamaoncol.2019.5915

Gobierno de México, I. (2023). Diferencias entre quimioterapia y radioterapia [Accedido en abril de 2025].

instituto nacional del Cáncer (NIH). (2023). Radioterapia contra el cáncer [Accedido en abril de 2025].

Kong, W., Huiskes, M., Habraken, S., Astreinidou, E., Rasch, C., Heijmen, B., & Breedveld, S. (2025). 'icycle-pbao': Automated patient-specific beam-angle selection in proton therapy applied to oropharyngeal cancer. *Radiotherapy and Oncology*, *206*, 110799. https://doi.org/10.1016/j.radonc.2025.110799

Liu, W., Frank, S. J., Li, Y., & Mohan, R. (2013). Effectiveness of robust optimization in intensity-modulated proton therapy planning for head and neck cancers. *Medical Physics*, *40*(5), 051711.

Liu, W., Zhang, X., Li, Y., & Mohan, R. (2012). Robust optimization of intensity modulated proton therapy. *Medical Physics*, *39*(2), 1079–1091.

Mohan, R., & Grosshans, D. (2017). Proton therapy—present and future. *Advanced Drug Delivery Reviews*, *109*, 26–44.

Neves, J., Rocha, H., Ferreira, B., & Dias, J. (2024a). Robust optimization for impt: Introducing and comparing different automated approaches. *ICCSA 2024 Workshops, LNCS 14816*, 324–340.

Neves, J., Rocha, H., Ferreira, B., & Dias, J. (2024b). Robust optimization for impt: Introducing and comparing different automated approaches. *International Conference on Computational Science and Its Applications*, 324–340.

OpenTPS Project. (2023). Opentps documentation [Available at https://www.opentps.org].

Quirónsalud. (2023). Tipos de radioterapia: Última tecnología para curar el cáncer [Accedido en abril de 2025].

(TCIA), T. C. I. A. (2017). Lung ct segmentation challenge 2017 (lctsc) [Available at https://www.cancerimagingarchive.net/collection/lctsc/].