P.I.
(043)62
2020
G 586

PROYECTO INTEGRADOR DE INGENIERÍA NUCLEAR

DISEÑO E IMPLEMENTACIÓN DE UN SIMULADOR/PLANIFICADOR DE PRODUCCIÓN DE RADIOISÓTOPOS

Andrea Gómez
Ingeniera

Ing. Daniel Hergenreder
Director

Dr. Eduardo Villarino
Co-director

Miembros del Jurado
Lic. Fernando Sanchez (CNEA)
Mgter Ariel Márquez (CNEA)

26 de Julio de 2020

Departamento de Ingeniería Nuclear - INVAP

Instituto Balseiro
Universidad Nacional de Cuyo
Comisión Nacional de Energía Atómica
Argentina

INVENTARIO 24123
07.06.21
Biblioteca Leo Falicov

A mi familia

Índice de contenidos

Índice de contenidos v

Índice de figuras ix

Índice de tablas xi

Resumen xiii

Abstract xv

1. Motivación 1

2. Introducción 3

2.1. Introducción a los radioisótopos 3

2.2. Aplicaciones de radioisótopos 4

2.2.1. Aplicaciones industriales 4

2.2.2. Aplicaciones científicas 4

2.2.3. Aplicaciones en agricultura 5

2.2.4. Aplicaciones en productos de consumo 5

2.2.5. Aplicaciones en medicina 6

3. Radioisótopos 9

3.1. Molibdeno 99 9

3.1.1. Características generales 9

3.1.2. Usos 9

3.1.3. Producción 10

3.1.4. Mercado 11

3.2. Ir-192 11

3.2.1. Características Generales 11

3.2.2. Irradiación 12

3.3. Lu-177 13

3.3.1. Características generales 13

3.3.2. Usos 13
3.3.3. Vía de producción directa ($^{176}_{71}Lu(n,\gamma)^{177}_{71}Lu$) 14
3.3.4. Vía de producción indirecta ($^{176}_{70}Yb(n,\gamma)^{177}_{70}Yb \xrightarrow{\beta-} {}^{177}_{71}Lu$) 14

4. Modelo **17**
4.1. Reacción (n,γ) 17
4.1.1. Quemado del target y del producto 18
4.1.2. Irradiación seguida de decaimiento $\beta-$ 20
4.2. Reacción de producción por fisión 21
4.2.1. Verificación y Validación 24

5. Planificador **27**
5.1. Rendimiento 27
5.2. Cambio de potencia 28
5.2.1. Efectos de cambios de potencia 29
5.3. Interrupción de la irradiación 30
5.3.1. Efecto de la interrupción de la radiación 30
5.3.2. Reanudación de la irradiación con el mismo flujo anterior 32
5.3.3. Reanudación de la irradiación con un flujo diferente al original . 33
5.3.4. Reanudación de la irradiación con un cambio de flujo posterior . 34
5.4. Algoritmo del planificador 34
5.4.1. Input 34
5.4.2. Cálculo de tiempos de irradiación 35
5.4.3. Output 36
5.4.4. Planificador para Lu^{177} producido por vía indirecta 36
5.4.5. Caso de aplicación 37

6. Simulador **41**
6.1. Descripción general 41
6.2. Descripción y modelado del OPAL 44
6.2.1. Modelado con CITVAP 44
6.3. Cálculo de la actividad por diferencias finitas 45
6.4. Resultados 48

7. Conclusiones **53**
7.1. Planificador 53
7.2. Simulador 53
7.3. Trabajo futuro 54

A. Simulador **55**

B. Planificador Mo^{99} **65**

C. Planificador Lu^{177} **73**

Bibliografía **77**

Índice de figuras

3.1. Esquema de decaimiento del Mo^{99} 10
3.2. Centellograma de perfusión cerebral con Tc^{99m} 11
3.3. Yield de fisión del U-235 12
3.4. Duración típica de las etapas de producción de Mo^{99} 13
3.5. Esquema de decaimiento del Ir^{192} 14
3.6. Esquema de decaimiento del Lu^{177} 14
3.7. Sección eficaz para la reacción $({}^{176}_{71}Lu(n,\gamma){}^{177}_{71}Lu)$ 15
3.8. Sección eficaz de la reacción $({}^{176}_{71}Lu(n,\gamma){}^{177m}_{71}Lu)$ 15
3.9. Sección eficaz de la reacción $({}^{176}_{70}Yb(n,\gamma){}^{177}_{70}Yb)$ 16

4.1. Actividad del Lu^{177} por vía directa ante una interrupción de la irradiación de 1 día, que ocurre en diversos momentos. 19
4.2. Actividad del Lu^{177} producido por vía indirecta en función del tiempo ante irradiación, interrupción de la misma y rerirradiación 21
4.3. Actividad del Mo^{99} comenzando con una irradiación, luego interrumpiéndola, y finalmente retomando la irradiación 23

5.1. Rendimiento del proceso de disolución del Mo^{99} en función del tiempo de irradiación 28
5.2. Actividad del Mo^{99} afectada por el rendimiento de disolución 28
5.3. Actividad del Mo^{99} en función del tiempo ante una reducción de potencia del 10 % 29
5.4. Actividad del Mo^{99} para un aumento y un descenso de pontencia. . . . 30
5.5. Actividad del Mo^{99} ante interrupciones de la irradiación de diversa longitud 31
5.6. Actividad del Mo^{99} ante interrupciones de la irradiación que ocurren en diferentes momentos. 32
5.7. Actividad del Mo^{99} ante una interrupción de la irradiación 33
5.8. Actividad del del Mo^{99} ante una interrupción de la irradiación y un cambio de potencia 34
5.9. Actividad del del Mo^{99} ante una interrupción de la irradiación y un cambio de potencia posterior 35

5.10. Algoritmo del planificador 38
5.11. Familia de curvas de actividad generada para el cálculo de tiempo de irradiación ante cambios de potencia 39
5.12. Actividad generada según la planificación 39

6.1. Diagrama de flujo del algoritmo del simulador 42
6.2. Corte transversal del modelo de CITVAP del reactor OPAL 45
6.3. Actividad del Mo^{99} generada en el blanco calculada analíticamente y con esquemas de diferencias finitas 46
6.4. Actividad del Mo^{99} generada en el blanco ante un cambio de potencia calculado analíticamente y con un esquema de diferencias finitas 47
6.5. Actividad del Mo^{99} generada en el blanco ante una parada calculada analíticamente y con un esquema de diferencias finitas 48
6.6. Actividad generada y flujo neutrónico de una posición de irradiación calculados con el simulador 49
6.7. Evolución de la potencia durante la simulación 50
6.8. Evolución de la posición de las barras de control durante la simulación 50
6.9. Actividad calculada por el simulador afectada por la eficiencia del proceso de disolución 51

Índice de tablas

4.1. Resultados del informe del proceso de demostración de Molibdeno de Fisión 25

5.1. Demanda del caso de estudio 37
5.2. Planificación 37

6.1. Actividad de Mo^{99} en el séptimo día de irradiación calculada analíticamente y con dos esquemas de diferencias finitas 46
6.2. Actividad de Mo^{99} ante un cambio de potencia calculada analíticamente y con un esquema de diferencias finitas 47
6.3. Actividad de Mo^{99} generada en el blanco ante una parada calculada analíticamente y con un esquema de diferencias finitas 47

Resumen

Los radioisótopos son un bien perecedero ya que su actividad disminuye a cada instante debido al decaimiento radiactivo, por ello es importante realizar una buena planificación de la producción en el reactor para garantizar producción continua y suministro a hospitales durante todo el año. Con este fin, se desarrolló un algoritmo Simulador-Planificador para desarrollar estrategias de irradiación a fin de garantizar el suministro ininterrumpido de radioisótopos.

Se diseñó e implementó un algoritmo planificador que a partir de las ecuaciones de activación de cada radioisótopo, el flujo y cantidad de las posiciones de irradiación, la cantidad demandada y la programación de potencia y paradas del reactor, calcula el tiempo de irradiación e indica cuántas posiciones de irradiación ocupar y por cuánto tiempo. Este se implementó para Mo^{99} y Lu^{177} producido por vía directa.

Se diseñó e implementó un algoritmo simulador que calcula la actividad generada de Mo^{99} minuto a minuto. El simulador toma datos históricos de posición de barras de control, estado de posiciones de irradiación, y potencia y si es necesario, realiza un cálculo de núcleo (con CTIVAP y un modelo simplificado del OPAL) para determinar el flujo en una determinada posición de irradiación y con el mismo la actividad generada minuto a minuto. Se probó el simulador con un archivo histórico de 9,65 días de posición de barras de control, estado de posiciones de irradiación y potencia y se corroboró su correcto funcionamiento.

Palabras clave: RADIOISÓTOPOS, MOLIBDENO99, PLANIFICADOR, SIMULADOR

Abstract

Radioisotopes are a perishable good as their activity decreases continuosly due to radioactive decay, so it is important to do a good production planning in the reactor to guarantee continuous production and supply of radioisotopes to hospitals throughout the year. To this end, a Simulator-Planner algorithm was developed to develop irradiation strategies to ensure the uninterrupted supply of radioisotopes.

A planning algorithm was designed and implemented that, based on the activation equations of each radioisotope, the flux and quantity of the irradiation positions, the quantity demanded and the programming of reactor power and stops, calculates the irradiation time and indicates how many irradiation positions to occupy and for how long. This was implemented for Mo^{99} and Lu^{177}produced by direct route.

A simulator algorithm was designed and implemented that calculates the activity generated from Mo^{99} minute to minute. The simulator takes historical data of control rod position, status of irradiation positions, and power and if necessary, performs a core calculation (with CTIVAP and a simplified model of OPAL) to determine the flow at a given irradiation position and with it the activity generated minute by minute. The simulator was tested with a 9.65-day history file of control rod position, radiation position status and power and its correct operation was verified.

Keywords: RADIOISOTOPES, MOLIBDENUM99, SCHEDULER, SIMULATOR

Capítulo 1

Motivación

Los radioisótopos son un bien perecedero ya que su actividad disminuye a cada instante debido al decaimiento radiactivo, por ello es importante realizar una buena planificación de la producción en el reactor para garantizar producción continua y suministro de radioisótopos a hospitales todo el año. Estos reactores productores deben suministrar los radioisótopos según los requerimientos del cliente en cuanto a actividad total, actividad específica y fecha de entrega teniendo en cuenta la capacidad y el régimen de operación del reactor.
Debido al decaimiento radiactivo, los radioisótopos no se pueden almacenar y su producción debe ser continua. Cualquier tipo de interrupción de la irradiación, ya sea programada (por mantenimiento o refuelling) o espuria (por ejemplo, por corte de suministro eléctrico) afecta a la producción de los mismos.
El incumplimiento de la entrega en tiempo y forma además de tener un costo económico tiene un costo social dado que los radioisótopos son un insumo para la detección y tratamiento de diversas enfermedades, particularmente el cáncer, para el cual la detección y tratamiento se deben hacer lo antes posible para obtener un buen prognosis. La falta del radioisótopo se traduce en tratamientos y estudios pospuestos o cancelados con consecuencias en la salud. Por otro lado, el historial de cumplimiento redunda en un beneficio extra para el reactor productor: el precio que puede obtener por cada Curie entregado.

La motivación de este proyecto es desarrollar un Simulador-Planificador que permita desarrollar estrategias de irradiación a fin de garantizar el suministro ininterrumpido de radioisótopos, a pesar de las paradas del reactor. Estas estrategias de irradiación tienen en cuenta las propiedades nucleares de cada radioisótopo, su historial de irradiación, las características de su blanco así como también las posiciones de irradiación disponibles en el reactor y su flujo. El producto final de las estrategias de irradiación es determinar tiempo de carga y descarga de los blancos con el fin de cumplir la entrega de la actividad requerida en la fecha indicada.

El Simulador-Planificador de Irradiación interactúa con el sistema control del reactor de donde obtiene datos de planta e interactúa con el código de gestión de núcleo para obtener los valores de flujo en cada posición de irradiación para la configuración planteada. Para el alcance de este trabajo, la interacción con el sistema de control del reactor es a través de un archivo histórico.

Capítulo 2

Introducción

2.1. Introducción a los radioisótopos

Los radioisótopos son la forma inestable de un elemento que emite radiación para transformarse en una forma más estable. Esta radiación es detectable fácilmente y puede causar cambios en la sustancia con la que interactúa. Estas características especiales hacen que los radioisótopos tengan aplicaciones útiles en medicina, industria y otras áreas.[1]

Un núcleo inestable puede ocurrir naturalmente o como resultado de una alteración artificial del átomo. La mayoría de los radioisótopos se producen artificialmente en reactores de investigación y aceleradores exponiendo un material blanco a partículas como neutrones o protones, seguidos de diferentes procesos químicos para llevarlos a la forma química requerida. Algunos radioisótopos utilizados en medicina nuclear tienen corta vida media, es decir que decaen rápidamente y son ideales para propósitos de diagnóstico mientras que aquellos de larga vida media tienen aplicación terapéutica. [2]

Los átomos con núcleo inestable logran la estabilidad deshaciéndose de partículas y energía en exceso en la forma de radiación. Esta radiación emitida puede tomar varias formas, estas incluyen gamma, beta, alfa, neutrones. Este proceso de desintegración se denomina decaimiento radiactivo y la tasa de desintegración se denomina actividad. El período de semidesintegración ($T_{1/2}$) es el tiempo que tarda un radioisótopo en decaer a la mitad de su actividad inicial. El decaimiento es exponencial y prácticamente toda la actividad desaparece luego de 7 vidas medias. La actividad se mide en una unidad llamada Becquerel (Bq), 1 Bq equivale a una desintegración por segundo. Otra unidad para cuantificar la actividad es el Curie (Ci) que equivale a $3,7 \cdot 10^{10}$ desintegraciones por segundo. [3]

2.2. Aplicaciones de radioisótopos

Los radioisótopos han tenido un rol importante en mejorar la calidad de vida de los seres humanos. Su amplio rango de aplicación se debe a las siguientes características de los radioisótopos y de la radiación ionizante[4]:

- Los radioisótopos pueden ser detectados fácilmente en cantidades pequeñas con instrumentos relativamente simples
- Se utilizan pequeñas cantidades de material con consecuencias ambientales despreciables en cuanto a efectos a la salud
- Utilizando radioisótopos se puede determinar la composición de una muestra desconocida con gran certeza
- Una fuente radiactiva entrega energía de forma confiable

A continuación se presentan los usos más comunes de los radioisótopos. [5]

2.2.1. Aplicaciones industriales

La industria utiliza radioisótopos de diversas maneras para mejorar la productividad y ganar información que no podría ser obtenida de otra manera. Los radioisótopos se utilizan como trazadores para monitorear flujo de fluido y el mezclado de líquidos, polvos y gases, detectar fugas, estimar el desgaste de motores y la corrosión de equipos. Se pueden detectar pequeñas concentraciones de radioisótopos de vida media corta sin dejar residuos en el medio ambiente.

Los materiales radioactivos son utilizados para inspeccionar partes metálicas y la integridad de soldaduras en varias industrias. Se utilizan medidores con fuentes gamma para chequear niveles de gases líquidos y solidos en diversas aplicaciones industriales. Estas miden la cantidad de radiación de una fuente que ha sido absorbida en el material. Dichos medidores son útiles en aplicaciones donde no es posible utilizar medidores de contacto directo. La habilidad de los radioisótopos para medir espesores con gran exactitud es utilizada ampliamente en la producción de láminas de diversos materiales incluyendo metal, textiles, papel, plástico y otros.

2.2.2. Aplicaciones científicas

En geología, arqueología, y paleontología, se utilizan los radioisótopos naturales para medir las edades de las rocas, minerales y materiales fósiles. En biología, los radioisótopos de carbono pueden servir como trazadores radioactivos porque son químicamente muy similares a los isótopos no radioactivos, por lo que la mayoría de los

procesos químicos, biológicos y ecológicos los tratan de una manera similar. En ecología, los radioisótopos se utilizan como trazadores de agentes contaminantes, para estudiar el movimiento de las aguas superficiales, para medir las corrientes de lluvia y nieve, así como los caudales de arroyos y ríos. Mediante activación neutrónica se puede obtener información sobre los radioisótopos constituyentes de una muestra desconocida. El beneficio de usar neutrones no es solo que la técnica no sea destructiva sino que es capaz de ver elementos ligeros (plásticos, materiales orgánicos) detrás de metales pesados.

2.2.3. Aplicaciones en agricultura

Al exponer los alimentos a altos niveles de radiación, se pueden obtener beneficios tales como eliminar las bacterias que transmiten enfermedades a través de los mismos y aumentar su vida útil. Esta técnica generalmente tiene grandes beneficios por sobre otras formas de conservación de alimentos, técnicas de limpieza o esterilización como el calentamiento y los tratamientos químicos, ya que con la irradiación, los alimentos generalmente no se ven alterados notablemente.

El mejoramiento por mutación de plantas es el proceso de exponer las semillas o gajos de una planta dada a la radiación gamma, para causar mutaciones. El material irradiado se cultiva luego para generar un plantín. Estos se seleccionan y multiplican si muestran los rasgos deseados. Los países que han utilizado el mejoramiento genético de mutaciones de plantas con frecuencia han obtenido grandes beneficios socioeconómicos.

Los fertilizantes que contienen con un isótopo particular, como el N^{15} o el P^{32}, proporcionan un medio para descubrir cuánto absorbe la planta y cuánto se pierde, permitiendo un mejor manejo de la aplicación de fertilizantes. El uso de N^{15} también permite evaluar la cantidad de nitrógeno que las bacterias de las raíces de legumbres fijan en el suelo.

Además, la radiación se usa para controlar las poblaciones de insectos a través de la Técnica de insectos estériles (SIT). La SIT implica criar grandes poblaciones de insectos que se esterilizan mediante irradiación (rayos gamma o rayos X) e introducirlos en poblaciones naturales.

2.2.4. Aplicaciones en productos de consumo

Los detectores de humo utilizan la ionización creada por partículas alfa para medir el contenido de aerosol en el aire. Si el contenido de aerosol (humo) excede un nivel razonable, la ionización del aire disminuye y la alarma sonará alertando a los ocupantes del peligro del humo. El radioisótopo utilizado en los detectores de humo suele ser de Am^{241}.

El tritio y el prometio-147 liberan partículas beta durante la descomposición, desencadenando una reacción química al contacto con otros materiales. Esta reacción crea un "brillo"tal como lo percibe el ojo humano. Estas propiedades, conocidas como radioluminiscencia, hacen que los isótopos sean útiles para proporcionar "luz.en ausencia de electricidad. Estos materiales se utilizan en relojes y miras de armas para que puedan usarse de noche, así como en letreros en edificios comerciales, aviones y barcos.

2.2.5. Aplicaciones en medicina

La medicina nuclear usa radiación para proporcionar información sobre el funcionamiento de los órganos específicos de una persona o para tratar enfermedades.[6]

Esterilización

La radiación se utiliza para esterilizar instrumental quirúrgico. El uso de grandes dosis de radiación puede matar fácilmente a bacterias y debilitar muchos agentes virales sin dejar ni rastro de radiactividad en los instrumentos.

Diagnóstico

Los radioisótopos son una parte esencial de los procedimientos de diagnóstico médico. Al utilizar radiofármacos para el diagnóstico, se administra una dosis radiactiva al paciente y la actividad en el órgano puede estudiarse como una imagen bidimensional o, mediante tomografía, como una imagen tridimensional. Las técnicas de diagnóstico en medicina nuclear utilizan trazadores radiactivos que emiten rayos gamma desde el interior del cuerpo. Estos trazadores son generalmente isótopos de vida corta unidos a compuestos químicos que permiten el escrutinio de procesos fisiológicos específicos. Se pueden administrar por inyección, inhalación u oral. En combinación con dispositivos de imágenes que registran los rayos gamma emitidos desde el interior, pueden estudiar los procesos dinámicos que tienen lugar en diversas partes del cuerpo. Un radioisótopo utilizado para el diagnóstico debe emitir rayos gamma de suficiente energía para escapar del cuerpo y debe tener una vida media lo suficientemente corta como para que se descomponga poco después de que se complete la imagen.

Terapia

El uso de radioisótopos para la terapia se basa en la evidencia de que la radiación ionizante puede matar las células tumorales (cancerosas) en dosis muy grandes.

- Braquiterapia: se refiere a la radioterapia administrada por una fuente de irradiación interna al cuerpo. Los radioisótopos encapsulados en pequeñas pastillas de metal se implantan quirúrgicamente en un tumor. Se usan emisores beta porque

tienen un rango corto en el tejido, su energía se deposita localmente dentro del tumor y la dosis de radiación de estos gránulos se administra al tumor y no al tejido sano circundante. Los gránulos se extraen quirúrgicamente después de la dosis acumulada prescrita.

- Teleterapia: se refiere a la radioterapia administrada por una fuente de radiación externa a una distancia del cuerpo. Es el tipo más común de radioterapia utilizada en el tratamiento del cáncer y generalmente se administra mediante una unidad de cobalto, que suministra rayos gamma de alta energía.

- Radiofármacos terapéuticos: en este caso, el radioisótopo que genera la radiación se puede localizar en el órgano requerido a través de un elemento radiactivo que sigue su ruta biológica habitual, o mediante la unión del elemento a un compuesto biológico adecuado. En la mayoría de los casos, es la radiación beta que provoca la destrucción de las células dañadas. Un radioisótopo terapéutico ideal es un emisor beta fuerte con suficiente gamma para permitir la obtención de imágenes.

Capítulo 3

Radioisótopos

En esta sección se describirán las características de interés de los radioisótopos a producir.

3.1. Molibdeno 99

3.1.1. Características generales

El Molibdeno-99 es un isótopo radiactivo que decae por emisión beta con una vida media de aproximadamente 66 horas. Su esquema de decaimiento se muestra en la figura 3.1. El 87 % de estos decaimientos resultan en la producción del isótopo metaestable Tecnecio-99, que decae a su estado fundamental con una vida media de 6 horas y emite fotones de 140 keV. Esta energía fotónica es ideal para la detección eficiente mediante instrumentos de centelleo como las cámaras gamma. Los datos recopilados por la cámara se analizan para producir imágenes estructurales y funcionales detalladas.El Tc^{99m}, es el principal radioisótopo utilizado en medicina nuclear para diagnóstico por imágenes. El Tc^{99m} se usa para la detección de enfermedades y para el estudio de la estructura y función de los órganos. [7]

El Mo^{99} se valúa y se vende en unidades de radioactividad calibradas a un tiempo futuro, esto es necesario por el decaimiento radiactivo. Esta unidad se denomina 6-day Curie y corresponde a la actividad en Curie presente luego de 6 días después de dejar la planta de procesamiento. [8]

3.1.2. Usos

El Tc^{99m} se usa en aproximadamente el 80 por ciento de todos los procedimientos de medicina nuclear realizados en todo el mundo cada año. Es particularmente útil para el diagnóstico por imagen porque: [8]

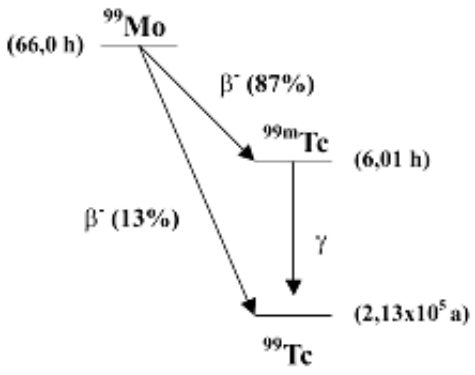

Figura 3.1: Esquema de decaimiento del Mo^{99}

- Puede incorporarse químicamente en radiofármacos que tienen afinidades por diferentes tejidos y sistemas de órganos.
- Tiene una vida media suficientemente larga (6 horas) para ser utilizable en procedimientos de medicina nuclear.
- Emite rayos gamma de 140 keV que pueden ser detectados eficientemente con tecnologías de cámara ampliamente disponibles
- Se puede suministrar eficientemente a hospitales y clínicas utilizando generadores de $Mo^{99} - Tc^{99m}$
- Proporciona dosis bajas al paciente para algunos procedimientos debido a su corta vida media y falta de radiaciones alfa o beta.

Los radiofármacos basados en Tc^{99m} se utilizan para diagnosticar enfermedades en una gran cantidad de sistemas de tejidos y órganos, incluidos huesos, cerebro, corazón, riñones, hígado y pulmones.

3.1.3. Producción

Actualmente, el proceso más utilizado para la producción de Tc^{99m} comienza con la irradiación en un reactor nuclear de mini placas de Uranio-Aluminio (U-Al_x). Esta irradiación provoca la fisión del U^{235} (principalmente, aunque otros isótopos también producen Mo^{99} por fisión) y produce Mo^{99} y muchos otros productos de fisión, incluidos I^{131} y Xe^{133}. A continuación se deben separar los radioisótopos de interés. La curva del yield de fisión se indica en la figura 3.3 donde se ve que el yield para fisiones térmicas en el U^{235} es 6.1 %, es decir que cada 100 fisiones de U^{235} se producen 6.1 átomos de Mo^{99}. La sección eficaz para esta reacción es aproximadamente de 584 barns para neutrones térmicos.[7]

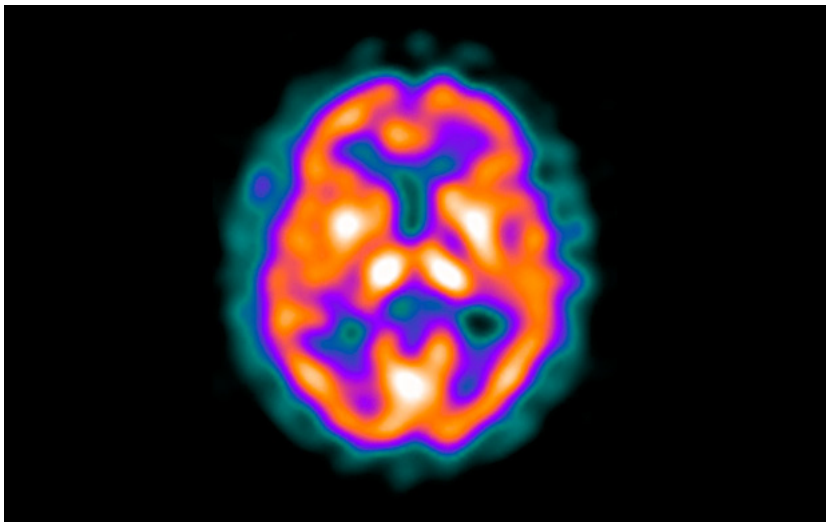

Figura 3.2: Centellograma de perfusión cerebral con Tc^{99m}

Luego de que las miniplacas son removidas del reactor, se dejan refrigerar en agua típicamente por medio día antes de ser transportadas en blindajes a la planta de separación, donde se manipulan en celdas calientes para su procesamiento químico. El proceso comienza con la disolución de las mini placas en un proceso químico de digestión alcalina y luego se recupera el Mo^{99} en sucesivos pasos de purificación con resinas de intercambio iónico. Dicho proceso químico demora 24 horas, aproximadamente, y tiene una eficiencia de alrededor del 60 %, es decir, no se puede recuperar el 100 % del Mo^{99} generado por fisión.[9]

La eficiencia de disolución disminuye a partir del quinto día de irradiación debido a que los productos de fisión generados dañan a las resinas involucradas en el proceso separativo.[10]

En la figura 3.4 se muestran los tiempos típicos de producción de Mo^{99}.

3.1.4. Mercado

La demanda global del Mo^{99} es de 12.000 6-day Curie por semana[7]

3.2. Ir-192

3.2.1. Características Generales

Tiene una vida media de 74 días, alta actividad especifica, decae por beta y gamma a un isótopo estable, Pt^{192}.Los rayos beta emitidos presentan una energía que va desde los 530 keV a 670 keV, y los gamma tienen una energía promedio de 370 keV. En la figura 3.5 se muestra su esquema de decaimiento. Se produce por la activación del Ir^{191} ($^{191}_{77}Ir(n,\gamma)^{192}_{77}Ir$) cuya sección eficaz de activación neutrónica es de 910 barn [11].

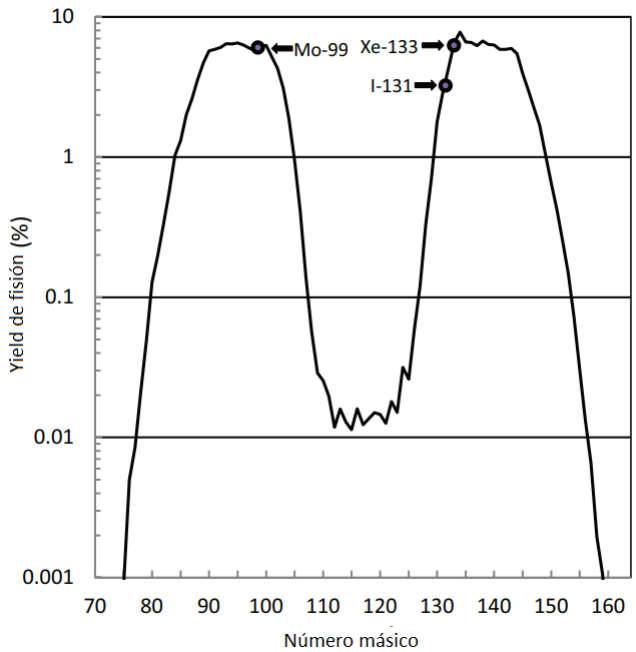

Figura 3.3: Yield de fisión del U-235

Usos

La braquiterapia a una distancia muy cercana es una forma de tratamiento de lesiones que se basa en la inserción de fuentes, en este caso alambres de Ir^{192} activados en el interior de los tumores. Durante este proceso, la radiación ionizante destruye las células malignas de manera muy eficiente.[11] La principal aplicación industrial del Ir^{192} es en gammagrafía, que consiste en un método de prueba no destructiva en el que se pueden examinar muchos tipos de componentes fabricados para verificar la estructura interna y la integridad de los mismos. Esta técnica utiliza los rayos gamma provenientes de fuentes de Ir^{192}.[9]

3.2.2. Irradiación

Los blancos de irradiación son alambres flexibles de 0.3mm o 0.5mm de diámetro que luego son cortados según la necesidad. Estos alambres son de un alma de Platino-Iridio (80/20), encapsulados en un tubo de Platino o Acero inoxidable. Estos recubrimientos tienen la finalidad de filtrar las partículas beta.[11] También hay blancos en forma de discos, con diámetros variando de 0.5 mm a 3.5 mm y con espesores de 0.2 mm a 0.5 mm. Los blancos deben ser colocados en posiciones de alto flujo neutrónico, para fuentes de gammagrafía el flujo debe ser mayor a $2 \cdot 10^{14} \frac{n}{cm^2 s}$. El tiempo de irradiación suele ser de 30 a 70 días. Dado su alto peso en reactividad, deben ser extraídos con el reactor apagado. [9]

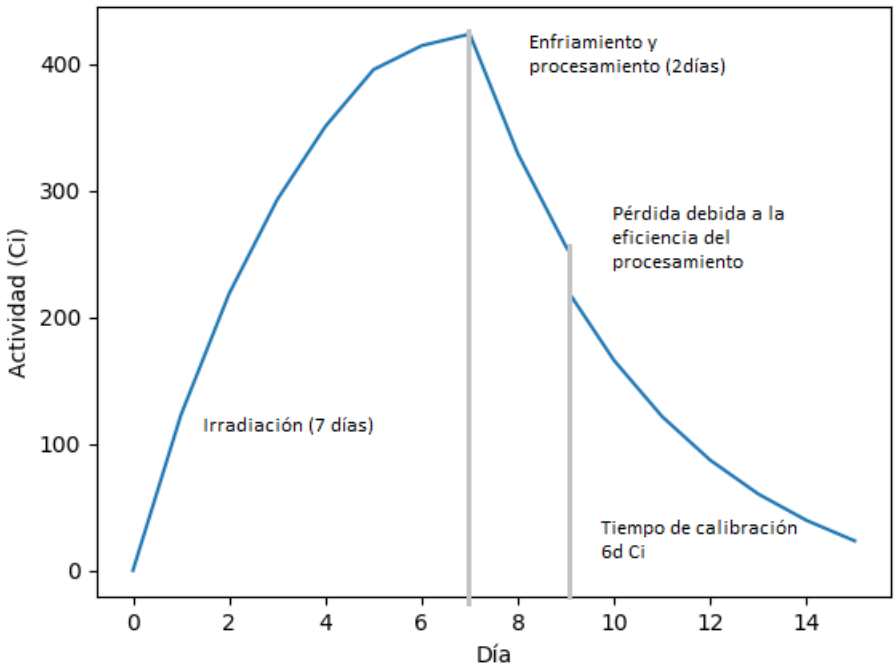

Figura 3.4: Duración típica de las etapas de producción de Mo^{99}

3.3. Lu-177

3.3.1. Características generales

Es un emisor β^- con una vida media de 6,65 días. Emite fotones gamma de baja energía, de 208 keV y 113 keV, ideales para realizar imágenes. En la figura 3.6 se muestra su esquema de decaimiento y sus dos modos de producción: vía directa e indirecta. La vía directa es por activación neutrónica del Lu^{176}, y la vía indirecta es por activación del Yb^{176} seguido por un decaimiento β^-. [12]

3.3.2. Usos

Es un emisor β^- de corto alcance para TRT (*Targeted Radionuclide Therapy*). La terapia con radioisótopos o TRT se encarga de administrar dosis de radiación preferiblemente a células cancerosas, incluso cuando su ubicación en el cuerpo es desconocida. Se aprovecha el hecho de que las células que se dividen rápidamente son particularmente sensibles al daño por irradiación. Por esta razón, algunos crecimientos cancerosos se pueden controlar o eliminar irradiando el área que contiene el crecimiento. Para que los radioisótopos lleguen al órgano deseado, en la mayoría de los casos tienen que ser llevados a cuestas de otra sustancia no radiactiva, una molécula biológicamente activa, llamada marcador. Tiene aplicaciones en paliación del dolor de hueso, radioinmunoterapia, radiosinovioterapia [13].

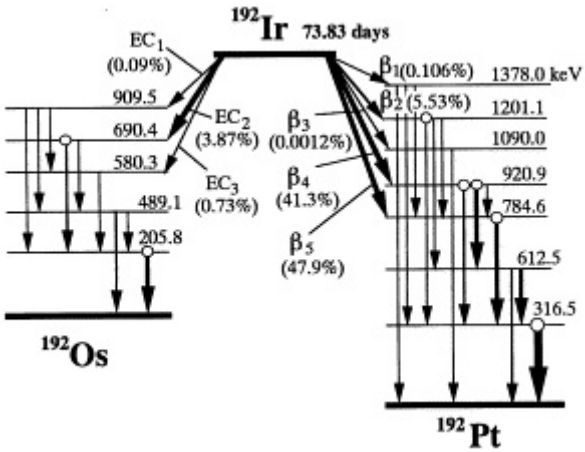

Figura 3.5: Esquema de decaimiento del Ir^{192}

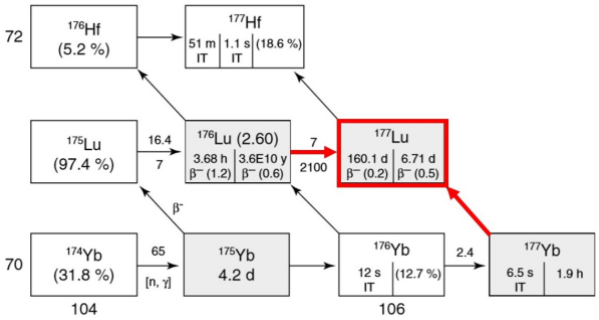

Figura 3.6: Esquema de decaimiento del Lu^{177}

3.3.3. Vía de producción directa $({}^{176}_{71}Lu(n,\gamma){}^{177}_{71}Lu)$

En la figura 3.7 se muestra la sección eficaz de esta reacción y se observa la existencia de una resonancia en 0.1413 eV. Al irradiar un target de Lu^{176} se coproduce un radioisótopo de larga vida media a través de la reacción $({}^{176}_{71}Lu(n,\gamma){}^{177m}_{71}Lu)$. En la figura 3.8 se muestra la sección eficaz de la reacción competitiva que baja la pureza radionucleidica del Lu^{177} y produce un desecho de larga vida media. Para esta vía de producción se requiere un target enriquecido ya que la abundancia isotópica del Lu^{176} es solo 2,6 %. Una limitación de esta vía de producción es que solo se puede obtener carrier added Lu^{177} ya que hay isótopos de Lu^{175} y Lu^{176} presentes en el target. [12]

3.3.4. Vía de producción indirecta $({}^{176}_{70}Yb(n,\gamma){}^{177}_{70}Yb) \xrightarrow{\beta-} {}^{177}_{71}Lu)$

Como una ruta de producción alternativa, se puede obtener Lu^{177} prácticamente sin carrier added del decaimiento β^- del Yb^{177} producido por la captura neutrónica de Yb^{176} en un reactor nuclear. La Figura 3.9 muestra la sección eficaz de la reacción $({}^{176}_{70}Yb(n,\gamma){}^{177}_{70}Yb)$. Para suprimir la formación de isótopos Yb^{169} y Yb^{175} y para aumentar la efectividad del proceso, se requiere un target enriquecido, para lo cual hay disponible comercialmente óxidos de iterbio enriquecidos hasta aproximadamente 97 %.

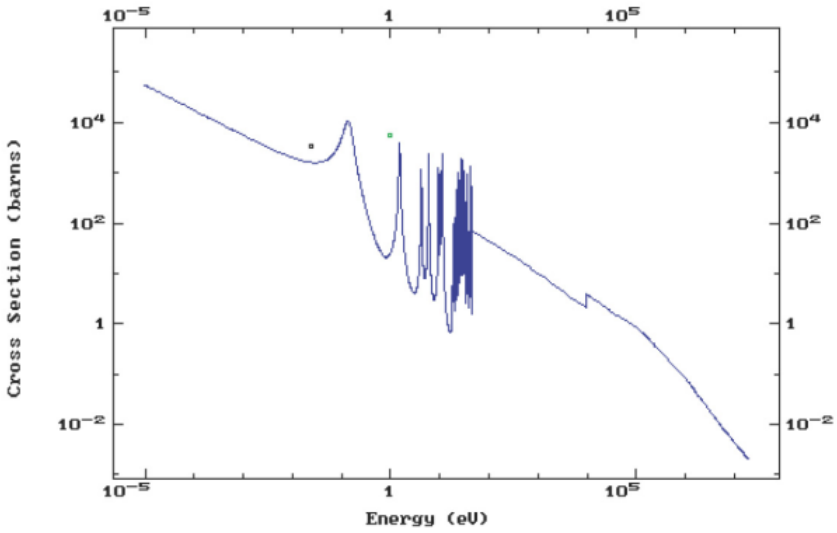

Figura 3.7: Sección eficaz para la reacción $({}^{176}_{71}Lu(n,\gamma){}^{177}_{71}Lu)$

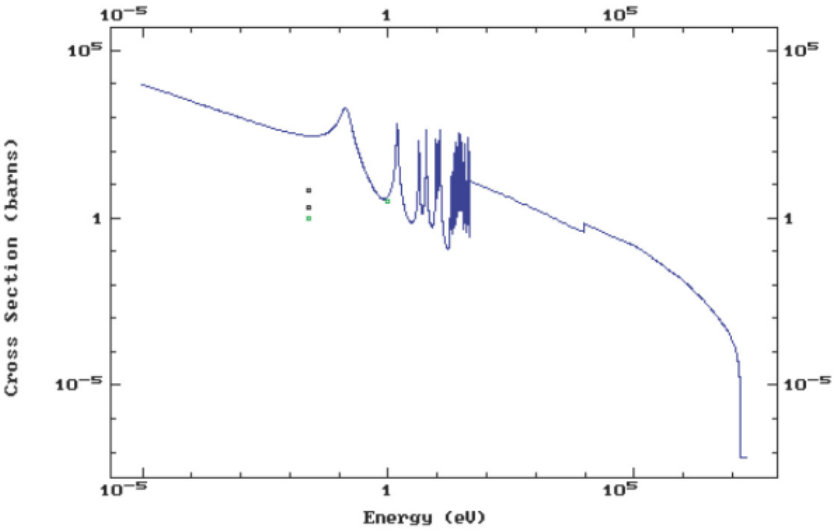

Figura 3.8: Sección eficaz de la reacción $({}^{176}_{71}Lu(n,\gamma){}^{177m}_{71}Lu)$

Para obtener Lu^{177} *non carrier added* de actividad específica máxima, el lutecio debe separarse del target de iterbio, porque el iterbio compite con el lutecio en la formación de complejos durante la producción radiofarmacéutica. [12]

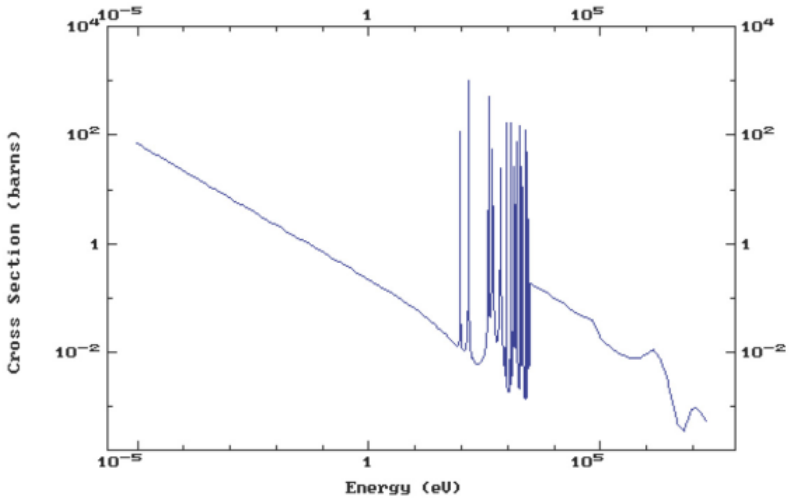

Figura 3.9: Sección eficaz de la reacción $({}^{176}_{70}Yb(n,\gamma){}^{177}_{70}Yb)$

Capítulo 4

Modelo

A continuación se detallarán las ecuaciones de Bateman particularizadas para diversos casos y para cada modelo de producción, por fisión, activación, y activación seguida por decaimiento β^-. Se analizarán además las ecuaciones de Bateman para los casos de interés que ocurren en el reactor:

- Interrupción de la irradiación
- Reanudación de la irradiación luego de una parada
- Cambio de potencia
- Reanudación de la irradiación luego de una parada con cambio de potencia

4.1. Reacción (n,γ)

Cuando un isótopo blanco ${}^{A}_{Z}X$ se encuentra bajo radiación en un reactor, se forma el radionucleido ${}^{A+1}_{Z}Y$ por la reacción (${}^{A}_{Z}X(n,\gamma){}^{A+1}_{Z}Y$) y tan pronto como se forma, comienza a decaer (${}^{A+1}_{Z}Y \xrightarrow{\beta} {}^{A+1}_{Z+1}S$). La tasa de producción neta de un radionucleido se expresa como su ritmo de reacción menos su ritmo de decaimiento.[14]

$$\frac{dN_Y}{dt} = N_X \phi \sigma_X - N_y \lambda_Y \tag{4.1}$$

Donde

- N_Y es la densidad numérica de átomos radiactivos de Y que se han formado
- N_X es es la densidad numérica del blanco X
- ϕ es el flujo neutrónico ($cm^{-2}s^{-1}$)
- σ_X es la sección eficaz de captura de X (cm^{-2})

- λ_Y es la constante de decaimiento de Y (s^{-1})

La integración de esta ecuación con actividad inicial nula da:

$$N_Y = \frac{N_X \phi \sigma_X}{\lambda_y}(1 - e^{-\lambda_Y t}) \tag{4.2}$$

La actividad de Y es:

$$A_Y = N_X \phi \sigma_X (1 - e^{-\lambda_Y t}) \tag{4.3}$$

En la realidad, la actividad inducida en el blanco bajo irradiación será menor que la calculada con dicha ecuación debido a diversos factores como:

- efectos de apantallamiento en el blanco
- variación de de potencia en el reactor
- alteración del flujo debido a movimimientos de barra de control, entre otros.
- quemado del blanco
- quemado del producto

4.1.1. Quemado del target y del producto

Esto se da cuanto el material blanco tiene una gran sección eficaz de absorción y los tiempos de irradiación son largos.

$$\frac{dN_y}{dt} = \left(\frac{dN_y}{dt}\right)_{\text{irradiación}} + \left(\frac{dN_y}{dt}\right)_{decaimiento} + \left(\frac{dN_y}{dt}\right)_{quemadodelproducto} \tag{4.4}$$

Es decir,

$$\frac{dN_y}{dt} = N_X \phi \sigma_x - N_Y \lambda_Y - N_Y \phi \sigma_Y \tag{4.5}$$

Debido a que el target se quema,

$$N_X = N_{Xo} e^{-\phi \sigma_x t}$$

donde σ_y es la sección eficaz del núcleo producto. Si esta es baja, se puede despreciar el consumo del producto. Integrando dicha ecuación con actividad inicial nula da:

$$N_Y = \frac{N_{X0} \phi \sigma_X}{\lambda_Y - \phi \sigma_X} \left(e^{-\phi \sigma_X t} - e^{-\lambda_Y t}\right) \tag{4.6}$$

Expresado en términos de actividad

$$A_Y(t) = \frac{N_{X0} \phi \sigma_X \lambda_Y}{\lambda_Y - \phi \sigma_X} \left(e^{-\phi \sigma_X t} - e^{-\lambda_Y t}\right) \tag{4.7}$$

Interrupción de la irradiación En caso de alguna posible parada de irradiación, el decaimiento durante la misma se expresa como:

$$A_Y(t) = \frac{N_{X0}\phi\sigma_X\lambda_Y}{\lambda_Y - \phi\sigma_X}\left(e^{-\phi\sigma_X t_i} - e^{-\lambda_Y t_i}\right)\cdot e^{-\lambda_Y\cdot(t-t_i)} \qquad t_i < t < t_f \tag{4.8}$$

donde t_i es el tiempo de irradiación y t_f es el tiempo de irradiación más el tiempo de parada.

Reanudación de la irradiación Si se reinicia la irradiación luego de la parada, se debe resolver la ecuación diferencial con la siguiente condición inicial

$$A_Y(t_f) = \frac{N_{X0}\phi\sigma_X\lambda_Y}{\lambda_Y - \phi\sigma_X}\left(e^{-\phi\sigma_X t_i} - e^{-\lambda_Y t_i}\right)\cdot e^{-\lambda_Y\cdot(t_f-t_i)} \tag{4.9}$$

La solución de la ecuación con esta condición inicial da:

$$A_Y(t) = \frac{N_{X0}\phi\sigma_X\lambda_Y}{\lambda_Y - \phi\sigma_X}\left[(e^{(-\phi\sigma_X+\lambda_Y)t_i} - 1 - e^{(-\sigma_X\phi+\lambda_Y)t_f})e^{-\lambda_Y t} + e^{-\phi\sigma_X t}\right] \qquad t > t_f \tag{4.10}$$

Este modelo aplica a la producción de Lu^{177} por vía directa. Se muestra la actividad para el mismo en función del tiempo para diversas situaciónes de interrupción de la irradiación en la figura 4.1. Se observa que ante una interrupción de la radiación, no se puede alcanzar la actividad original, esto es debido al quemado del target.

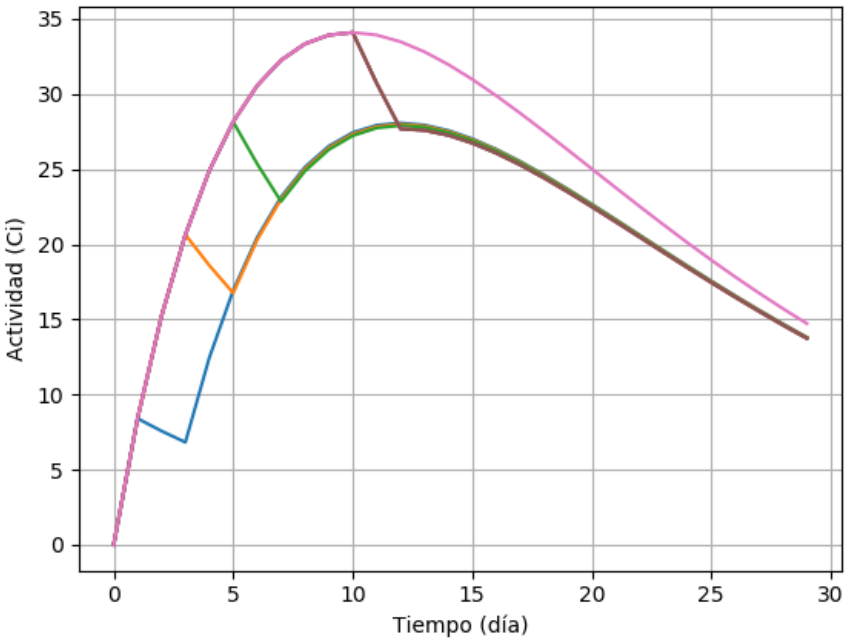

Figura 4.1: Actividad del Lu^{177} por vía directa ante una interrupción de la irradiación de 1 día, que ocurre en diversos momentos.

4.1.2. Irradiación seguida de decaimiento $\beta-$

Algunos radionucleidos son producidos por la reacción (n,γ) seguida por un decaimiento $\beta-$, es decir[14],

$$(X(n,\gamma)Y) \xrightarrow{\beta-} Z \xrightarrow{\beta-} S$$

En este caso, la tasa de producción del radionucleido Z está dada por:

$$\frac{dN_z}{dt} = N_X(1-e^{-\lambda t})\phi\sigma_X - N_Z\lambda_Z \tag{4.11}$$

Si no se tiene en cuenta el quemado del target, al integrar dicha ecuación con condición inicial nula se obtiene

$$N_Z = N_X\phi\sigma_X\left[\left(\frac{1-e^{-\lambda_Z t}}{\lambda_Z} + \frac{e^{-\lambda_Y t}-e^{-\lambda_Z t}}{\lambda_Y-\lambda_Z}\right)\right] \tag{4.12}$$

Expresado como actividad:

$$A_Z = N_X\phi\sigma_X\left[\left(1-e^{-\lambda_Z t} + (e^{-\lambda_Y t}-e^{-\lambda_Z t})\left(\frac{\lambda_Z}{\lambda_Y-\lambda_Z}\right)\right)\right] \tag{4.13}$$

Interrupción de la irradiación Si hubiera una interrupción de la irradiación, el decaimiento luego de la misma se expresa como:

$$A_Z = N_X\phi\sigma_X\left[\left(1-e^{-\lambda_Z t_i} + (e^{-\lambda_Y t_i}-e^{-\lambda_Z t_i})\left(\frac{\lambda_Z}{\lambda_Y-\lambda_Z}\right)\right)\right]\cdot e^{-\lambda_Z\cdot(t-t_i)} \qquad t_i < t < t_f \tag{4.14}$$

donde t_i es el tiempo de irradiación y t_f es el tiempo de irradiación más el tiempo de parada.

Renudación de la irradiación Al reanudar la irradiación, se comienza con una actividad de:

$$A_Z(t_f) = N_X\phi\sigma_X\left[\left(1-e^{-\lambda_Z t_i} + (e^{-\lambda_Y t_i}-e^{-\lambda_Z t_i})\left(\frac{\lambda_Z}{\lambda_Y-\lambda_Z}\right)\right)\right]\cdot e^{-\lambda_Z\cdot(t_f-t_i)} \tag{4.15}$$

Resolviendo la ecuación con esta condición de borde se obtiene la siguiente expresión para la actividad obtenida luego de reanudar la irradiación.

$$\begin{aligned} A_Z(t) = N_X\phi\sigma_X\left[\left(e^{-\lambda_Z t_i}-1-e^{-\lambda_Z t_f}\right)\cdot e^{-\lambda_Z t}+1\right] + \\ +N_X\phi\sigma_X\left(\frac{\lambda_Z}{\lambda_Y-\lambda_Z}\right)\left[\left(e^{t_i\cdot(\lambda_Z-\lambda_Y)}-1-e^{t_f\cdot(\lambda_Z-\lambda_Y)}\right)\cdot e^{-\lambda_Z t}+e^{-\lambda_Y t}\right] \qquad t > t_f \end{aligned} \tag{4.16}$$

Este modelo aplica a la producción de Lu^{177} por vía indirecta. En la figura 4.2

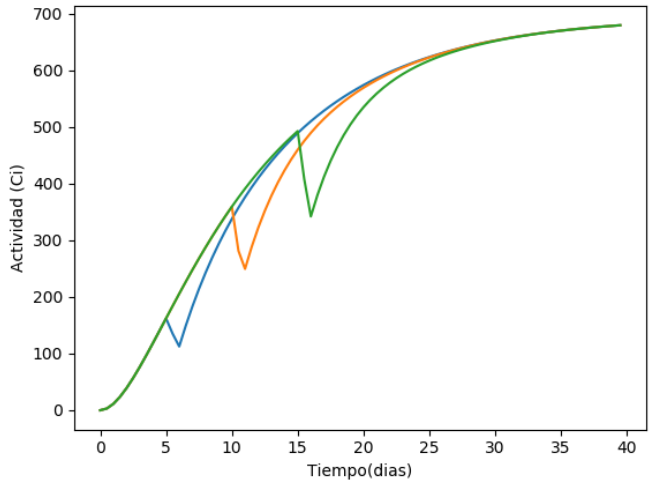

Figura 4.2: Actividad del Lu^{177} producido por vía indirecta en función del tiempo ante irradiación, interrupción de la misma y rerirradiación

se grafican las ecuaciones correspondientes a irradiación, interrupción de la misma y reirradiación, para una parada de 24 hs que ocurre en distintos momentos de la irradiación. A diferencia de la producción de Lu^{177} por vía directa, no se observa un máximo, y ser observa que al retomar la irradiación, si se logra alcanzar la actividad máxima.

4.2. Reacción de producción por fisión

El ^{99}Mo se produce a partir de fisión de placas de uranio. La tasa de producción del ^{99}Mo se expresa como[15]:

$$\frac{dN_{Mo}}{dt} = R - \lambda_{Mo} N_{Mo} \tag{4.17}$$

donde N_{Mo} es la densidad numérica del ^{99}Mo, λ_{Mo} es su constante de decaimiento, y R es el ritmo de reacción. Este está dado por:

$$R = y_{Mo}\, \sigma_{f^{235}U}\, \varepsilon\, \frac{\rho_U N_{av}}{M_U}\, \phi \tag{4.18}$$

Donde:

- y_{Mo} es el yield de fisión del ^{99}Mo para el U^{235}: $6,13 \cdot 10^{-2}$.
- $\sigma_{fU^{235}}$ es la sección eficaz térmica de fisión del ^{235}U: 580 b.
- ε es el enriquecimiento: 19,75 %.
- ρ_U es la densidad del uranio en el target, calculada como $\frac{\text{masa U235 en el meat}}{\text{volumen meat}}$.

- M_{U5} es la masa molar del U^{235}.
- N_{av} es el número de Avogadro.
- ϕ es el flujo térmico .

Expresando la ecuación 4.17 como actividad, se tiene la siguiente ecuación diferencial:

$$\frac{1}{\lambda}\frac{dA_{Mo}}{dt} = R - A_{Mo} \tag{4.19}$$

Esta ecuación tiene como solución general

$$A(t) = R + \mathbb{C} \cdot e^{-\lambda t} \tag{4.20}$$

donde la constante $\mathbb{C}$ de integración depende de la condición inicial. Finalmente, resolviendo la ecuación 4.19 con actividad inicial nula, la expresión para la actividad volumétrica es la siguiente:

$$A_{Mo}(t) = R(1 - e^{-\lambda_{Mo} t}) = y_{Mo}\, \sigma_{fU^{235}}\, \varepsilon\, \frac{\rho_U N_{av}}{M_U}\, \phi(1 - e^{-\lambda Mot}) \tag{4.21}$$

Interrupción de la irradiación Ahora bien, si hubo una interrupción de la irradiación por causa de un refueling, por ejemplo, la actividad decae exponencialmente:

$$A_{Mo}(t_f) = R \cdot [1 - e^{-\lambda t_i}] \cdot e^{-\lambda \cdot (t - t_i)} \tag{4.22}$$

Reanudación de la irradiación En este caso, se debe modificar la condición inicial de la ecuación diferencial, ya que se comienza con una actividad de:

$$A_{Mo}(t_f) = R \cdot [1 - e^{-\lambda t_i}] \cdot e^{-\lambda \cdot (t_f - t_i)} \tag{4.23}$$

donde t_i es el tiempo de irradiación antes de la parada y t_f es el tiempo de fin de parada. Resolviendo la ecuación 4.19 con dicha condición inicial, se tiene la siguiente expresión para la actividad en caso de una interrupción de la irradiación.

$$A_{Mo}(t) = R(1 - e^{-\lambda_{Mo} t}) + R[e^{\lambda(t_i - t)} - e^{\lambda(t_f - t)}] \qquad t > t_f \tag{4.24}$$

En la figura 4.3 se observa la Actividad del Mo^{99} durante una irradiación (donde vale la ecuación 4.21), interrupción de la irradiación (correspondiente a la ecuación 4.22) y finalmente reanudando la irradiación (ecuación 4.24).

Cambio de potencia Si ocurre un cambio de potencia en el reactor, que pasa de la potencia 1 a la potencia 2, en el momento t_c, se tiene la siguiente actividad en el

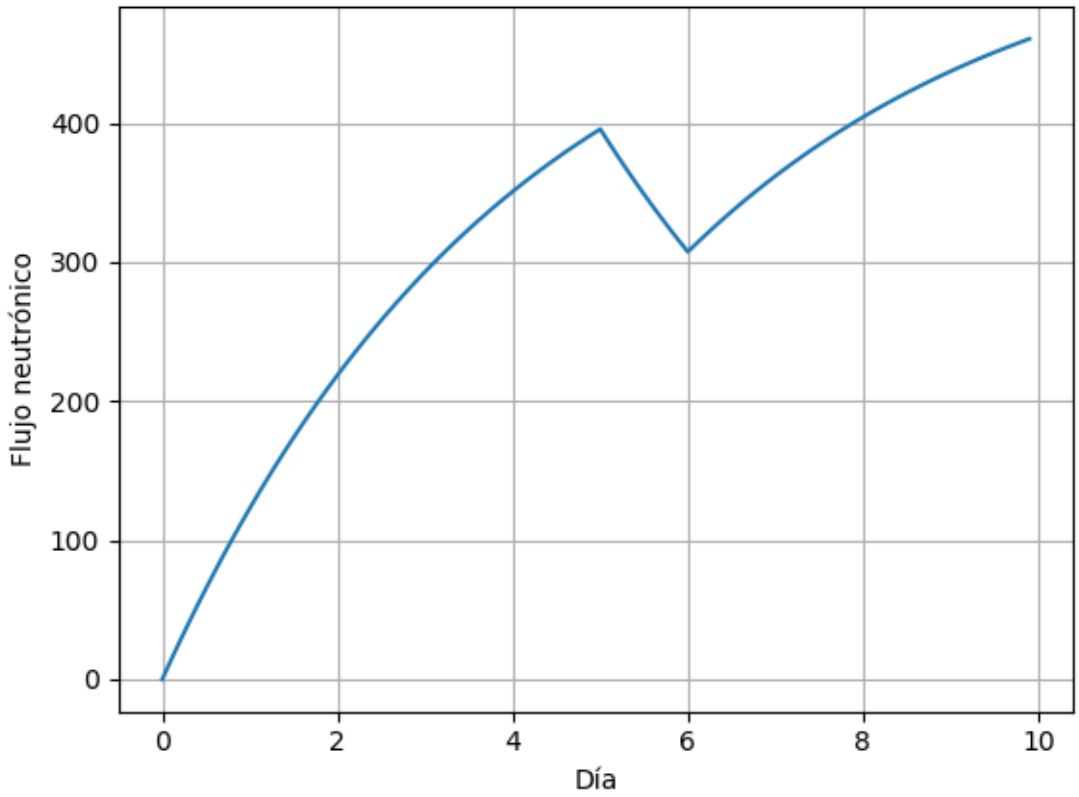

Figura 4.3: Actividad del Mo^{99} comenzando con una irradiación, luego interrumpiéndola, y finalmente retomando la irradiación

momento de cambio de potencia:

$$A(t_c) = R_1 \cdot (1 - e^{-\lambda t_c}) \tag{4.25}$$

donde R_1 es el ritmo de reacción correspondiente al flujo antes del cambio de potencia. Integrando la ecuación diferencial con dicha condición de borde se tiene:

$$A(t) = R_2 + \left[(R_1 - R_2) \cdot e^{\lambda \cdot (t_c - t)}\right] - R_1 \cdot e^{-\lambda \cdot t}, t > tc \tag{4.26}$$

donde R_2 es el ritmo de reacción luego del cambio de potencia.

Interrupción de la irradiación con cambio de potencia Si hubo una interrupción de la irradiación, y luego se retoma la irradiación con otra potencia, se modifica la condición de borde de la ecuación diferencial a:

$$A(t_f) = R_1(1 - e^{-\lambda t_i}) \cdot e^{-\lambda(t_f - t_i)} \tag{4.27}$$

donde R_1 es el ritmo de reacción previo al cambio de potencia. Con dicha condición de borde, la expresión para la actividad luego de reanudar la irradiación es:

$$A(t) = R_2(1 - e^{(\lambda(t_f - t))}) + R_1[e^{\lambda(t_i - t_f + t)}] - e^{-\lambda(t_f - t))}] \tag{4.28}$$

donde R_2 es el ritmo de reacción luego del cambio de potencia.

Interrupción de la irradiación con cambio de potencia posterior Si se reanuda la irradiación con el flujo original, pero luego ocurre un cambio de potencia en el momento t_c, la actividad en ese momento es:

$$A(t_c) = R_1(1 - e^{-\lambda t_c}) + R_1(e^{\lambda(t_i - t_c)} - e^{\lambda(t_f - t_c)}) \tag{4.29}$$

donde t_i es el tiempo de irradiación previa, y t_f es el final de la parada.

Resolviendo con dicha condición de borde se tiene la siguiente expresión:

$$A = R_2(1 - e^{\lambda(t_c - t)}) + R_1[e^{\lambda t_c} - 1 + e^{\lambda t_i} - e^{\lambda t_f}] \cdot e^{-\lambda t} \tag{4.30}$$

4.2.1. Verificación y Validación

La verificación del modelo consistió en determinar si las ecuaciones fueron bien programadas en el código. Para ello se utilizó una planilla Excel para corroborar que los resultados obtenidos del modelo se correspondan con los resultados obtenidos de Excel.

Se validó este modelo contra los resultados medidos del Informe del proceso de demostración de molibdeno de fisión - demostración para INVAP/ANSTO [16] y la referencia titulada “Thermal neutron measurements in the RA-3 irradiation facility”[17] con los datos de la tabla 4.1.

Tabla 4.1: Resultados del informe del proceso de demostración de Molibdeno de Fisión

Flujo térmico medio	1,30E+14
Yield de fisión del Mo	0,0613
Masa de U^{235} en el meat [g]	1,4
Sección eficaz de fisión del U^{235}	5,80E-22
Enriquecimiento %	0,1975
Largo del meat [cm]	11,7
Alto del meat [cm]	3
Ancho del meat [cm]	0,07
Densidad de uranio en el meat [g/cm3]	2,885
Cantidad de placas	12
Constante de decaimiento (λ) [1/día]	0,252
Inicio de irradiación	25/nov 22:00
Fin de irradiación	30/nov 00:00
Actividad medida [Ci]	**1750**
Momento de medición	02/dic 02:20
Eficiencia de procesamiento	0,6
Tiempo de procesamiento + enfriamiento [hora]	28,33
Tiempo de irradiación [día]	5
Actividad calculada [Ci]	**1717,05**

Capítulo 5

Planificador

El planificador es un código que a partir de los siguientes datos de entrada, satisface la producción de radioisótopos en la fecha solicitada:

- la cantidad demandada de Ci
- la fecha de entrega de dicha demanda
- las paradas programadas del reactor
- los cambios de potencia programados del reactor
- la cantidad de posiciones de irradiación
- los flujos de cada posición de irradiación

Utilizando las ecuaciones vistas en el capítulo Modelo, el planificador calcula el tiempo de irradiación con un proceso de prueba y error iterativo. El output del planificador consiste en indicar cuántas posiciones de irradiación ocupar y por cuánto tiempo.

5.1. Rendimiento

En el capítulo 4 se vio la resolución de las ecuaciones diferenciales de actividad del Mo^{99}, obteniéndose expresiones de la actividad en función del tiempo para diversas situaciones. A continuación debe multiplicarse dicha actividad por el rendimiento del proceso de disolución del Mo^{99} para obtener los Ci realmente producidos.

Como se mencionó anteriormente, el rendimiento del proceso de disolución del Mo^{99} es decreciente a partir del quinto día de irradiación. Para este trabajo se tomó un modelo lineal decreciente del rendimiento a partir del día 5. Este se indica en la figura 5.1

Aplicando dicho rendimiento a la actividad, se obtiene la figura 5.2 en la cual se observa que la actividad del Mo^{99} alcanza un máximo a los 7,8 días y luego disminuye. Esto se debe a que el rendimiento de disolución disminuye con el tiempo de irradiación.

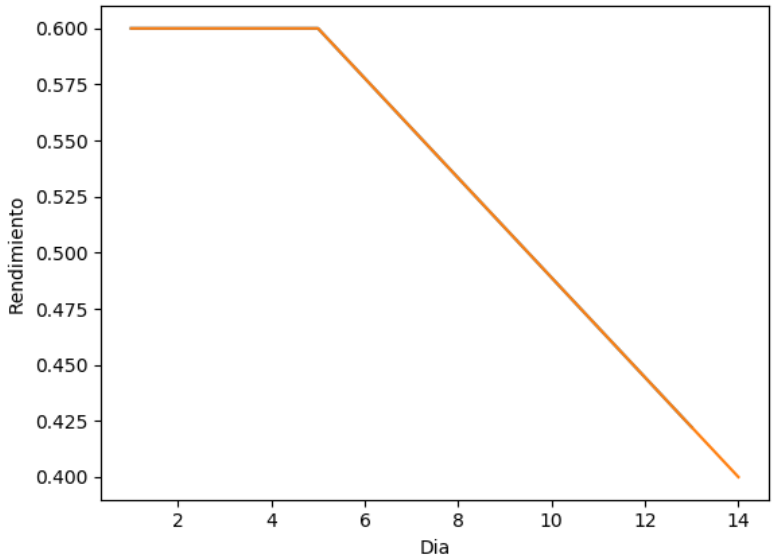

Figura 5.1: Rendimiento del proceso de disolución del Mo^{99} en función del tiempo de irradiación

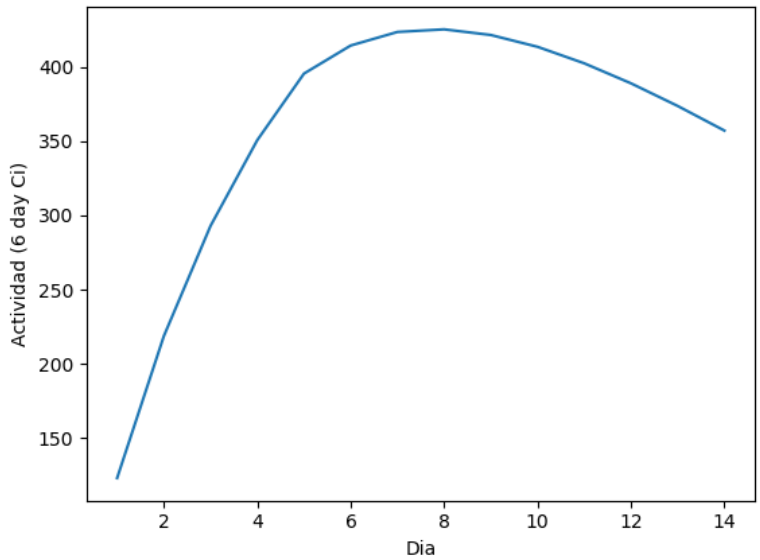

Figura 5.2: Actividad del Mo^{99} afectada por el rendimiento de disolución

5.2. Cambio de potencia

En la figura 5.3 se observa cómo se modifica la actividad producida efectivamente (afectada por el modelo de rendimiento mencionado anteriormente) ante una reducción de flujo neutrónico (es decir, un cambio de potencia) del 10 % en el día 5. Se observa que a partir del día de cambio de potencia, la curva de actividad sigue la misma forma funcional pero disminuida. Además, comparando con la figura 5.2 se obtiene

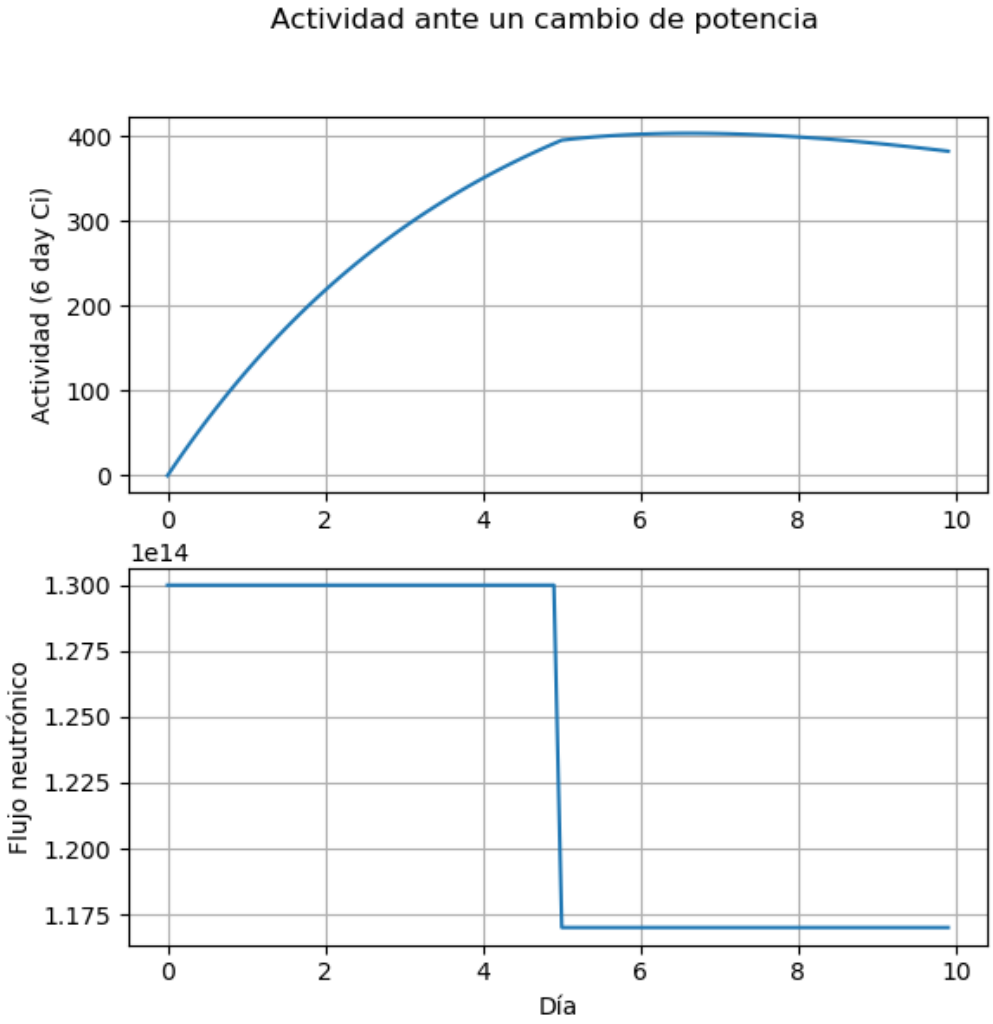

Figura 5.3: Actividad del Mo^{99} en función del tiempo ante una reducción de potencia del 10 %

una actividad menor que cuando no había cambio de potencia.

5.2.1. Efectos de cambios de potencia

En la figura 5.4 se muestran dos curvas de irradiación: en la azul se comienza con un flujo de $1{,}3 \cdot 10^{14} \frac{n}{cm^2 s}$ y en el día 5 se reduce un 10 %. En cambio, en la curva naranja, la situación es la inversa. Se observa que las dos curvas no llegan a la misma actividad final, es decir que no es los mismo irradiar 5 días al 100 % y luego 5 días al 90 % que viceversa. Este comportamiento indica que es importante cuando ocurre el cambio de potencia.

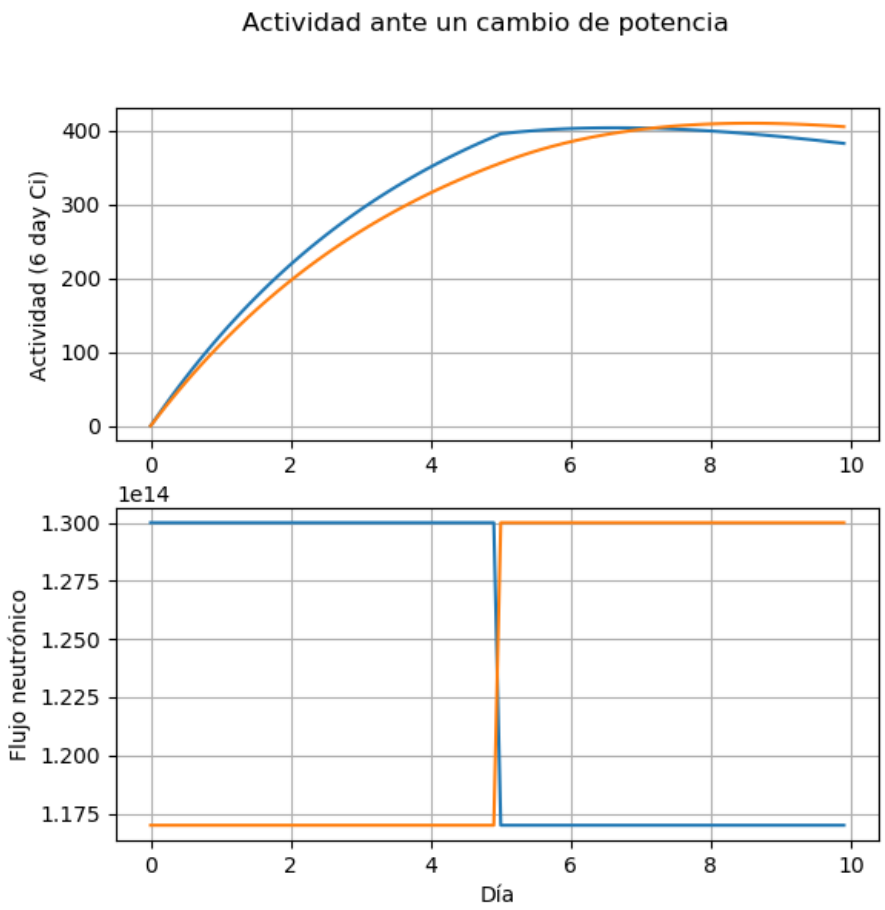

Figura 5.4: Actividad del Mo^{99} para un aumento y un descenso de pontencia.

5.3. Interrupción de la irradiación

5.3.1. Efecto de la interrupción de la radiación

Duración de la interrupción de la irradiación En la figura 5.5 se muestran curvas de la actividad del Mo^{99} ante interrupciones de diferente longitud, en las cuales el final de la interrupción ocurre un día antes de la entrega para todos los casos. Se observa que mientras más larga sea la interrupción de la irradiación, más tiempo previo de irradiación se requerirá para alcanzar la misma actividad final.

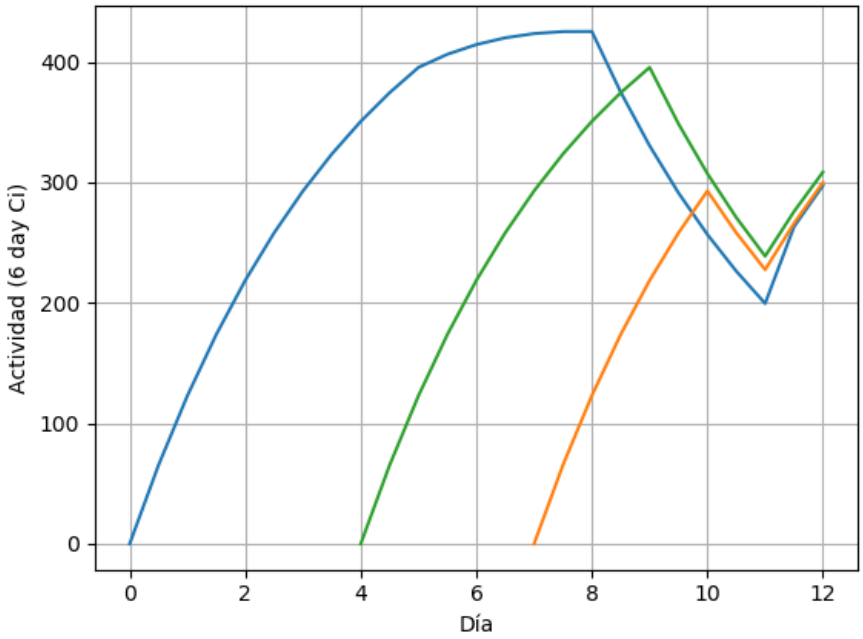

Figura 5.5: Actividad del Mo^{99} ante interrupciones de la irradiación de diversa longitud

Momento de ocurrencia de la interrupción de la irradiación En la figura 5.6 se grafican curvas de la actividad del Mo^{99} con un tiempo de parada de tres días, modificando el momento de ocurrencia de la misma. Se observa que las interrupciones de irradiación que ocurren más cerca de la entrega son más perjudiciales.

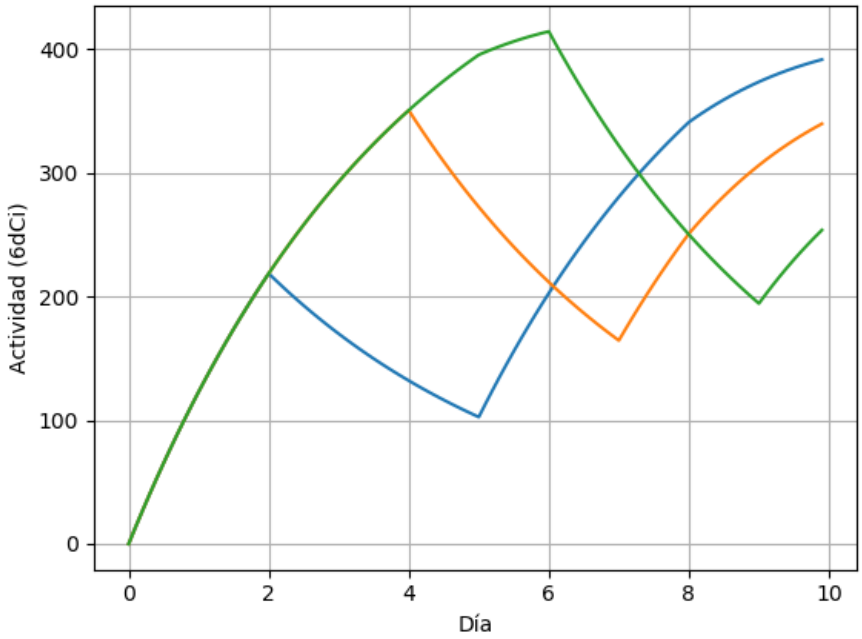

Figura 5.6: Actividad del Mo^{99} ante interrupciones de la irradiación que ocurren en diferentes momentos.

Dada una interrupción de la irradiación, se identificaron tres escenarios:

- Se reanuda la irradiación con el mismo flujo neutrónico con el cual se irradió previo a la interrupción
- Se reanuda la irradiación con un flujo diferente al que había antes de la interrupción
- Se reanuda la irradiación con el mismo flujo que había antes pero posteriormente hay un cambio de flujo.

A continuación se desarrolla cada uno de ellos.

5.3.2. Reanudación de la irradiación con el mismo flujo anterior

En la figura 5.7 se grafica la actividad en función del tiempo ante una interrupción de la irradiación, y una posterior reanudación de la misma, como podría suceder en un refueling.

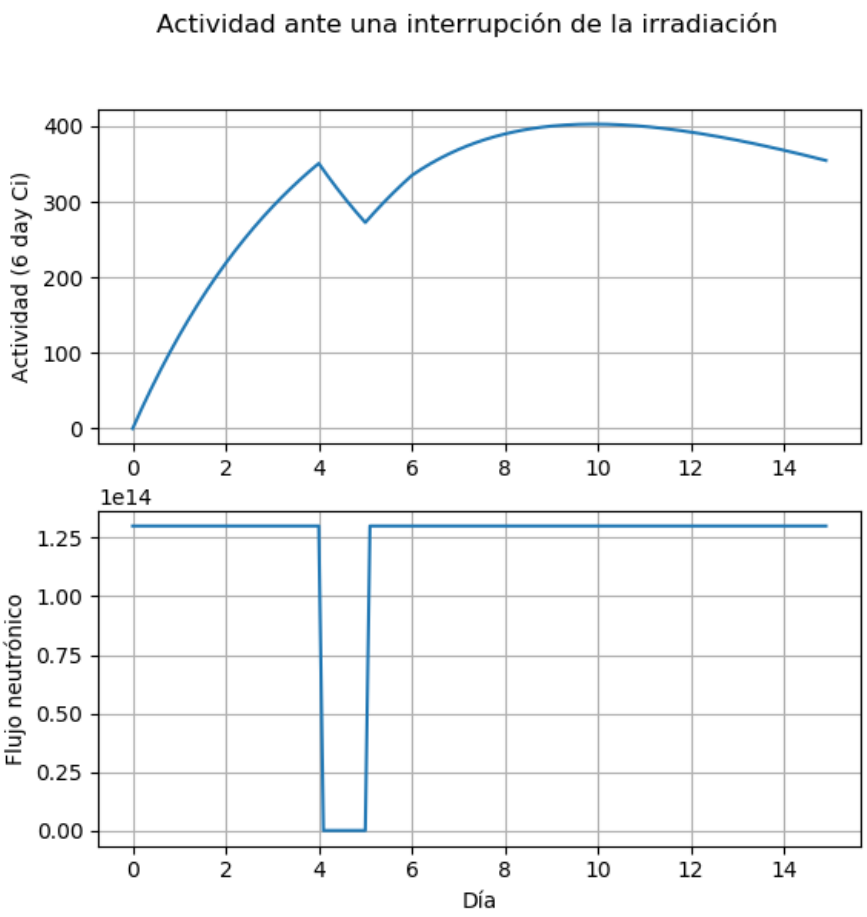

Figura 5.7: Actividad del Mo^{99} ante una interrupción de la irradiación

5.3.3. Reanudación de la irradiación con un flujo diferente al original

En la figura 5.8 se muestra la actividad ante una interrupción de la irradiación y una reanudación de la irradiación con un flujo 10 % menor al original. Comparando con el caso anterior, la figura 5.7, se observa que efectivamente la actividad luego de la irradiación es menor que el caso anterior y presenta una menor pendiente.

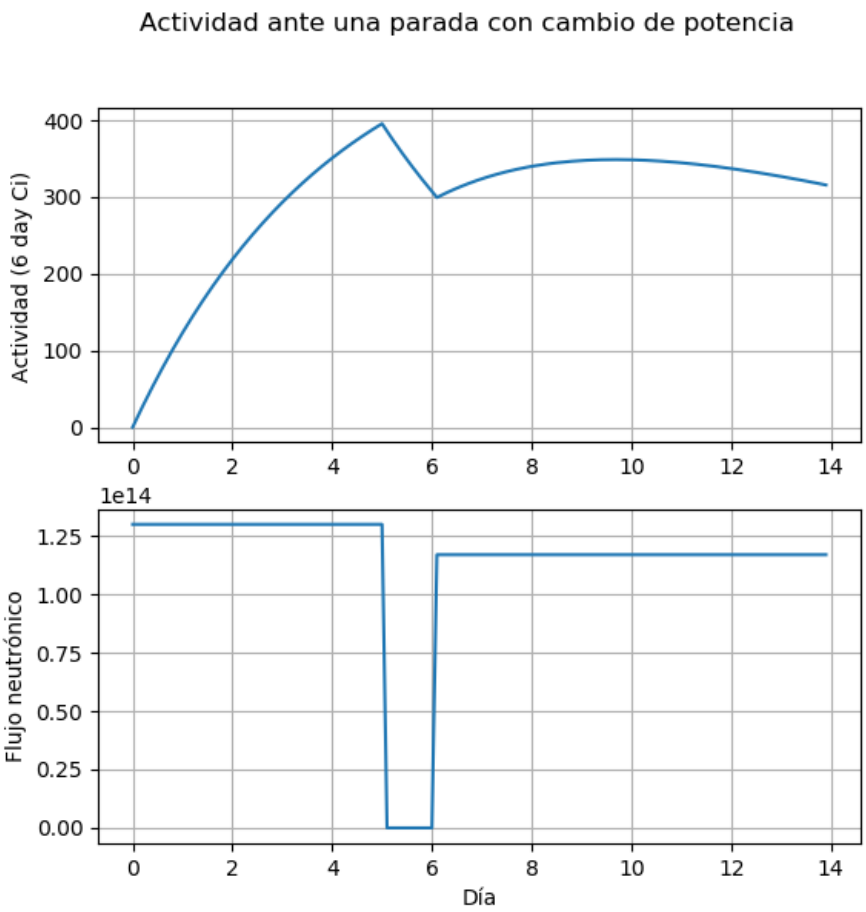

Figura 5.8: Actividad del del Mo^{99} ante una interrupción de la irradiación y un cambio de potencia

5.3.4. Reanudación de la irradiación con un cambio de flujo posterior

En la figura 5.9 se muestra la actividad ante una parada, una reanudación de la irradiación con el mismo flujo que antes, y un cambio de potencia en el día 7.

5.4. Algoritmo del planificador

En la figura 5.12 se muestra el algoritmo del planificador en un diagrama de bloques. El código para el planificador de Mo^{99} se encuentra en el apéndice B.

5.4.1. Input

El input al planificador es una matriz con los días a planificar en las filas, y en las columnas tiene el día, la demanda, las paradas planificadas (indicadas con un 1 si hay parada y un 0 si no la hay), y las posiciones de irradiación con su flujo correspon-

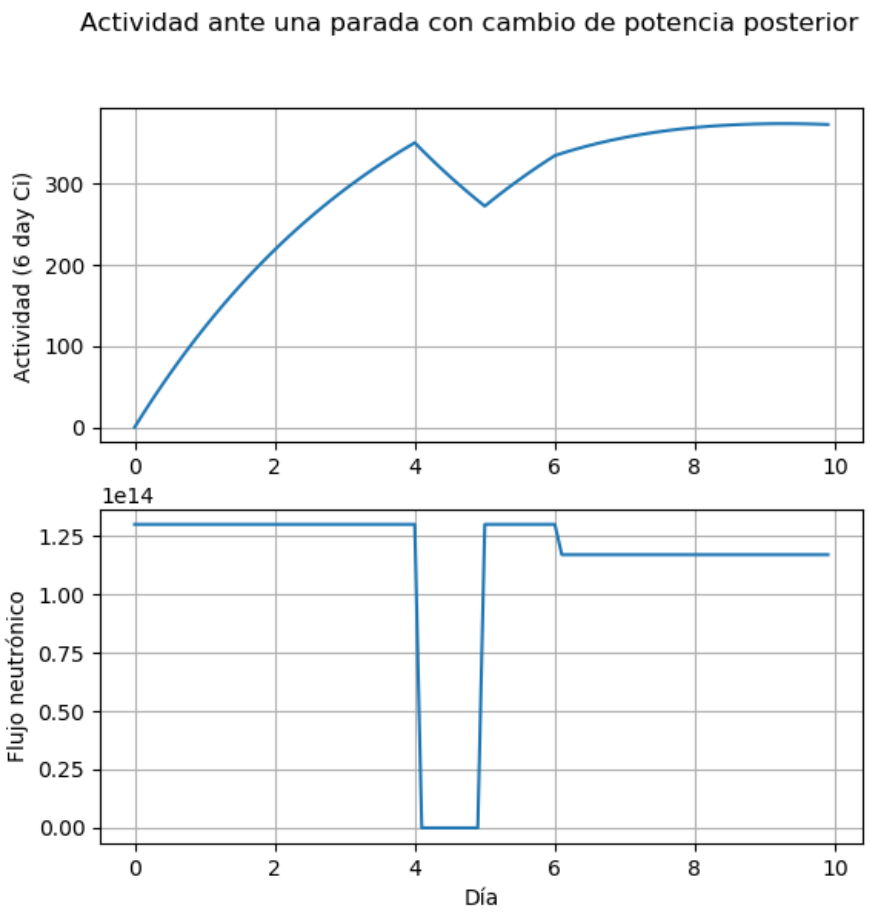

Figura 5.9: Actividad del del Mo^{99} ante una interrupción de la irradiación y un cambio de potencia posterior

diente. Además, se deben indicar las propiedades de las placas a irradiar, el tiempo de enfriamiento y procesamiento, y el rendimiento del proceso de disolución. Para este trabajo se consideró que los dispositivos de irradiación están ocupados por placas en su totalidad. Al ocupar una posición de irradiación, se llena el lugar correspondiente en la matriz con un 1. El flujo de cada posición de irradiación se lee del output de CITVAP (archivo de texto .cio) de referencia que indique el usuario. Se cuenta con funciones que permiten determinar si hubo una parada y/o un cambio de potencia a partir de dicha matriz.

5.4.2. Cálculo de tiempos de irradiación

En primer lugar se determina cuántas posiciones de irradiación es necesario ocupar, esto se determina realizando el cociente entre la demanda y la actividad máxima obtenible con el flujo de la posición de irradiación (se vio anteriormente que la actividad del Mo^{99} llega a un máximo y luego disminuye). Luego se procede a calcular el tiempo de irradiación. Dado que no se puede despejar el tiempo de irradiación analíticamente,

se lo calcula con un proceso de prueba y error iterativo.

Si no hay cambios de potencia ni interrupciones de irradiación, se trata de la situación de la figura 5.2. Para determinar el tiempo de irradiación dada una determinada actividad deseada y un flujo, se genera un vector de tiempo y actividad. Se recorre el vector de actividades y se compara cada uno de sus valores con el valor de la actividad objetivo. Si la actividad objetivo es menor o igual que la actividad del vector ese día, el tiempo de irradiación es el correspondiente a esa actividad. Si la actividad objetivo no se encuentra (es decir, que es mayor que todos los valores de la curva) se le asigna el tiempo de irradiación correspondiente a la máxima actividad de la curva.

Si hay un cambio de potencia, el tiempo de irradiación se calcula por prueba y error generando una familia de curvas de actividad para todos los posibles tiempos de irradiación. Por ejemplo, supongamos que el día 9 se deben entregar 400 Ci, y el día 7 hay una reducción de flujo de 100 % a 90 %. Entonces se tiene un número incógnita de días de irradiación al 100 % y 2 días de irradiación al 90 %. A continuación se generan curvas de irradiación con diversos tiempos de irradiación al 100 % y 2 días de irradiación al 90 %, como se ve en la figura 5.11. Luego se chequea para cada curva de actividad si para el último día de irradiación la actividad es mayor que la actividad objetivo. Si coincide, este es el tiempo de irradiación. Para el ejemplo, el tiempo de irradiación sería de 5 días al 100 % y de 2 días al 90 %. Para los casos de interrupción de la irradiación se procede de forma análoga.

5.4.3. Output

El output del planificador consiste en indicar qué posiciones cargar y cuánto tiempo irradiar. Ello se indica llenando el casillero correspondiente de la matriz input con un 1 durante los días de irradiación.

5.4.4. Planificador para Lu^{177} producido por vía indirecta

Adicionalmente, se realizó un planificador para Lu^{177} producido de forma indirecta, este solo contempla paradas de irradiación. El código se detalla en el apéndice C. El algoritmo y la metodología de tiempos de irradiación son iguales que las desarrolladas para el Mo^{99}. La única diferencia es que la actividad del Lu^{177} no presenta un máximo y luego decrece como el Mo^{99}, sino que crece y luego satura. Por ello se determinó que la actividad máxima posible a generarse en una posición de irradiación sea la correspondiente a la de 30 días de irradiación. Con esta actividad y la demanda se calcula la cantidad de posiciones de irradiación a ocupar y el tiempo de irradiación se calcula de manera análoga a las del Mo^{99} con las ecuaciones correspondientes.

5.4.5. Caso de aplicación

Para ilustrar el funcionamiento del planificador se lo probó con un caso hipotético en el cual se presenta una parada programada de duración de 1 día en el día 3 y la demanda que se muestra en la tabla 5.1. Se propuso un flujo de 1.3E+14 $\frac{n}{cm^2 s}$ para todas las posiciones de irradiación. Con este flujo se necesitaría ocupar 1 posición para la demanda del día 8, 2 posiciones para la del día 9 y 3 posiciones para la del día 10.

Tabla 5.1: Demanda del caso de estudio

Día	Demanda (6dCi)
8	400
9	800
10	1200

La solución que propone el simulador se indica en la tabla 5.2.

Tabla 5.2: Planificación

Día	Posición 1	Posición 2	Posición 3	Posición 4	Posición 5	Posición 6
0	Ocupado	Libre	Libre	Libre	Libre	Libre
1	Ocupado	Ocupado	Ocupado	Libre	Libre	Libre
2	Ocupado	Ocupado	Ocupado	Libre	Libre	Libre
3	Ocupado	Ocupado	Ocupado	Libre	Libre	Libre
4	Ocupado	Ocupado	Ocupado	Libre	Libre	Libre
5	Ocupado	Ocupado	Ocupado	Ocupado	Ocupado	Ocupado
6	Ocupado	Ocupado	Ocupado	Ocupado	Ocupado	Ocupado
7	Ocupado	Ocupado	Ocupado	Ocupado	Ocupado	Ocupado
8	Ocupado	Ocupado	Ocupado	Ocupado	Ocupado	Ocupado
9	Libre	Ocupado	Ocupado	Ocupado	Ocupado	Ocupado
10	Libre	Libre	Libre	Ocupado	Ocupado	Ocupado

La actividad generada según la planificación propuesta se grafica en la figura 5.12.

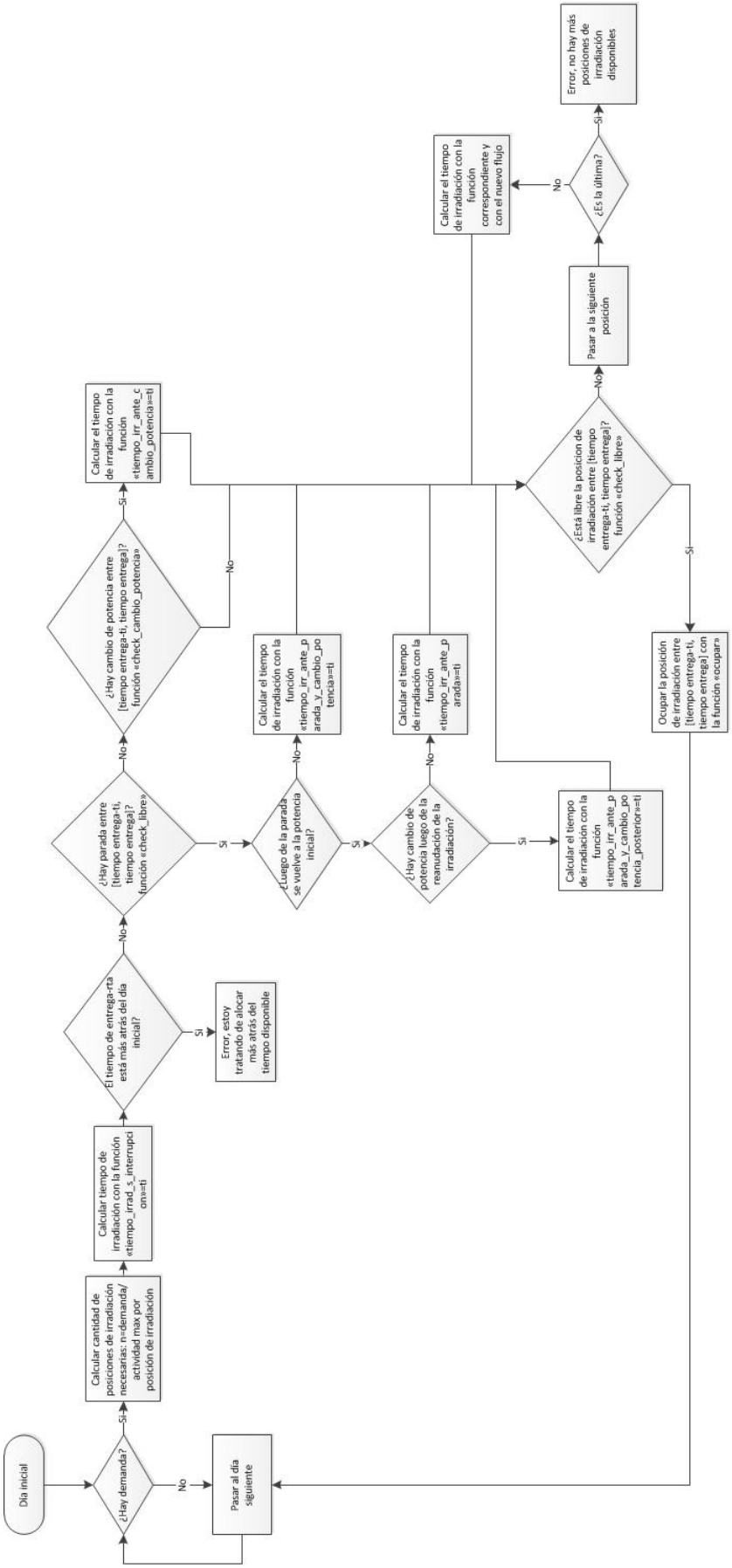

Figura 5.10: Algoritmo del planificador

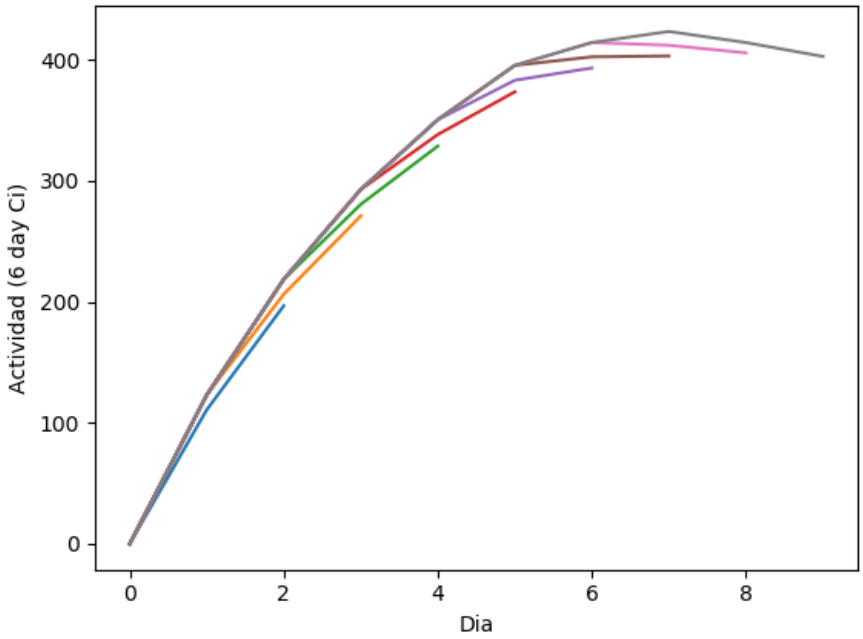

Figura 5.11: Familia de curvas de actividad generada para el cálculo de tiempo de irradiación ante cambios de potencia

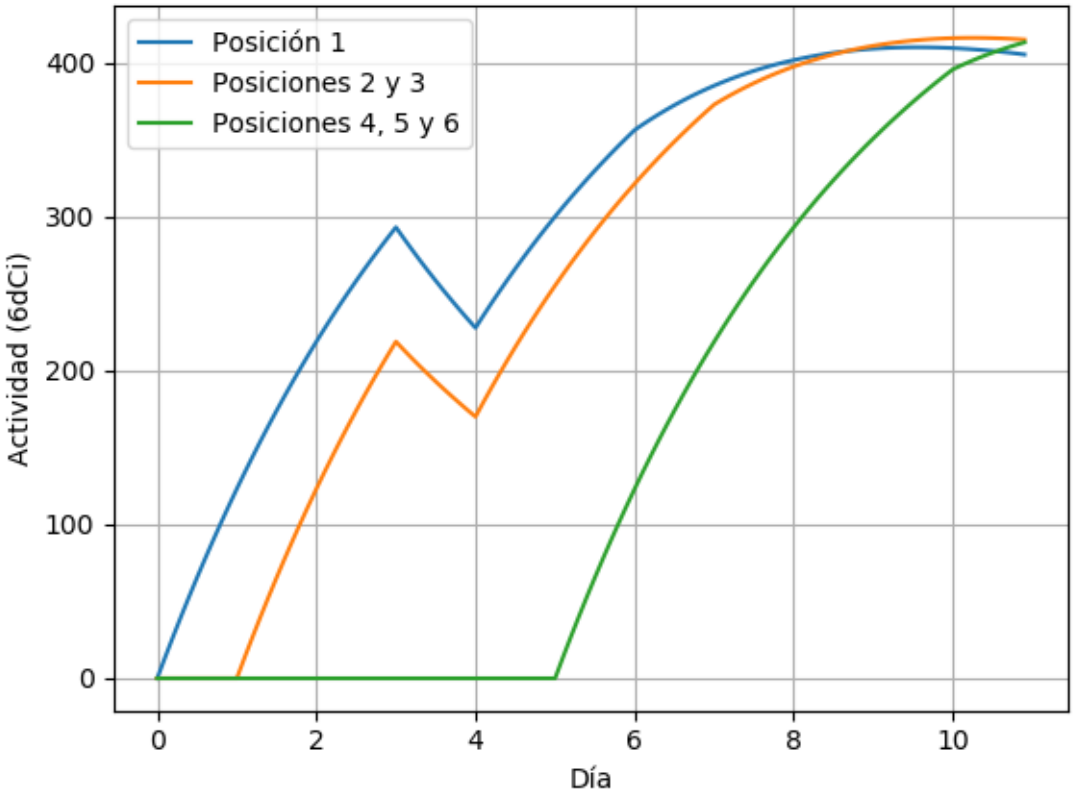

Figura 5.12: Actividad generada según la planificación

Capítulo 6

Simulador

El objetivo del simulador es correr o ejecutarse junto al reactor y calcular la actividad de Mo^{99} generada minuto a minuto. Si bien el planificador indica cuánto tiempo a irradiar según el flujo programado, pueden haber desviaciones en el mismo que generen discrepancias en la actividad generada respecto a la programada. Estas desviaciones pueden deberse a alteraciones del flujo causadas por el movimiento de una barra de control cercana o la carga/descarga de posiciones de irradiación cercanas o a variaciones de potencia. De aquí queda evidenciada la necesidad de un algoritmo que indique minuto a minuto la actividad generada.

6.1. Descripción general

A continuación se describe el algoritmo de cálculo del simulador. El mismo se realizó en el lenguaje Python y se detalla en el apéndice A. En la figura 6.1 se describe un diagrama de flujo del mismo.

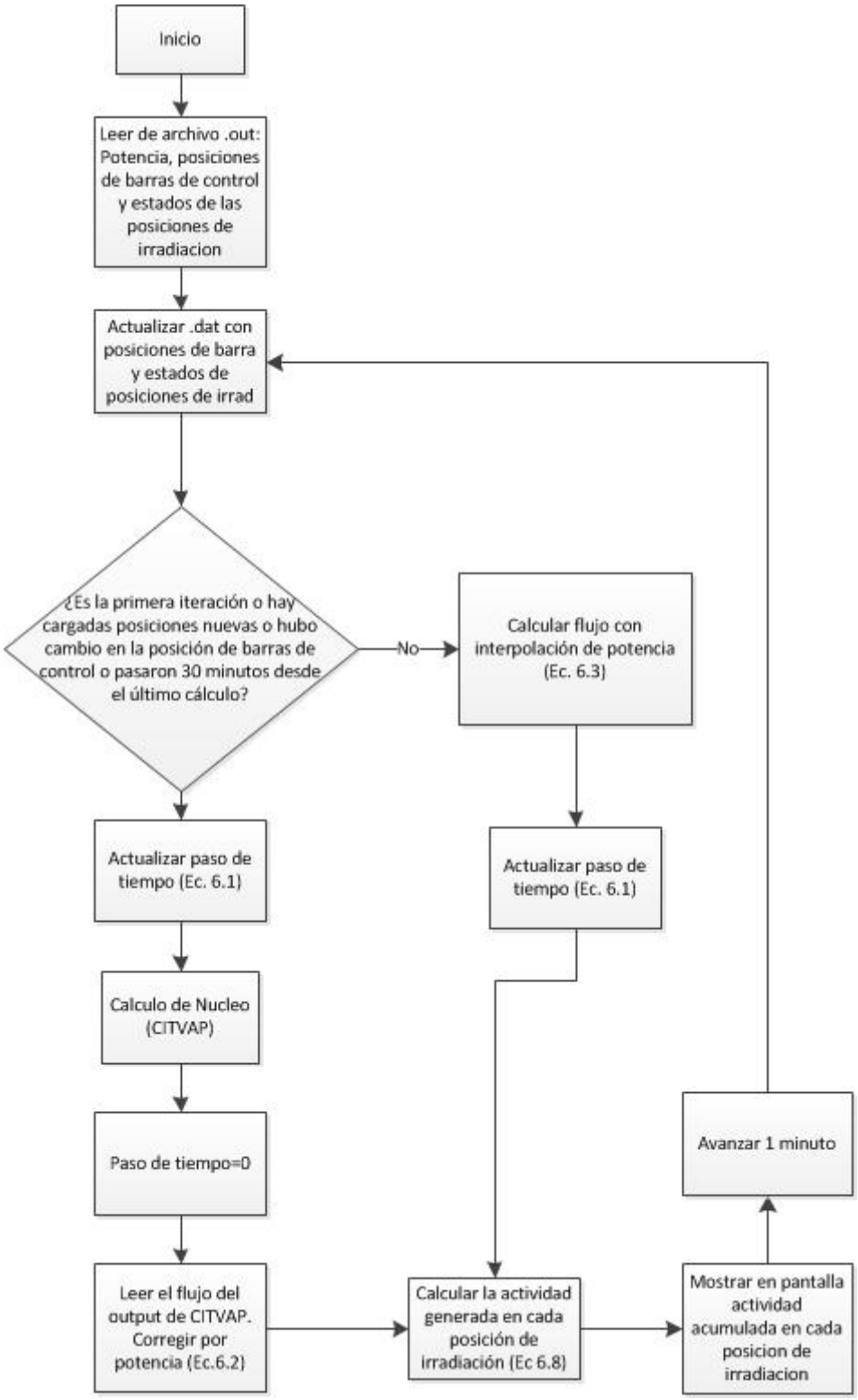

Figura 6.1: Diagrama de flujo del algoritmo del simulador

Se cuenta con un archivo histórico de control de reactor que indica minuto a minuto la posición de barras de control, potencia y estado de posiciones de irradiación. Este se denomina "Dato-Reactor.out". El simulador lee este archivo y minuto a minuto actualiza dichas variables además del paso de tiempo (en días a plena potencia, teniendo en cuenta que el cálculo se realiza a potencia constante de 20 MW) para el cálculo de núcleo en "Variables.dat". Este es utilizado por CITVAP para el cálculo de núcleo junto con un modelo simplificado del OPAL.

Luego de que lee el archivo y actualiza "Variables.dat", el simulador chequea si han sucedido cualquiera de las siguientes situaciones:

- Es la primera iteración.
- Se han cargado nuevas posiciones de irradiación o si se ha movido la barra más de 2 % desde el último cálculo de CITVAP. Se ha decidido poner una "banda muerta" ya que a cada minuto hay variaciones pequeñas en la posición de barras de control causadas por el piloto automático, y dado que dichas variaciones no alteran significativamente el flujo no se justifica realizar un cálculo de núcleo para estas situaciones. De esta manera se economiza el tiempo de cálculo.
- Han pasado más de 30 minutos desde el último cálculo de CITVAP.

Si cualquiera de los ítems mencionados es verdadero, se actualiza el paso de tiempo a:

$$paso_n = paso_{n-1} \cdot \frac{(Potencia_n + Potencia_{n-1})}{40} \cdot \frac{1}{1440} \tag{6.1}$$

A continuación se realiza un cálculo con CITVAP a potencia constante y se lee el flujo del output de CITVAP de la posición de irradiación de interés. El flujo calculado con CITVAP se corrige con el valor de potencia de la siguiente manera:

$$\phi_n = \phi_{\text{CITVAP}} \cdot \frac{Potencia_n}{20} \tag{6.2}$$

A continuación se resetea el paso de tiempo a cero y se calcula la actividad acumulada hasta ese minuto a partir del flujo con el esquema de diferencias finitas que se presenta en la sección siguiente. Finalmente se avanza un minuto y se repite el algoritmo.

Por otro lado, si ninguno de los ítems mencionados anteriormente es verdadero, no se realiza un cálculo de núcleo y solo se hace una interpolación del flujo con la potencia 6.3. Esto se hace para no realizar cálculos de núcleo a cada minuto, economizando tiempo de cálculo. Además si no ha habido movimiento de barras o carga/descarga de posiciones de irradiación, no se espera una variación significativa del flujo, por ello no se considera necesario realizar un cálculo de núcleo minuto a minuto si no han ocurrido cambios en el mismo.

$$\phi_n = \phi_{n-1} \cdot \frac{Potencia_n}{Potencia_{n-1}} \tag{6.3}$$

Además el paso de quemado se calcula como se mencionó anteriormente en la ecuación 6.1.

Con el flujo obtenido minuto a minuto de 6.3 se calcula la actividad acumulada, se avanza un minuto y se repite el algoritmo hasta el tiempo final de simulación indicado.

6.2. Descripción y modelado del OPAL

El reactor **OPAL (Open Pool Australian Lightwater)** es una instalación multipropósito, siendo uno de sus principales objetivos la producción de radioisótopos.

Posee una potencia térmica de 20 MW y tiene un diseño de pileta abierta, lo cual significa que el núcleo está contenido dentro de una pileta abierta de agua desmineralizada que provee refrigeración y protección contra la radiación proveniente del núcleo. El diseño de pileta abierta facilita la carga y descarga de posiciones de irradiación utilizadas para la producción de radioisótopos.

Su núcleo consiste en 16 elementos combustibles del tipo MTR, con 21 placas combustibles de uranio de bajo enriquecimiento.

La reactividad es controlada con 5 barras de control con placas absorbentes, cuatro de las cuales separan el núcleo en 4 cuadrantes iguales, mientras que la quinta tiene forma de cruz y es usada para control fino de potencia debido a su posición central en el núcleo. [18] [19]

6.2.1. Modelado con CITVAP

Se utilizó un modelo simplificado del reactor OPAL que incluye: la fuente fría de neutrones, los tubos de extracción de haces, las posiciones de irradiación neumáticas y las posiciones de irradiación de Silicio.

El objetivo del modelo simplificado fue contar con un modelo de pocas mallas que pudiera correr en menos de un minuto. El modelo simplificado tiene 113 x 118 x 78 mallas para las direcciones X, Y y Z respectivamente.

El modelo contiene el núcleo de 16 combustibles, la chimenea del núcleo, las cinco barras de control y 17 posiciones de irradiación alrededor del núcleo dedicadas a la producción de radioisótopos.

La figura 6.2 muestra un corte transversal del modelo de CITVAP.

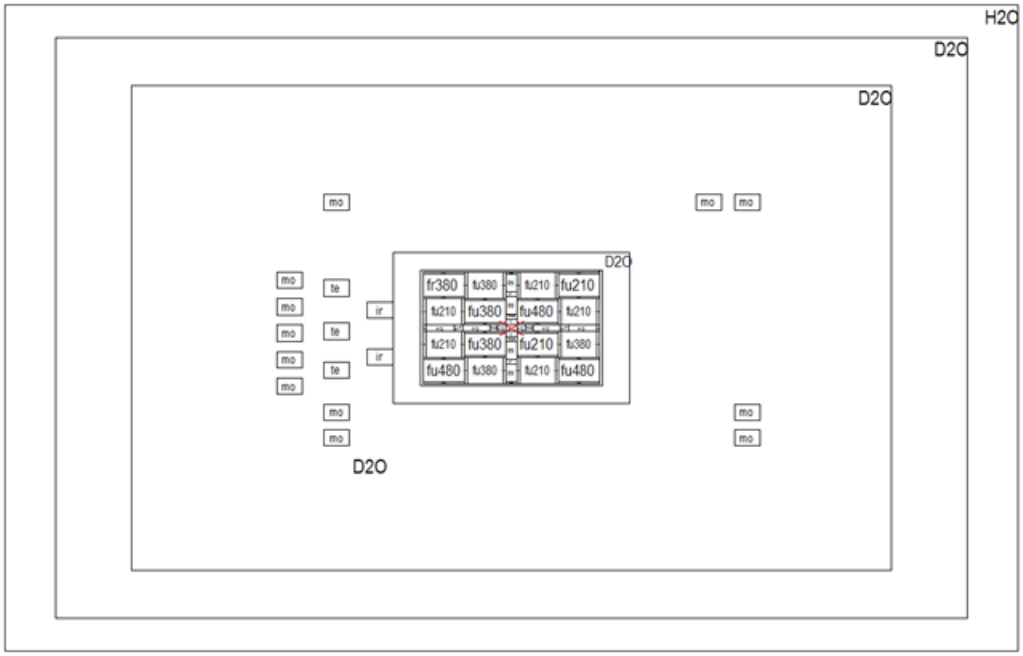

Figura 6.2: Corte transversal del modelo de CITVAP del reactor OPAL

6.3. Cálculo de la actividad por diferencias finitas

El simulador debe determinar la actividad generada minuto a minuto a partir de una lectura de flujo. Es decir que como se tiene una lectura de flujo discretizada en el tiempo se debe proponer un esquema de diferencias finitas para determinar la actividad en cada paso, con la limitación de que no se puede reducir el paso de tiempo a menos de un minuto ya que esto es lo que se demora en hacer un calculo de núcleo con CITVAP en el caso de que hayan fuertes perturbaciones por movimientos de barras de control y/o carga y descarga de posiciones de irradiación.

La ecuación diferencial a discretizar es:

$$\frac{1}{\lambda}\frac{dA}{dt} = R - A \tag{6.4}$$

Su solución analítica es:

$$A(t) = R\left(1 - e^{-\lambda t}\right) \tag{6.5}$$

Por un lado se propuso el siguiente esquema de diferencias finitas, derivado a partir de la ecuación diferencial:

$$\frac{1}{\lambda}\frac{A_{i+1} - A_i}{\Delta t} = R_{i+1} - A_{i+1} \tag{6.6}$$

$$A_{i+1} = \left(R_{i+1} + \frac{A_i}{\lambda \Delta t}\right) \cdot \left(\frac{1}{\lambda \Delta t} + 1\right)^{-1} \tag{6.7}$$

Por otro lado, se propuso el siguiente esquema donde se descompuso la actividad en cada paso como el decaimiento de la actividad en el paso anterior más la actividad

generada en ese paso.

$$A_{i+1} = A_i \cdot e^{-\lambda \Delta t} + R_{i+1} \left(1 - e^{-\lambda \Delta t}\right) \tag{6.8}$$

Se compararon ambos esquemas contra la solución analítica para una irradiación constante de una semana. Al de la ecuación 6.7 se lo denominó esquema 1 y al de la ecuación 6.8 se lo denominó esquema 2. Las actividades obtenidas se muestran en la figura 6.3. Estos valores no fueron afectados por el tiempo de enfriamiento y procesamiento ni por el rendimiento de disolución. En la imagen no se logra distinguir una diferencia entre los esquemas de diferencias finitas y la solución analítica. Por ello, en la tabla 6.1 se muestra la actividad en el séptimo día de irradiación, junto con los errores relativos de cada esquema. Se observa que el esquema 2 tiene un error relativo 8 órdenes de magnitud menor que el esquema 1, por eso se decidió que este sea el esquema de trabajo porque no habrían errores por tener una discretización temporal grande. Se

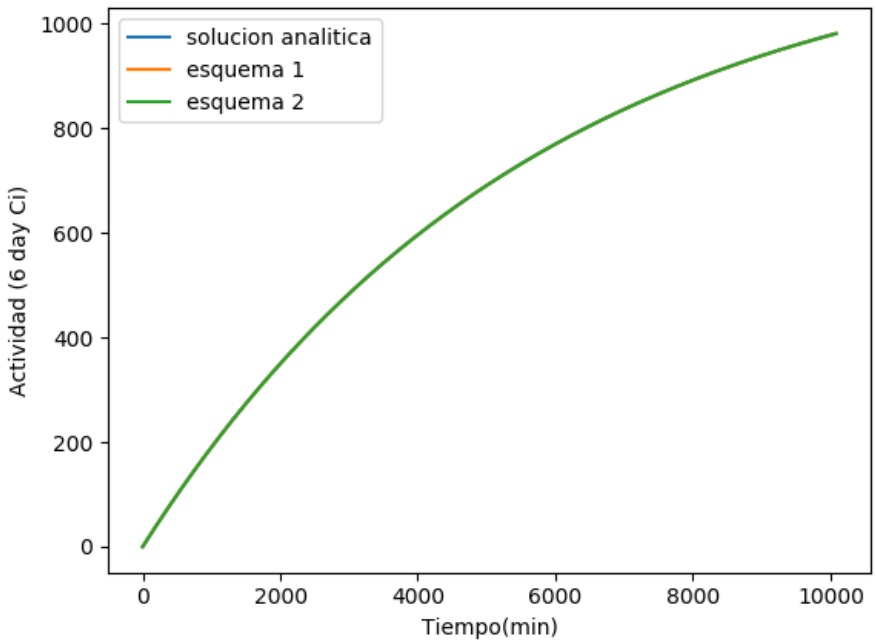

Figura 6.3: Actividad del Mo^{99} generada en el blanco calculada analíticamente y con esquemas de diferencias finitas

Tabla 6.1: Actividad de Mo^{99} en el séptimo día de irradiación calculada analíticamente y con dos esquemas de diferencias finitas

	Analítico	Esquema 1	Esquema 2
Actividad (6 d Ci)	981,18	981,15	981,18
Error relativo	-	-3,19E-05	1,43E-13

corroboró el esquema 2 contra la solución analítica para los casos de parada y cambio

de potencia.

En la figura 6.4 se muestra la actividad generada en función del tiempo para una reducción en la potencia del 10 % en el tercer día de irradiación, calculada con la solución analítica (ecuación 4.26) y con el esquema 2. En la tabla 6.2 se muestra la actividad en el día 7 calculada de ambas maneras, además de el error relativo.

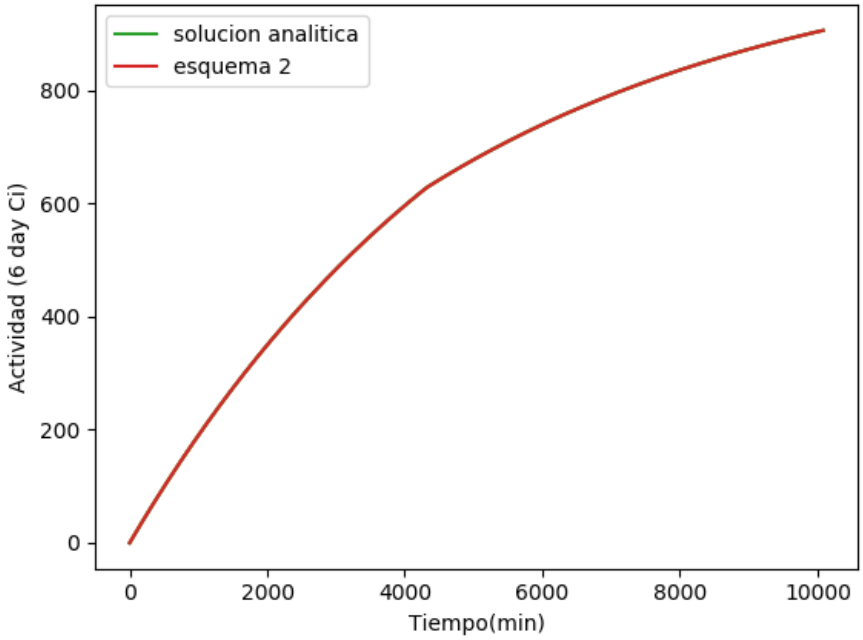

Figura 6.4: Actividad del Mo^{99} generada en el blanco ante un cambio de potencia calculado analíticamente y con un esquema de diferencias finitas

Tabla 6.2: Actividad de Mo^{99} ante un cambio de potencia calculada analíticamente y con un esquema de diferencias finitas

Actividad analítica (6 d Ci)	Actividad esquema 2 (6 dCi)	Error relativo
905,98	905,97	-8,34E-06

En la figura 6.5 se muestra la actividad ante una parada de un día en el día 3 de irradiación, calculada de manera analítica (ecuaciones 6.5, 4.22 y 4.2) y con el esquema de diferencias finitas 2.

En la tabla 6.3 se muestra una comparación de la actividad en el día 7.

Tabla 6.3: Actividad de Mo^{99} generada en el blanco ante una parada calculada analíticamente y con un esquema de diferencias finitas

Actividad analítica (6 d Ci)	Actividad esquema 2 (6 dCi)	Error relativo
857,35	857,37	2,53E-05

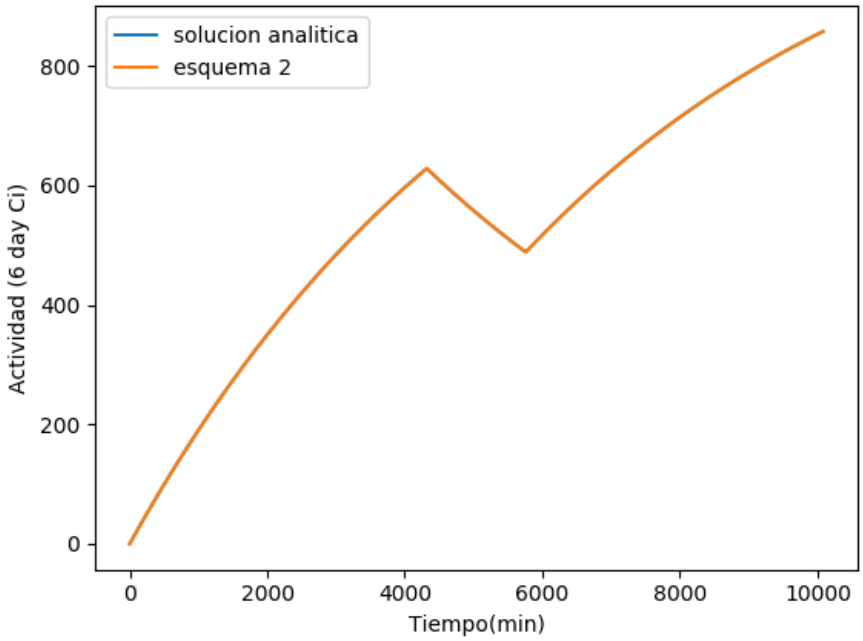

Figura 6.5: Actividad del Mo^{99} generada en el blanco ante una parada calculada analíticamente y con un esquema de diferencias finitas

6.4. Resultados

A partir de un conjunto de datos históricos de posiciones de barra, posiciones de irradiación y potencia del reactor, se realizó una simulación de 13893 minutos (9,65 días). La actividad y flujo obtenidos se grafican en la figura 6.6. Se puede observar que la actividad en función del tiempo es similar al gráfico de su expresión analítica como fue mostrado en la sección anterior.

En primer lugar se debe remarcar que el tiempo de cálculo del simulador es menor que un minuto (en los casos en los que se debe realizar cálculo de núcleo y aún menor en los casos en los que no), por lo tanto en una futura implementación del mismo se podría ejecutar en tiempo real junto con el reactor.

De la figura 6.6 es posible concluir que es válido realizar la interpolación del flujo con la potencia, ya que de lo contrario se vería una desviación respecto del flujo medio que se corregiría cada 30 minutos. De la misma manera se puede concluir que la elección del valor de la "banda muerta" en la posición de las barras de control es correcta, ya que si variaciones pequeñas causaran grandes cambios en el flujo, estos se verían reflejados en el gráfico del flujo como grandes variaciones de flujo que se corrigen cada 30 minutos.

Para esta simulación de 9,65 días se realizaron 473 cálculos con CITVAP, los cuales en su mayoría se deben a los cálculos de chequeo cada 30 minutos.

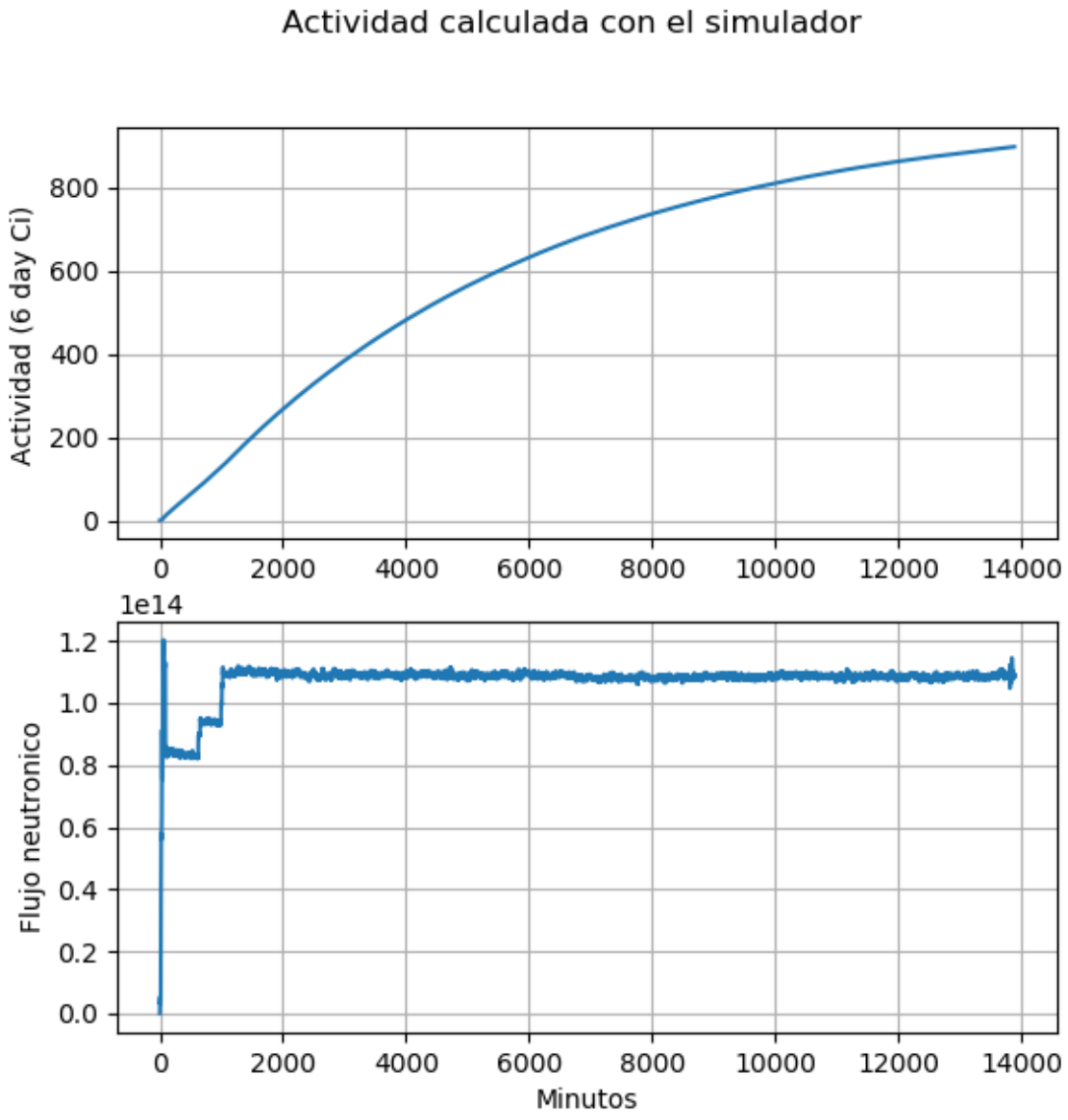

Figura 6.6: Actividad generada y flujo neutrónico de una posición de irradiación calculados con el simulador

En la figura 6.7 se observa la evolución de la potencia durante la simulación, se observa que es proporcional al flujo. Adicionalmente, se muestra la evolución de la posición de las barras de control en la figura 6.8.

En la figura 6.9 se muestra la actividad obtenida del simulador afectada por la eficiencia del proceso de disolución, donde se observa un máximo en el día 7,77.

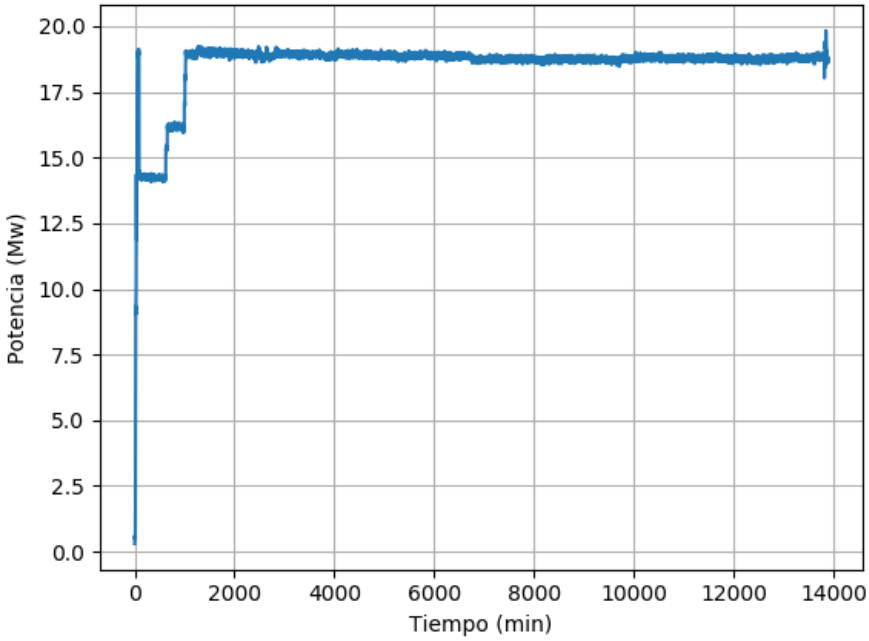

Figura 6.7: Evolución de la potencia durante la simulación

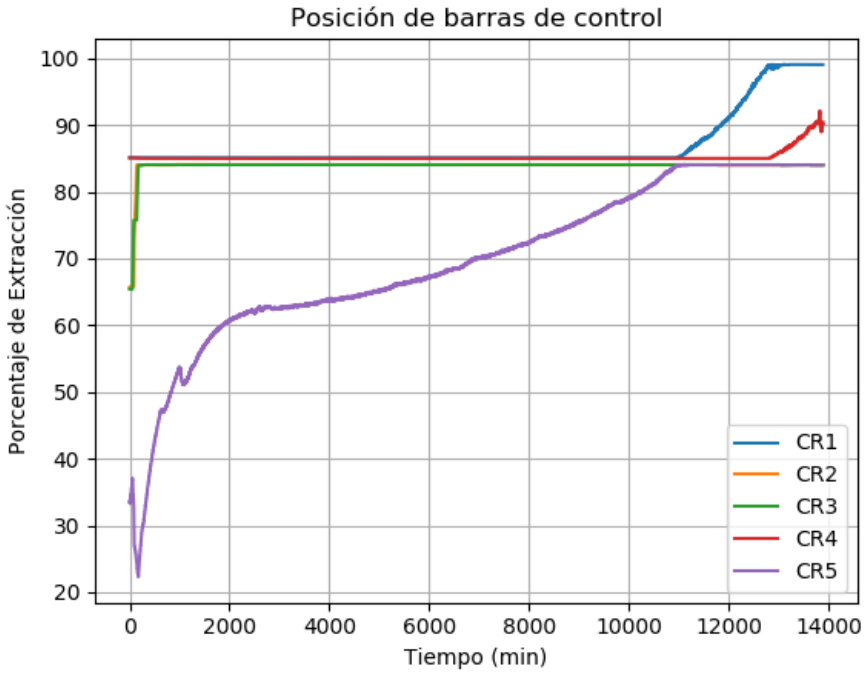

Figura 6.8: Evolución de la posición de las barras de control durante la simulación

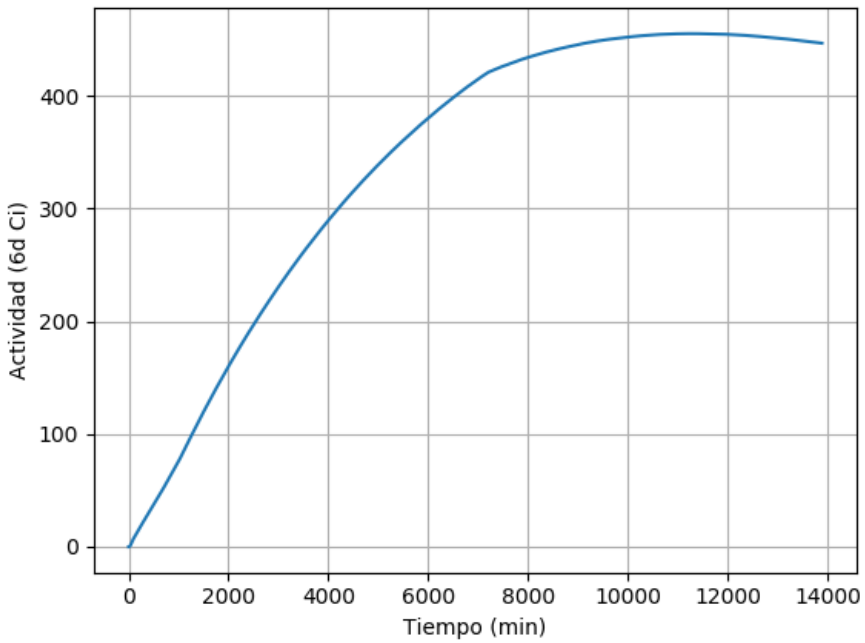

Figura 6.9: Actividad calculada por el simulador afectada por la eficiencia del proceso de disolución

Capítulo 7

Conclusiones

7.1. Planificador

Se diseñó e implementó un algoritmo planificador que a partir de las ecuaciones de activación de cada radioisótopo, el flujo y cantidad de las posiciones de irradiación, la cantidad demandada y la programación de potencia y paradas del reactor, calcula el tiempo de irradiación e indica cuántas posiciones de irradiación ocupar y por cuánto tiempo. Se identificaron diversas situaciones tales como cambios de potencia, e interrupción de la irradiación y reanudación de la misma con igual o distinta potencia, y se estudió el impacto de factores como la duración de la parada o el momento de ocurrencia de la misma en la actividad generada. El planificador se implementó para Mo^{99} y Lu^{177} producido por vía directa, dos radioisótopos que tienen evoluciones de la actividad diferentes al irradiarse. La actividad del Mo^{99} llega a un máximo y decrece, debido a los rendimientos del proceso de disolución decrecientes con el tiempo, mientras que la actividad del Lu^{177} crece asintóticamente. Sin embargo, teniendo en cuenta estas diferencias, el algoritmo y la metodología de cálculo de los tiempos de irradiación son los mismos. Por ello, se podría extender fácilmente el planificador para abarcar más isótopos con el mismo algoritmo.

7.2. Simulador

Se diseñó e implementó un algoritmo simulador que calcula la actividad generada de Mo^{99} minuto a minuto. El simulador toma datos históricos de posición de barras de control, estado de posiciones de irradiación, y potencia y si es necesario, realiza un cálculo de núcleo (con CTIVAP y un modelo simplificado del OPAL) para determinar el flujo en una determinada posición de irradiación y con el mismo la actividad generada minuto a minuto. Para economizar tiempo de cálculo, solo se realiza cálculo de núcleo si es la primera iteración, o si se han cargado/descargado posiciones de irradiación desde

Biblioteca Leo Falicov CAB-IB

el último cálculo o si alguna de las barras de control se ha movido más del 2 % desde el último cálculo o han pasado mas de 30 minutos desde el último cálculo de CITVAP. Si no ha sucedido ninguno de estos casos, no se realiza cálculo de núcleo y solo se calcula el flujo nuevo con una interpolación.

Se probó el simulador con un archivo histórico de 9,65 días de posición de barras de control, estado de posiciones de irradiación y potencia. Se corroboró su correcto funcionamiento, es decir que actualice las posiciones de barras de control, estado de las posiciones de irradiación, y paso de tiempo minuto a minuto, y que realice cálculo de núcleo en las situaciones mencionadas anteriormente. Se obtuvo una curva de actividad en función del tiempo similar a la analítica. Se corroboró la validez de trabajar con una "banda muerta" en la posición de barras de control de 2 % y un tiempo de chequeo cada 30 minutos. El simulador corre en menos de un minuto en cada paso, por lo que sería posible utilizarlo para una implementación futura en tiempo real durante la operación.

7.3. Trabajo futuro

Para el planificador, un trabajo futuro es agregar el Lu^{177} por vía indirecta, y el Ir^{192}. Además para el caso del Mo^{99} se debería poder modificar la cantidad de placas por rig de irradiación. Otra alternativa es acoplar propiedades neutrónicas con las salida de CONDOR-CITVAP. Adicionalmente se podrían considerar las limitaciones de la celda caliente, es decir que no puedan extraerse muchas posiciones de irradiación en forma simultánea del reactor, ya que se produciría un cuello de botella en la celda caliente. Es decir, el planificador debería escalonar los tiempos de extracción según la capacidad de procesamiento de la celda caliente.

Para el simulador, un trabajo futuro posible es que en lugar de leer datos de un archivo histórico, lo haga desde el sistema de control del reactor para así poder ejecutarlo en tiempo real y determinar la actividad generada en cada posición de irradiación minuto a minuto. Además se podría extender a otros radioisótopos, particularmente a Lu^{177} por ambas vías de producción, e Ir^{192}.

Apéndice A

Simulador

```
import os
import numpy as np
import matplotlib.pyplot as plt
Delta_t=1#min
lambda_mo=np.log(2)/(65.94*60)#en minutos
yield_mo=0.0613
masa_u5=1.4
sigma_fision_u5=5.8E-22
enriquecimiento=0.1975
largo=11.7
alto=3
ancho=0.07
dens_u=masa_u5/(largo*alto*ancho)/enriquecimiento
cant_placas=12
command1="pre_cit.exe␣ciclo-03.cii"
command2="citvap.exe␣ciclo-03.cii"
#la posicion cargada es 3201
tiempo=13893#tiempo final de simulacion en minutos
#%% Funcion que lee el flujo de cierta posicion de irradiacion
def leer_flujo(archivo,nombre_pos_irrad):
    f = open(archivo, "r")
    x=f.read()
    substring="␣ZONE␣␣␣"+str(nombre_pos_irrad)
    y=x.find(substring)
    z=x.find("\n",y,len(x))
    b=""
    if(z!=1):
        for j in range(y+len(substring)+1,z):
            b=b+x[j]
    if(z==-1):
        for j in range(y+len(substring)+1,len(x)):
            b=b+x[j]
    f.close()
    flujo_th=""
    for j in range (len(b)-11,len(b)):
        flujo_th=flujo_th+b[j]
    flujo_th=float(flujo_th)
    return flujo_th
#%%   lectura del .out
with open("Dato-reactor1.out") as f:
    lines = f.readlines()
substring1="␣"
substring2="␣␣"
y=np.zeros(60)
y[0]=lines[5].find(substring1,0,len(lines[5]))
for i in range (1,len(y)):
    y[i]=lines[5].find(substring1,int(y[i-1]+1),len(lines[5]))
x=np.zeros([len(lines),60])
for j in range (1,len(lines)):
    x[j][0]=y[14]
    for i in range (1,60):
        x[j][i]=lines[j].find(substring2,int(x[j][i-1]+1),len(lines[j]))

CR1=np.empty(len(lines)-1)
CR2=np.empty(len(lines)-1)
CR3=np.empty(len(lines)-1)
```

```
CR4=np.empty(len(lines)-1)
CR5=np.empty(len(lines)-1)
POWER=np.empty(len(lines)-1)
cr1=""
cr2=""
cr3=""
cr4=""
cr5=""
power=""
HF1=np.full(len(lines)-1,"Empty")
HF2=np.full(len(lines)-1,"Empty")
MF1=np.full(len(lines)-1,"Empty")
MF2=np.full(len(lines)-1,"Empty")
MF3=np.full(len(lines)-1,"Empty")
LF1=np.full(len(lines)-1,"Empty")
LF2=np.full(len(lines)-1,"Empty")
LF3=np.full(len(lines)-1,"Empty")
LF4=np.full(len(lines)-1,"Empty")
LF5=np.full(len(lines)-1,"Empty")
LF6=np.full(len(lines)-1,"Empty")
LF7=np.full(len(lines)-1,"Empty")
LF8=np.full(len(lines)-1,"Empty")
LF9=np.full(len(lines)-1,"Empty")
LF10=np.full(len(lines)-1,"Empty")
LF11=np.full(len(lines)-1,"Empty")
LF12=np.full(len(lines)-1,"Empty")
LF13=np.full(len(lines)-1,"Empty")
hf1=""
hf2=""
mf1=""
mf2=""
mf3=""
lf1=""
lf2=""
lf3=""
lf4=""
lf5=""
lf6=""
lf7=""
lf8=""
lf9=""
lf10=""
lf11=""
lf12=""
lf13=""

for j in range (1,len(lines)):
    for i in range(int(y[5]),int(y[6])):
        cr1=cr1+lines[j][i]
    CR1[j-1]=cr1
    cr1=""

    for i in range (int(y[6]),int(y[7])):
        cr2=cr2+lines[j][i]
    CR2[j-1]=cr2
    cr2=""

    for i in range (int(y[7]),int(y[8])):
        cr3=cr3+lines[j][i]
    CR3[j-1]=cr3
    cr3=""
    for i in range (int(y[8]),int(y[9])):
        cr4=cr4+lines[j][i]
    CR4[j-1]=cr4
    cr4=""
    for i in range(int(y[9]),int(y[10])):
        cr5=cr5+lines[j][i]
    CR5[j-1]=cr5
    cr5=""

    for i in range(int(y[12]+1),int(y[13])):
        power=power+lines[j][i]
    POWER[j-1]=power
    power=""

    for i in range(int(x[j][0]+2),int(x[j][1])):
        hf1=hf1+lines[j][i]
    HF1[j-1]=hf1
    for i in range(int(x[j][1]+2),int(x[j][2])):
        hf2=hf2+lines[j][i]
```

```
    HF2[j-1]=hf2
    hf2=""
    for i in range(int(x[j][2]+2),int(x[j][3])):
        mf1=mf1+lines[j][i]
    MF1[j-1]=mf1
    mf1=""

    for i in range(int(x[j][3]+2),int(x[j][4])):
        mf2=mf2+lines[j][i]
    MF2[j-1]=mf2
    mf2=""
    for i in range(int(x[j][4]+2),int(x[j][5])):
        mf3=mf3+lines[j][i]
    MF3[j-1]=mf3
    mf3=""
    for i in range(int(x[j][5]+2),int(x[j][6])):
        lf1=lf1+lines[j][i]
    LF1[j-1]=lf1
    lf1=""
    for i in range(int(x[j][6]+2),int(x[j][7])):
        lf2=lf2+lines[j][i]
    LF2[j-1]=lf2
    lf2=""
    for i in range(int(x[j][7]+2),int(x[j][8])):
        lf3=lf3+lines[j][i]
    LF3[j-1]=lf3
    lf3=""
    for i in range(int(x[j][8]+2),int(x[j][9])):
        lf4=lf4+lines[j][i]
    LF4[j-1]=lf4
    lf4=""
    for i in range(int(x[j][9]+2),int(x[j][10])):
        lf5=lf5+lines[j][i]
    LF5[j-1]=lf5
    lf5=""
    for i in range(int(x[j][10]+2),int(x[j][11])):
        lf6=lf6+lines[j][i]
    LF6[j-1]=lf6
    lf6=""
    for i in range(int(x[j][11]+2),int(x[j][12])):
        lf7=lf7+lines[j][i]
    LF7[j-1]=lf7
    lf7=""
    for i in range(int(x[j][12]+2),int(x[j][13])):
        lf8=lf8+lines[j][i]
    LF8[j-1]=lf8
    lf8=""
    for i in range(int(x[j][13]+2),int(x[j][14])):
        lf9=lf9+lines[j][i]
    LF9[j-1]=lf9
    lf9=""
    for i in range(int(x[j][14]+2),int(x[j][15])):
        lf10=lf10+lines[j][i]
    LF10[j-1]=lf10
    lf10=""
    for i in range(int(x[j][15]+2),int(x[j][16])):
        lf11=lf11+lines[j][i]
    LF11[j-1]=lf11
    lf11=""
    for i in range(int(x[j][16]+2),int(x[j][17])):
        lf12=lf12+lines[j][i]
    LF12[j-1]=lf12
    lf12=""
    for i in range(int(x[j][17]+2),len(lines[j])):
        lf13=lf13+lines[j][i]
    LF13[j-1]=lf13
    lf13=""


barras_control=np.zeros([len(lines)-1,5])
barras_control[:,0]=CR1
barras_control[:,1]=CR2
barras_control[:,2]=CR3
barras_control[:,3]=CR4
barras_control[:,4]=CR5
posiciones_irrad=np.full([len(lines)-1,18],"␣␣␣␣␣␣")
posiciones_irrad[:,0]=HF1
posiciones_irrad[:,1]=HF2
posiciones_irrad[:,2]=MF1
posiciones_irrad[:,3]=MF2
```

```
posiciones_irrad[:,4]=MF3
posiciones_irrad[:,5]=LF1
posiciones_irrad[:,6]=LF2
posiciones_irrad[:,7]=LF3
posiciones_irrad[:,8]=LF4
posiciones_irrad[:,9]=LF5
posiciones_irrad[:,10]=LF6
posiciones_irrad[:,11]=LF7
posiciones_irrad[:,12]=LF8
posiciones_irrad[:,13]=LF9
posiciones_irrad[:,14]=LF10
posiciones_irrad[:,15]=LF11
posiciones_irrad[:,16]=LF12
posiciones_irrad[:,17]=LF13


#%% Funcion que chequea si alguna barra de control se ha movido mas
#de 2% o si se ha cargada/descargado una posicion de irradiacion
def check_barrascontrol(barras_control,t1,t2):
    if(abs(CR1[t1]-CR1[t2])>=2 or abs(CR2[t1]-CR2[t2])>=2 or abs(CR3[t1]-CR3[t2])>=2 or abs(CR4[t1]-CR4[t2])>=2 or
         abs(CR5[t1]-CR5[t2])>=2):
        return True
    else:
        return False

def check_posiciones_irrad(posiciones_irrad,t1,t2):
    a=False
    b=False
    c=False
    d=False
    if(posiciones_irrad[t1,0]!=posiciones_irrad[t2,0] or posiciones_irrad[t1,1]!=posiciones_irrad[t2,1] or
         posiciones_irrad[t1,2]!=posiciones_irrad[t2,2] or posiciones_irrad[t1,3]!=posiciones_irrad[t2,3] or
         posiciones_irrad[t1,4]!=posiciones_irrad[t2,4]):
        a=True
    if(posiciones_irrad[t1,5]!=posiciones_irrad[t2,5] or posiciones_irrad[t1,6]!=posiciones_irrad[t2,6] or
         posiciones_irrad[t1,7]!=posiciones_irrad[t2,7] or posiciones_irrad[t1,8]!=posiciones_irrad[t2,8] or
         posiciones_irrad[t1,9]!=posiciones_irrad[t2,9]):
        b=True
    if(posiciones_irrad[t1,10]!=posiciones_irrad[t2,10] or posiciones_irrad[t1,11]!=posiciones_irrad[t2,11] or
         posiciones_irrad[t1,12]!=posiciones_irrad[t2,12] or posiciones_irrad[t1,13]!=posiciones_irrad[t2,13] or
         posiciones_irrad[t1,14]!=posiciones_irrad[t2,14]):
        c=True
    if(posiciones_irrad[t1,15]!=posiciones_irrad[t2,15] or posiciones_irrad[t1,16]!=posiciones_irrad[t2,16] or
         posiciones_irrad[t1,17]!=posiciones_irrad[t2,17]):
        d=True
    if(a or b or c or d):
        return True
    else:
        return False
#%% Funcion para actualizar posicion de barras de control
def check_barra(barra,t1,t2):
    if(barra[t1]!=barra[t2]):
        return True
    else:
        return False
def actualizar_barra1(CR1,t):
    with open("Variables.dat") as f:
        variables = f.readlines()
    with open("Variables.dat",'w') as f:
            for i,line in enumerate(variables,1):
                if i == 22:
                    f.writelines("^^ MOVE CONTROL ROD [S1] "+str(CR1[t])+"\n")
                else:
                    f.writelines(line)
def actualizar_barra2(CR2,t):
    with open("Variables.dat") as f:
        variables = f.readlines()
    with open("Variables.dat",'w') as f:
            for i,line in enumerate(variables,1):
                if i == 23:
                    f.writelines("^^ MOVE CONTROL ROD [S2] "+str(CR2[t])+"\n")
                else:
                    f.writelines(line)
def actualizar_barra3(CR3,t):
    with open("Variables.dat") as f:
        variables = f.readlines()
    with open("Variables.dat",'w') as f:
            for i,line in enumerate(variables,1):
                if i == 24:
                    f.writelines("^^ MOVE CONTROL ROD [S3] "+str(CR3[t])+"\n")
```

```
                else:
                    f.writelines(line)

def actualizar_barra4(CR4,t):
    with open("Variables.dat") as f:
        variables = f.readlines()
    with open("Variables.dat",'w') as f:
            for i,line in enumerate(variables,1):
                if i == 25:
                    f.writelines("^^ MOVE CONTROL ROD [S4] "+str(CR4[t])+"\n")
                else:
                    f.writelines(line)
def actualizar_barra5(CR5,t):
    with open("Variables.dat") as f:
        variables = f.readlines()
    with open("Variables.dat",'w') as f:
            for i,line in enumerate(variables,1):
                if i == 26:
                    f.writelines("^^ MOVE CONTROL ROD [S5] "+str(CR5[t])+"\n")
                else:
                    f.writelines(line)
def actualizar_barras(CR1,CR2,CR3,CR4,CR5,t):
    actualizar_barra1(CR1,t)
    actualizar_barra2(CR2,t)
    actualizar_barra3(CR3,t)
    actualizar_barra4(CR4,t)
    actualizar_barra5(CR5,t)
#%%  Funcion para actualizar posiciones de irradiaci n
def check_pos(pos,t1,t2):
    if(pos[t1]!=pos[t2]):
        return True

def actualizar_pos_irrad1(pos_irrad,t):
    with open("Variables.dat") as f:
        variables = f.readlines()
    if(pos_irrad[t,0]!="Empty"):
        with open("Variables.dat",'w') as f:
            for i,line in enumerate(variables,1):
                if i == 2:
                    f.writelines("^^ CHANGE MATERIAL 3001 PER 2\n")
                else:
                    f.writelines(line)
    else:
        with open("Variables.dat",'w') as f:
            for i,line in enumerate(variables,1):
                if i == 2:
                    f.writelines("^^ CHANGE MATERIAL 3001 PER 1\n")
                else:
                    f.writelines(line)
def actualizar_pos_irrad2(pos_irrad,t):
    with open("Variables.dat") as f:
        variables = f.readlines()
    if(pos_irrad[t,1]!="Empty"):
        with open("Variables.dat",'w') as f:
            for i,line in enumerate(variables,1):
                if i == 3:
                    f.writelines("^^ CHANGE MATERIAL 3002 PER 2\n")
                else:
                    f.writelines(line)
    else:
        with open("Variables.dat",'w') as f:
            for i,line in enumerate(variables,1):
                if i == 2:
                    f.writelines("^^ CHANGE MATERIAL 3002 PER 1\n")
                else:
                    f.writelines(line)
def actualizar_pos_irrad3(pos_irrad,t):
    with open("Variables.dat") as f:
          variables = f.readlines()
    if(pos_irrad[t,2]!="Empty"):
        with open("Variables.dat",'w') as f:
            for i,line in enumerate(variables,1):
                if i == 5:
                    f.writelines("^^ CHANGE MATERIAL 3101 PER 2\n")
                else:
                    f.writelines(line)
    else:
        with open("Variables.dat",'w') as f:
            for i,line in enumerate(variables,1):
                if i == 5:
```

```
                    f.writelines("^^␣CHANGE␣MATERIAL␣3101␣PER␣1\n")
                else:
                    f.writelines(line)
def actualizar_pos_irrad4(pos_irrad,t):
    with open("Variables.dat") as f:
          variables = f.readlines()
    if(pos_irrad[t,3]!="Empty"):
        with open("Variables.dat",'w') as f:
            for i,line in enumerate(variables,1):
                if i == 6:
                    f.writelines("^^␣CHANGE␣MATERIAL␣3102␣PER␣2\n")
                else:
                    f.writelines(line)
    else:
        with open("Variables.dat",'w') as f:
            for i,line in enumerate(variables,1):
                if i == 6:
                    f.writelines("^^␣CHANGE␣MATERIAL␣3102␣PER␣1\n")
                else:
                    f.writelines(line)
def actualizar_pos_irrad5(pos_irrad,t):
    with open("Variables.dat") as f:
          variables = f.readlines()
    if(pos_irrad[t,4]!="Empty"):
        with open("Variables.dat",'w') as f:
            for i,line in enumerate(variables,1):
                if i == 7:
                    f.writelines("^^␣CHANGE␣MATERIAL␣3103␣PER␣2\n")
                else:
                    f.writelines(line)
    else:
        with open("Variables.dat",'w') as f:
            for i,line in enumerate(variables,1):
                if i == 7:
                    f.writelines("^^␣CHANGE␣MATERIAL␣3103␣PER␣1\n")
                else:
                    f.writelines(line)
def actualizar_pos_irrad6(pos_irrad,t):
    with open("Variables.dat") as f:
          variables = f.readlines()
    if(pos_irrad[t,5]!="Empty"):
        with open("Variables.dat",'w') as f:
            for i,line in enumerate(variables,1):
                if i == 9:
                    f.writelines("^^␣CHANGE␣MATERIAL␣3201␣PER␣2\n")
                else:
                    f.writelines(line)
    else:
        with open("Variables.dat",'w') as f:
            for i,line in enumerate(variables,1):
                if i == 9:
                    f.writelines("^^␣CHANGE␣MATERIAL␣3201␣PER␣1\n")
                else:
                    f.writelines(line)
def actualizar_pos_irrad7(pos_irrad,t):
    with open("Variables.dat") as f:
          variables = f.readlines()
    if(pos_irrad[t,6]!="Empty"):
        with open("Variables.dat",'w') as f:
            for i,line in enumerate(variables,1):
                if i == 10:
                    f.writelines("^^␣CHANGE␣MATERIAL␣3202␣PER␣2\n")
                else:
                    f.writelines(line)
    else:
         with open("Variables.dat",'w') as f:
            for i,line in enumerate(variables,1):
                if i == 10:
                    f.writelines("^^␣CHANGE␣MATERIAL␣3202␣PER␣1\n")
                else:
                    f.writelines(line)
def actualizar_pos_irrad8(pos_irrad,t):
    with open("Variables.dat") as f:
          variables = f.readlines()
    if(pos_irrad[t,7]!="Empty"):
        with open("Variables.dat",'w') as f:
            for i,line in enumerate(variables,1):
                if i == 11:
                    f.writelines("^^␣CHANGE␣MATERIAL␣3203␣PER␣2\n")
                else:
```

```
                    f.writelines(line)
    else:
        with open("Variables.dat",'w') as f:
            for i,line in enumerate(variables,1):
                if i == 11:
                    f.writelines("^^ CHANGE MATERIAL 3203 PER 1\n")
                else:
                    f.writelines(line)
def actualizar_pos_irrad9(pos_irrad,t):
    with open("Variables.dat") as f:
          variables = f.readlines()
    if(pos_irrad[t,8]!="Empty"):
        with open("Variables.dat",'w') as f:
            for i,line in enumerate(variables,1):
                if i == 12:
                    f.writelines("^^ CHANGE MATERIAL 3204 PER 2\n")
                else:
                    f.writelines(line)
    else:
        with open("Variables.dat",'w') as f:
            for i,line in enumerate(variables,1):
                if i == 12:
                    f.writelines("^^ CHANGE MATERIAL 3204 PER 1\n")
                else:
                    f.writelines(line)
def actualizar_pos_irrad10(pos_irrad,t):
    with open("Variables.dat") as f:
          variables = f.readlines()
    if(pos_irrad[t,9]!="Empty"):
        with open("Variables.dat",'w') as f:
            for i,line in enumerate(variables,1):
                if i == 13:
                    f.writelines("^^ CHANGE MATERIAL 3205 PER 2\n")
                else:
                    f.writelines(line)
    else:
        with open("Variables.dat",'w') as f:
            for i,line in enumerate(variables,1):
                if i == 13:
                    f.writelines("^^ CHANGE MATERIAL 3205 PER 1\n")
                else:
                    f.writelines(line)
def actualizar_pos_irrad11(pos_irrad,t):
    with open("Variables.dat") as f:
          variables = f.readlines()
    if(pos_irrad[t,10]!="Empty"):
        with open("Variables.dat",'w') as f:
            for i,line in enumerate(variables,1):
                if i == 14:
                    f.writelines("^^ CHANGE MATERIAL 3206 PER 2\n")
                else:
                    f.writelines(line)
    else:
        with open("Variables.dat",'w') as f:
            for i,line in enumerate(variables,1):
                if i == 14:
                    f.writelines("^^ CHANGE MATERIAL 3206 PER 1\n")
                else:
                    f.writelines(line)
def actualizar_pos_irrad12(pos_irrad,t):
    with open("Variables.dat") as f:
          variables = f.readlines()
    if(pos_irrad[t,11]!="Empty"):
        with open("Variables.dat",'w') as f:
            for i,line in enumerate(variables,1):
                if i == 15:
                    f.writelines("^^ CHANGE MATERIAL 3207 PER 2\n")
                else:
                    f.writelines(line)
    else:
        with open("Variables.dat",'w') as f:
            for i,line in enumerate(variables,1):
                if i == 15:
                    f.writelines("^^ CHANGE MATERIAL 3207 PER 1\n")
                else:
                    f.writelines(line)
def actualizar_pos_irrad13(pos_irrad,t):
    with open("Variables.dat") as f:
          variables = f.readlines()
    if(pos_irrad[t,12]!="Empty"):
```

```
        with open("Variables.dat",'w') as f:
            for i,line in enumerate(variables,1):
                if i == 16:
                    f.writelines("^^ CHANGE MATERIAL 3208 PER 2\n")
                else:
                    f.writelines(line)
    with open("Variables.dat",'w') as f:
            for i,line in enumerate(variables,1):
                if i == 16:
                    f.writelines("^^ CHANGE MATERIAL 3208 PER 1\n")
                else:
                    f.writelines(line)
def actualizar_pos_irrad14(pos_irrad,t):
    with open("Variables.dat") as f:
          variables = f.readlines()
    if(pos_irrad[t,13]!="Empty"):
        with open("Variables.dat",'w') as f:
            for i,line in enumerate(variables,1):
                if i == 17:
                    f.writelines("^^ CHANGE MATERIAL 3209 PER 2\n")
                else:
                    f.writelines(line)
    else:
         with open("Variables.dat",'w') as f:
            for i,line in enumerate(variables,1):
                if i == 17:
                    f.writelines("^^ CHANGE MATERIAL 3209 PER 1\n")
                else:
                    f.writelines(line)
def actualizar_pos_irrad15(pos_irrad,t):
    with open("Variables.dat") as f:
          variables = f.readlines()
    if(pos_irrad[t,14]!="Empty"):
        with open("Variables.dat",'w') as f:
            for i,line in enumerate(variables,1):
                if i == 18:
                    f.writelines("^^ CHANGE MATERIAL 3210 PER 2\n")
                else:
                    f.writelines(line)
    else:
        with open("Variables.dat",'w') as f:
            for i,line in enumerate(variables,1):
                if i == 18:
                    f.writelines("^^ CHANGE MATERIAL 3210 PER 1\n")
                else:
                    f.writelines(line)
def actualizar_pos_irrad16(pos_irrad,t):
    with open("Variables.dat") as f:
          variables = f.readlines()
    if(pos_irrad[t,15]!="Empty"):
        with open("Variables.dat",'w') as f:
            for i,line in enumerate(variables,1):
                if i == 19:
                    f.writelines("^^ CHANGE MATERIAL 3211 PER 2\n")
                else:
                    f.writelines(line)
    else:
        with open("Variables.dat",'w') as f:
            for i,line in enumerate(variables,1):
                if i == 19:
                    f.writelines("^^ CHANGE MATERIAL 3211 PER 1\n")
                else:
                    f.writelines(line)
def actualizar_pos_irrad17(pos_irrad,t):
    with open("Variables.dat") as f:
          variables = f.readlines()
    if(pos_irrad[t,16]!="Empty"):
        with open("Variables.dat",'w') as f:
            for i,line in enumerate(variables,1):
                if i == 20:
                    f.writelines("^^ CHANGE MATERIAL 3212 PER 2\n")
                else:
                    f.writelines(line)
    else:
        with open("Variables.dat",'w') as f:
            for i,line in enumerate(variables,1):
                if i == 20:
                    f.writelines("^^ CHANGE MATERIAL 3212 PER 1\n")
                else:
                    f.writelines(line)
```

```
def actualizar_posiciones_irrad(pos_irrad,t):
    actualizar_pos_irrad1(pos_irrad,t)
    actualizar_pos_irrad2(pos_irrad,t)
    actualizar_pos_irrad3(pos_irrad,t)
    actualizar_pos_irrad4(pos_irrad,t)
    actualizar_pos_irrad5(pos_irrad,t)
    actualizar_pos_irrad6(pos_irrad,t)
    actualizar_pos_irrad7(pos_irrad,t)
    actualizar_pos_irrad8(pos_irrad,t)
    actualizar_pos_irrad9(pos_irrad,t)
    actualizar_pos_irrad10(pos_irrad,t)
    actualizar_pos_irrad11(pos_irrad,t)
    actualizar_pos_irrad12(pos_irrad,t)
    actualizar_pos_irrad13(pos_irrad,t)
    actualizar_pos_irrad14(pos_irrad,t)
    actualizar_pos_irrad15(pos_irrad,t)
    actualizar_pos_irrad16(pos_irrad,t)
    actualizar_pos_irrad17(pos_irrad,t)

# Funcion que actualiza el paso de tiempo de c lculo
def actualizar_pasobu(POWER,pasobu,t1,t2):
    with open("Variables.dat") as f:
        variables = f.readlines()
    with open("Variables.dat",'w') as f:
        for i,line in enumerate(variables,1):
                if i == 28:
                    f.writelines("^^ pseudovar PASOBU = "+str(pasobu))
                else:
                    f.writelines(line)



#%% Simulador
phi=np.zeros(tiempo+1)#flujo termico
R=np.zeros(tiempo+1)#ritmo de reacci n
actividad=np.zeros(tiempo+1)
t=np.arange(0,tiempo+1,1)
pasobu=np.zeros(tiempo+1)#paso de tiempo de calculo
pasobu[1]=0
contador=0
tiempo_calculo=0
for n in range (1,len(lines)):
    actualizar_barras(CR1,CR2,CR3,CR4,CR5,t[n-1])
    actualizar_posiciones_irrad(posiciones_irrad,t[n-1])
    if(n==1 or check_posiciones_irrad(posiciones_irrad,t[n-1],tiempo_calculo) or check_barrascontrol(barras_control
         ,t[n-1],tiempo_calculo) or n==tiempo_calculo+30):
        pasobu[n]=pasobu[n-1]+(POWER[n-1]+POWER[n-2])/40*1/1440
        actualizar_pasobu(POWER,pasobu[n-1],n,n-1)
        os.system(command1)
        os.system(command2)
        pasobu[n]=0
        phi[n]=leer_flujo("ciclo-03.cio",3201)*POWER[n]/20
        contador=contador+1
        tiempo_calculo=t[n]
    else:
        phi[n]=phi[n-1]*POWER[n-1]/POWER[n-2]
        pasobu[n]=pasobu[n-1]+(POWER[n-1]+POWER[n-2])/40*1/1440
    R[n]=phi[n]*yield_mo*sigma_fision_u5*dens_u*(6.022E+23)/235*enriquecimiento*alto*largo*ancho/(3.7E+10)*
         cant_placas*np.exp(-lambda_mo*6*24*60)
    actividad[n]=actividad[n-1]*np.exp(-lambda_mo*Delta_t)+R[n]*(1-np.exp(-lambda_mo*Delta_t))
    print(t[n],actividad[n])

plt.plot(t,actividad, label="actividad calculada por el simulador")
plt.legend()
plt.grid()
plt.xlabel("Tiempo (min)")
plt.ylabel("Actividad (6d Ci)")
```

Apéndice B

Planificador Mo^{99}

```
import matplotlib.pyplot as plt
import numpy as np
##################### Moli Datos generales##################
lambda_mo=np.log(2)/(65.94/24)#en dias
flujo=1.3E+14
yield_mo=0.0613
masa_u5=1.4
sigma_fision_u5=5.8E-22
enriquecimiento=0.1975
largo=11.7
alto=3
ancho=0.07
dens_u=masa_u5/(largo*alto*ancho)/enriquecimiento #ok
cant_placas=12
t_enf_proc=1#tiempo de enfriamiento y procesamiento en dias

def actividad_eficaz_6dci(flujo):#funci n que me da la actvidad afectada por
# tiempo de enfriamiento y procesamiento, y 6 d ci. Hay que aplicarle el rendimiento del proceso de disoluci n
    actividad_bqcm3=yield_mo*flujo*sigma_fision_u5*dens_u*(6.022E+23)/235*enriquecimiento
    actividad_ci=actividad_bqcm3*alto*largo*ancho/(3.7E+10)*cant_placas
    actividad_6dci=actividad_ci*np.exp(-lambda_mo*6)
    actividad_eficaz_6dci=actividad_6dci*np.exp(-lambda_mo*t_enf_proc)#actvidad afectada por
    return actividad_eficaz_6dci
def actividad_6dci(flujo):#funcion que me da la actvidad afectada por
# tiempo de enfriamiento y procesamiento, y 6 d ci.hay que aplicarle rendimiento
    actividad_bqcm3=yield_mo*flujo*sigma_fision_u5*dens_u*(6.022E+23)/235*enriquecimiento
    actividad_ci=actividad_bqcm3*alto*largo*ancho/(3.7E+10)*cant_placas
    actividad_6dci=actividad_ci*np.exp(-lambda_mo*6)
    return actividad_6dci

def rendimiento(dia):# Funcion rendimiento decreciente #######
    if(dia<=5):
        return 0.6
    if(dia==14):
        return 0.4
    if(dia>5 and dia<14):
        return 0.6-(0.6-0.4)/9*(dia-5)
    if(dia>14):
        return 0.4

def act_mo(t,flujo):# Funcion que da la actividad durante la irradiacion #####
       actividad=actividad_eficaz_6dci(flujo)*(1-np.exp(-lambda_mo*t))*rendimiento(t)
       return actividad
def decai_mo(Act,t):# Funcion que da la actividad durante la interrupcion de la irradiaci n, decaimiento
     exponencial
    return Act*np.exp(-lambda_mo*t)
def reirrad_mo(ti,trf,t,flujo): # Funcion que me da la actividad durante la reirradiacion luego de una parada
#ti es el tiempo de irradiacion previo, trf es el fin de la parada, trf-ti es la duracion de la parada
    a=(1-np.exp(-lambda_mo*t)+np.exp(lambda_mo*(ti-t))-np.exp(lambda_mo*(trf-t)))*actividad_eficaz_6dci(flujo)*
         rendimiento(ti+t-(ti+trf))
    return a
def act_mo_cambio_flujo(t,tf,x1,x2,flujo):#Funcion que da la actividad al cambiar la potencia
#x1 es el % de potencia antes del cambio
#x2 es el % de potencia despues
#tf es el momento donde ocurre el cambio
```

```
    actividad=(x2*(1-np.exp(lambda_mo*(tf-t)))+x1*(np.exp(lambda_mo*tf)-1)*np.exp(-lambda_mo*t))*rendimiento(t)*
         actividad_eficaz_6dci(flujo)
    return actividad
def act_mo_max(flujo):#Funcion que me da la actividad maxima dado cierto flujo
    t=np.zeros(14)
    a=np.zeros(14)
    for i in range (0,len(a)):
        t[i]=i
        a[i]=act_mo(t[i],flujo)
    amax=np.amax(a)
    return amax
def tiempo_irrad_s_interrupcion(act_deseada,flujo):# Funcion que calcula el tiempo de irradiacion en caso que no
     haya interrupcion
    t=np.arange(0,14.5,0.5)
    found=0
    a=[0]*len(t)
    ti=np.zeros(1)
    for i in range (0,len(t)):
        a[i]=act_mo(t[i],flujo)
    for i in range (0,len(a)):
        if(a[i]>=act_deseada):
            ti=i/2
            found=1
            break
        else:
            ti=np.where(a==np.amax(a))
            ti=int(ti[0])/2
    return ti



def ocupar(pos_irr,a,b):# Funcion ocupar posicion de irradiacion desde el momento a a b,me pone un 1 en el vector
     posicion en 1 desde a a b
    for i in range (a-1,b):
        pos_irr[i]=1
def check_libre(pos_irr,a,b):
    for i in range (a,b):
        z=pos_irr[i]
        if(z!=0):
            bool=False
            break
        else:
            bool=True
    return bool
#%%#################Conjunto de funciones para el calculo del tiempo de irradiacion ante una parada
def familia_curvas(trf,flujo):#Familia de curvas de la actividad ante una parada de la irradiacion para un dado trf
      variando ti, trf es la longitud de la parada
    t=np.arange(0,15,0.5)
    a=np.zeros((len(t),12))
    for ti in range (1,13):
         #para antes de la interrupcion tenemos asi
        for i in range (0,int(ti*2)+1):
                a[i][ti-1]= act_mo(t[i],flujo)
        for i in range (int(ti*2)+1,int((ti+trf)*2)+1):
                a[i][ti-1]=decai_mo(a[i-1][ti-1],t[i]-t[i-1])
        for i in range (int((ti+trf)*2)+1,len(t)):
                a[i][ti-1]=reirrad_mo(ti,trf+ti,t[i],flujo)
    return a

def familia_curvas1(trf,tparada,flujo):#familia de curvas para un dado tparada y un dado trf
#esta funcion me corta la curva de reirradiacion en tparada dias despues de trf
#tparada es el tiempo de reirradiac-ion
        #cuantos dias antes del rf occurre la parada
    t=np.arange(0,18.5,0.5)
    a=np.zeros((37,14))
    for ti in range (1,13):
         #para antes de la interrupcion tenemos asi
        for i in range (0,int(ti*2)+1):
                a[i][ti-1]= act_mo(t[i],flujo)
        for i in range (int(ti*2)+1,int((ti+trf)*2+1)):
                a[i][ti-1]=decai_mo(a[i-1][ti-1],t[i]-t[i-1])
        for i in range (int((ti+trf)*2)+1,int(2*(ti+trf+tparada)+1)):
                a[i][ti-1]=reirrad_mo(ti,trf+ti,t[i],flujo)
    return a

def act_max_luego_reirradiar(a):#esta funcion me da la actividad en el ultimo dia
    x=0
    b=np.zeros(12)
    for j in range (0, len(b)-1):
        for i in range (1, 34):
```

```
            if(a[i,j]==0):
                        x=i-1
                        b[j]=a[x,j]
                        break
    b[len(b)-1]=a[32,len(b)-1]
    return b

def tiempo_irr_ante_parada(act_deseada,trf,t_parada,flujo):
    result=np.zeros(1)
    a=familia_curvas1(trf,t_parada,flujo)
    b=act_max_luego_reirradiar(a)
    found=0
    for i in range (0,len(b)):
        if(b[i]>=act_deseada):#si la actividad es la deseada
            found=1
            result=i+1
            break
        else:#no lo encontre asi que voy a buscar el max
            result=np.where(b==np.amax(b))
            result=int(result[0])+1
    return result
tiempo_irr_ante_parada(400,1,1,1.3E+14)

#%%#################Conjunto de funciones para el calculo del tiempo de irradiacion ante un cambio de potencia


def familia_curvas_cambio_pot(x1,x2,tc,tent,flujo):
    t=np.arange(0,tent+1,1)
    a=np.zeros((len(t),tc+1))
    for j in range (0,tc+1):
        for i in range (0,j+1):
            a[i][j]=act_mo(t[i],flujo)*x1
        for i in range (j+1,j+tent-tc+1):
            a[i][j]=act_mo_cambio_flujo(t[i],j,x1,x2,flujo)
    return a

def act_max_luego_cambio_pot(a):
    x=0
    b=np.zeros(len(a[0,:]))
    for j in range (0, len(b)):
        for i in range (1, len(a)):
            if(a[i][j]==0):
                        x=i-1
                        b[j]=a[x,j]
                        break
    b[len(b)-1]=a[len(a)-1,len(b)-1]
    return b

def tiempo_irr_ante_cambio_potencia(act_deseada,x1,x2,tent,tc,flujo):#x1 es la potencia antes
#tent: tiempo de entrega, tc:tiempo en el que ocurre cambio de potencia.
    act_deseada=400
    result=np.zeros(1)
    a=familia_curvas_cambio_pot(x1,x2,tc,tent,flujo)
    b=act_max_luego_cambio_pot(a)
    found=0
    for i in range (0,len(b)):
        if(b[i]>=act_deseada):#si la actividad es la deseada
            found=1
            result=i#a este igual le sumo 1?
            break
        else:#no lo encontre asi que voy a buscar el max
            result=np.where(b==np.amax(b))
            result=int(result[0])
    return result
tiempo_irr_ante_cambio_potencia(400,1,0.923,11,10,1.3E+14)
#%%
#################Conjunto de funciones para el calculo del tiempo de irradiacion ante una innterrupcion de la
      irradiacion con cambio de potencia
def act_mo_reirrad_c_cambio_flujo(x1,x2,ti,tf,t,flujo):
    if (x1>1  or tf<ti or t<tf):
        return ("error")
    actividad=(x2*(1-np.exp(lambda_mo*(tf-t)))+x1*(np.exp(lambda_mo*ti)-1)*np.exp(-lambda_mo*t))*rendimiento(t)*
         actividad_eficaz_6dci(flujo)
    return actividad

def familia_curvas_parada_cambio(trf,tparada,flujo,x1,x2):#familia de curvas para un dado tparada y un dado trf
#esta funcion me corta la curva de reirradiacion en tparada dias despues de trf
#tparada es el tiempo de reirradiacion
        #cuantos dias antes del rf occurre la parada
```

```
    for ti in range (1,13):
         #para antes de la interrupcion tenemos asi
        for i in range (0,int(ti*2)+1):
                a[i][ti-1]= act_mo(t[i],flujo)
        for i in range (int(ti*2)+1,int((ti+trf)*2+1)):
                a[i][ti-1]=decai_mo(a[i-1][ti-1],t[i]-t[i-1])
        for i in range (int((ti+trf)*2)+1,int(2*(ti+trf+tparada)+1)):
                a[i][ti-1]=act_mo_reirrad_c_cambio_flujo(x1,x2,ti,ti+trf,t[i],flujo)

    return a

def act_max_luego_cambio_pot(a):
    x=0
    b=np.zeros(12)
    for j in range (0, len(b)):
        for i in range (1, len(a)):
            if(a[i,j]==0):
                        x=i-1
                        b[j]=a[x,j]
                        break
  #  b[len(b)-1]=a[32,len(b)-1]
    return b

def tiempo_irr_ante_parada_y_cambio_potencia(act_deseada,trf,tparada,flujo,x1,x2):
    result=np.zeros(1)
    a=familia_curvas_parada_cambio(trf,tparada,flujo,x1,x2)
    b=act_max_luego_cambio_pot(a)
    found=0
    for i in range (0,len(b)):
        if(b[i]>=act_deseada):#si la actividad es la deseada
            found=1
            result=i#a este igual le sumo 1?
            break
        else:#no lo encontre asi que voy a buscar el max
            result=np.where(b==np.amax(b))
            result=int(result[0])+1
    return result

#%%%Conjunto de funciones para el calculo del tiempo de irradiacion ante una innterrupcion de la irradiacion con
      cambio de potencia posterior

def act_mo_reirrad_c_cambio_flujo_posterior(x1,x2,ti,tf,tc,t,flujo):

    actividad=(x2*(1-np.exp(lambda_mo*(tc-t)))+x1*(np.exp(lambda_mo*tc)-1+np.exp(lambda_mo*ti)-np.exp(lambda_mo*tf)
         )*np.exp(-lambda_mo*t))*rendimiento(t)*actividad_eficaz_6dci(flujo)
    return actividad

t=np.arange(0,18,1)
a=np.zeros(shape=(len(t),13))#np.empty((len(t),10))
def familia_curvas_parada_cambio_posterior(trf,tri,tparada,flujo,x1,x2):#familia de curvas para un dado tparada y
      un dado trf
#esta funcion me corta la curva de reirradiaci n en tparada dias despues de trf
#tparada es el tiempo de reirradiacion


    for ti in range (1,13):
        for i in range (0,ti+1):
                a[i][ti-1]= act_mo(t[i],flujo)
        for i in range (ti+1,ti+trf+1):
                a[i][ti-1]=decai_mo(a[i-1][ti-1],t[i]-t[i-1])
        for i in range (ti+trf+1,ti+trf+tri+1):
                a[i][ti-1]=reirrad_mo(ti,trf+ti,t[i],flujo)
        for i in range (ti+trf+tri+1,ti+trf+tri+tparada+1):
                a[i][ti-1]=act_mo_reirrad_c_cambio_flujo_posterior(x1,x2,ti,ti+trf,ti+trf+tri,t[i],flujo)

    return a

def act_max_luego_cambio_pot(a):
    x=0
    b=np.zeros(12)
    for j in range (0, len(b)):
        for i in range (1, len(a)):
            if(a[i,j]==0):
                        x=i-1
                        b[j]=a[x,j]
                        break
  #  b[len(b)-1]=a[32,len(b)-1]
    return b
```

```
def tiempo_irr_ante_parada_y_cambio_potencia_posterior(act_deseada,trf,tri,tparada,flujo,x1,x2):#x1 es la potencia
      antes

    result=np.zeros(1)
    a=familia_curvas_parada_cambio_posterior(trf,tri,tparada,flujo,x1,x2)
    b=act_max_luego_cambio_pot(a)
    found=0
    for i in range (0,len(b)):
        if(b[i]>=act_deseada):#si la actividad es la deseada
            found=1
            result=i#a este igual le sumo 1?
            break
        else:#no lo encontre asi que voy a buscar el max
            result=np.where(b==np.amax(b))
            result=int(result[0])+1
    return result

#%%%%%%%Input de la matriz reactor

reactor=np.zeros((30,9))
#dia | demanda | parada | flujo1 | x 1| pos_irr1| flujo2 | x 2| pos_irr2
for i in range (0,len(reactor)):
    reactor[i][0]=i+1#col 0 son los dias
    reactor[i][3]=1.3E+14 #col2 son los flujos pos 1
    reactor[i][6]=1.3E+14 #col6 son los flujos pos 2
reactor[10][1]=400
reactor[11][1]=800
for i in range (8,30):
    reactor[i][3]=1.2E+14#el d a 8 el flujo se reduce a 1,2
    reactor[i][6]=1.2E+14
reactor[8][2]=1#el dia 4 hay parada de un dia
reactor[0][4]=1#r=1
reactor[0][7]=1#r=1
for i in range (1,len(reactor)):
    reactor[i][4]=reactor[i][3]/reactor[i-1][3]
    reactor[i][7]=reactor[i][6]/reactor[i-1][6]

#tambi n se pueden leer los flujo de citvap con la siguiente funci n
def leer_flujo(archivo,nombre_pos_irrad):
    f = open(archivo, "r")
    x=f.read()
    substring="␣ZONE␣␣␣"+str(nombre_pos_irrad)
    y=x.find(substring)
    z=x.find("\n",y,len(x))
    b=""
    if(z!=1):
        for j in range(y+len(substring)+1,z):
            b=b+x[j]
    if(z==-1):
        for j in range(y+len(substring)+1,len(x)):
            b=b+x[j]
    f.close()
    flujo_th=""
    for j in range (len(b)-11,len(b)):
        flujo_th=flujo_th+b[j]
    flujo_th=float(flujo_th)
    return flujo_th

def check_cambio_potencia(reactor,a,b):
    for i in range (a,b):
        if(reactor[i]!=reactor[i-1]):
            bool=1
            break
        else:
            bool=0
    return bool


def momento_inicio_parada(reactor,a,b):
    momento_parada=0
    for i in range (a-1,b):
        if(reactor[i]==1):
            momento_parada=i+1
            break
    return momento_parada

def momento_fin_parada(reactor,a,b):
    momento_fin_parada=0
    for i in range (a,b):
        if(reactor[i]==0 and reactor[i-1]==1):
```

```
            momento_fin_parada=i+1
    return momento_fin_parada

def duracion_parada(reactor,a,b):
    duracion_parada=-momento_inicio_parada(reactor,a,b)+momento_fin_parada(reactor,a,b)
    return duracion_parada
def momento_cambio_potencia(reactor,a,b):
    for i in range (a,b):
        if(reactor[i]!=reactor[i-1]):
            momento_cambio=i
    return momento_cambio

def potencia_antes(reactor,a,b):
    tc=momento_cambio_potencia(reactor,a,b)
    return reactor[tc-2]



#%%PLANIFICADOR PARA mas de una POSICION DE IRRADAICI N

cant_posiciones_irrad=2#la cantidad total en el reactor
for i in range (0,len(reactor)):
    if(reactor[i][1]!=0):#si hay demanda
        if(reactor[i][1]>act_mo_max(reactor[i][3])*1.05):#si la actividadad deseada es mayor que la que puedo
             generar con ese flujo, error
            print(" la actividad es muy grande para una sola posicion irradiacion")
            cant_pos_irr_necesarias=int(np.ceil(reactor[i][1]/(act_mo_max(reactor[i][3]*1.05))))
            if (cant_pos_irr_necesarias>cant_posiciones_irrad):
                print(" no hay suficientes posiciones de irradiaci n")
                break
        else:#sisolo hay que ocupar una posici n de irrad
            cant_pos_irr_necesarias=1

        ti=tiempo_irrad_s_interrupcion(reactor[i][1]/cant_pos_irr_necesarias,reactor[i][3])

        if(check_libre(reactor[:,2],int(reactor[i][0]-ti),int(reactor[i][0]))==True and check_cambio_potencia(
             reactor[:,4],int(reactor[i][0]-ti),int(reactor[i][0]))==0):
                #si no hay parada ni cambio de potencia
            print("no hay nada")
            for k in range (0,cant_pos_irr_necesarias):
                if(check_libre(reactor[:,5+3*k],int(reactor[i][0]-ti),int(reactor[i][0]))):
                    ocupar(reactor[:,5+3*k],int(reactor[i][0]-ti),int(reactor[i][0]))
                else:
                    print("pasar a la siguiente posici n de irradiaci n")
                    for j in range (1,cant_posiciones_irrad+1):
                        if(j==cant_posiciones_irrad):
                            print("error, se acabaron las posiciones de irradiaci n")
                            break
                        ti=tiempo_irrad_s_interrupcion(reactor[i][1]/cant_pos_irr_necesarias,reactor[i][3+3*j])
                        if(check_libre(reactor[:,5+3*j],int(reactor[i][0]-ti),int(reactor[i][0]))):
                            ocupar(reactor[:,5+3*j],int(reactor[i][0]-ti),int(reactor[i][0]))
                            break

        if(check_libre(reactor[:,2],int(reactor[i][0]-ti),int(reactor[i][0]))==False and check_cambio_potencia(
             reactor[:,4],int(reactor[i][0]-ti),int(reactor[i][0]))==0):
                #si hay parada pero no cambio de potencia
            print("hay parada")
            trf=duracion_parada(reactor[:,2],int(reactor[i][0]-ti),int(reactor[i][0]))
            t_parada=reactor[i][0]-momento_fin_parada(reactor[:,2],int(reactor[i][0]-ti),int(reactor[i][0]))+1
            t_parada=int(t_parada)
            for k in range (0,cant_pos_irr_necesarias):
                ti=tiempo_irr_ante_parada(reactor[i][1]/cant_pos_irr_necesarias,trf,t_parada,reactor[i][3+3*k])
                if(check_libre(reactor[:,5+3*k],int(reactor[i][0]-(ti+trf+t_parada+1)),int(reactor[i][0]))):
                    ocupar(reactor[:,5+3*k],int(reactor[i][0]-(ti+trf+t_parada+1)),int(reactor[i][0]))
                else:
                    print("pasar a la siguiente posici n de irradiaci n")
                    for j in range (1,cant_posiciones_irrad+1):
                            if(j==cant_posiciones_irrad):
                                print("error, se acabaron las posiciones de irradiaci n")
                                break
                            ti=tiempo_irr_ante_parada(reactor[i][1]/cant_pos_irr_necesarias,trf,t_parada,reactor[i
                                 ][3+3*j])
                            if(check_libre(reactor[:,5+3*j],int(reactor[i][0]-(ti+trf+t_parada+1)),int(reactor[i
                                 ][0]))):
                                ocupar(reactor[:,5+3*j],int(reactor[i][0]-(ti+trf+t_parada+1)),int(reactor[i][0]))
                                break
        if(check_libre(reactor[:,2],int(reactor[i][0]-ti),int(reactor[i][0]))==True and check_cambio_potencia(
             reactor[:,4],int(reactor[i][0]-ti),int(reactor[i][0]))==True):
                #si no hay parada pero si hay cambio de potencia
            print("hay cambio de potencia")
```

```
            x1=potencia_antes(reactor[:,4+3*k],int(reactor[i][0]-ti),int(reactor[i][0]))
            tc=momento_cambio_potencia(reactor[:,4],int(reactor[i][0]-ti),int(reactor[i][0]))
            for k in range (0,cant_pos_irr_necesarias):
                ti=tiempo_irr_ante_cambio_potencia(reactor[i][1]/cant_pos_irr_necesarias,x1,reactor[tc-1][4+3*k],
                     int(reactor[i][0]),tc,reactor[tc-2][3+3*k])
                if(check_libre(reactor[:,5+3*k],int(tc-ti-1),int(reactor[i][0]))):
                    ocupar(reactor[:,5+3*k],int(tc-ti-1),int(reactor[i][0]))
                else:
                    print("pasar␣a␣la␣siguiente␣posici n␣de␣irradiaci n")
                    for j in range (1,cant_posiciones_irrad+1):
                            if(j==cant_posiciones_irrad):
                                print("error,␣se␣acabaron␣las␣posiciones␣de␣irradiaci n")
                                break
                            ti=tiempo_irr_ante_cambio_potencia(reactor[i][1],x1,reactor[tc-1][4+3*j],int(reactor[i
                                 ][0]),tc,reactor[tc-2][3+3*j])
                            if(check_libre(reactor[:,5+3*j],int(tc-ti-1),int(reactor[i][0]))):
                                ocupar(reactor[:,5+3*j],int(tc-ti-1),int(reactor[i][0]))
                                break
        if(check_libre(reactor[:,2],int(reactor[i][0]-ti),int(reactor[i][0]))==False and check_cambio_potencia(
             reactor[:,4],int(reactor[i][0]-ti),int(reactor[i][0]))==True):
            #si hay parada y cambio de potencia
            print("hay␣cambio␣de␣potencia␣+␣parada")
            tc=momento_cambio_potencia(reactor[:,4],int(reactor[i][0]-ti),int(reactor[i][0]))
            trf=duracion_parada(reactor[:,2],int(reactor[i][0]-ti),int(reactor[i][0]))
            t_parada=reactor[i][0]-momento_fin_parada(reactor[:,2],int(reactor[i][0]-ti),int(reactor[i][0]))
            t_parada=int(t_parada)
            t_fin_parada=momento_fin_parada(reactor[:,2],int(reactor[i][0]-ti),int(reactor[i][0]))
            t_inicio_parada=momento_fin_parada(reactor[:,2],int(reactor[i][0]-ti),int(reactor[i][0]))-trf
            for k in range (0,cant_pos_irr_necesarias):
                if(tc>=t_inicio_parada and tc<=t_fin_parada):
                    print("el␣cambio␣de␣potencia␣se␣da␣en␣la␣parada")
                    ti=tiempo_irr_ante_parada_y_cambio_potencia(reactor[i][1]/cant_pos_irr_necesarias,trf,t_parada,
                         reactor[t_inicio_parada-1][3+3*k],reactor[t_inicio_parada-2][4+3*k],reactor[
                         t_inicio_parada-1][4+3*k])
                    if(check_libre(reactor[:,5+3*k],int(reactor[i][0]-(ti+trf+t_parada)),int(reactor[i][0]))):
                        ocupar(reactor[:,5+3*k],int(reactor[i][0]-(ti+trf+t_parada)),int(reactor[i][0]))
                    else:
                        print("error,␣la␣posici n␣de␣irradiaci n␣esta␣ocupada")
                        for j in range (1,cant_posiciones_irrad+1):
                                if(j==cant_posiciones_irrad):
                                    print("error,␣se␣acabaron␣las␣posiciones␣de␣irradiaci n")
                                    break
                                ti=tiempo_irr_ante_parada_y_cambio_potencia(reactor[i][1]/cant_pos_irr_necesarias,
                                     trf,t_parada,reactor[t_inicio_parada-1][3+3*j],reactor[t_inicio_parada-2][4+3*
                                     j],reactor[t_inicio_parada-1][4+3*j])
                                if(check_libre(reactor[:,5+3*j],int(reactor[i][0]-(ti+trf+t_parada)),int(reactor[i
                                     ][0]))):
                                    ocupar(reactor[:,5+3*j],int(reactor[i][0]-(ti+trf+t_parada)),int(reactor[i][0])
                                         )
                                    break
                else:
                    print("el␣cambio␣de␣potencia␣se␣da␣despues")
                    t_parada=reactor[i][0]-tc
                    t_parada=int(t_parada)
                    ti=tiempo_irr_ante_parada_y_cambio_potencia_posterior(reactor[i][1]/cant_pos_irr_necesarias,trf
                         ,tc-t_fin_parada,t_parada,reactor[tc-1][3+3*k],reactor[tc-2][4+3*k],reactor[tc-1][4+3*k])
                    if(check_libre(reactor[:,5+3*k],int(reactor[i][0]-(ti+tc-t_fin_parada+trf+t_parada)),int(
                         reactor[i][0]))):
                        ocupar(reactor[:,5+3*k],int(reactor[i][0]-(ti+tc-t_fin_parada+trf+t_parada)),int(reactor[i
                             ][0]))
                        print(int(reactor[i][0]-(ti+tc-t_fin_parada+trf+t_parada)))
                    else:
                        print("error,␣la␣posici n␣de␣irradiaci n␣esta␣ocupada")
                        for j in range (1,cant_posiciones_irrad+1):
                            if(j==cant_posiciones_irrad):
                                    print("error,␣se␣acabaron␣las␣posiciones␣de␣irradiaci n")
                                    break
                            ti=tiempo_irr_ante_parada_y_cambio_potencia_posterior(reactor[i][1]/
                                 cant_pos_irr_necesarias,trf,tc-t_fin_parada,t_parada,reactor[tc-1][3+3*j],reactor[
                                 tc-2][4+3*j],reactor[tc-1][4+3*j])
                            if(check_libre(reactor[:,5+3*j],int(reactor[i][0]-(ti+tc-t_fin_parada+trf+t_parada)),
                                 int(reactor[i][0]))):
                                ocupar(reactor[:,5+3*j],int(reactor[i][0]-(ti+tc-t_fin_parada+trf+t_parada)),int(
                                     reactor[i][0]))
                            print(int(reactor[i][0]-(ti+tc-t_fin_parada+trf+t_parada)))
                            break
    else:
        i=i+1
```

Apéndice C

Planificador Lu^{177}

```
import numpy as np
import matplotlib.pyplot as plt
N=2.76E+22
flujo=3.00E+14
sigma=2.85E-24
lambda_y=0.104232659

l_z=0.364814306

vol=1.090512541
ti=172800
tf=345600
t=432000

def Alu(t,flujo):

    A1=(1-np.exp(-l_z*t)+l_z/(lambda_y-l_z)*(np.exp(-lambda_y*t)-np.exp(-l_z*t)))*N*flujo*sigma/(3.7E+10)*vol
    return A1

def Alu_reirrad(t,ti,tf,flujo):
    a=N*flujo*sigma*((np.exp(l_z*ti)-1-np.exp(l_z*tf))*np.exp(-l_z*t)+1+l_z/(lambda_y-l_z)*((np.exp(ti*(l_z-
         lambda_y))-1-np.exp(tf*(l_z-lambda_y)))*np.exp(-l_z*t)+np.exp(-lambda_y*t)))/(3.7E+10)*vol
    return a
def decai_lu(Act,t):# Funcion que da la actividad durante la interrupci n de la irradiaci n, decaimiento
     exponencial
    return Act*np.exp(-l_z*t)


#%%
def familia_curvas1(trf,tparada,flujo):#familia de curvas para un dado tparada y un dado trf
#esta funcion me corta la curva de reirradiaci n en tparada dias despues de trf
#tparada es el tiempo de reirradiac-ion
        #cuantos dias antes del rf occurre la parada
    t=np.arange(0,18.5,0.5)
    a=np.zeros((37,14))
    for ti in range (1,13):
        for i in range (0,int(ti*2)+1):
                a[i][ti-1]= Alu(t[i],flujo)
        for i in range (int(ti*2)+1,int((ti+trf)*2+1)):
                a[i][ti-1]=decai_lu(a[i-1][ti-1],t[i]-t[i-1])
        for i in range (int((ti+trf)*2)+1,int(2*(ti+trf+tparada)+1)):
                a[i][ti-1]=Alu_reirrad(t[i],ti,trf+ti,flujo)
    return a
t=np.arange(0,18.5,0.5)
a=familia_curvas1(1,1,1.3E+14)
plt.plot(t,a)
#%%
def act_max_luego_reirradiar(a):#esta funcion me da la actividad en el ultimo  d  a
    x=0
    b=np.zeros(12)
    for j in range (0, len(b)-1):
        for i in range (1, 34):
            if(a[i,j]==0):
                        x=i-1
                        b[j]=a[x,j]
                        break
```

```
    b[len(b)-1]=a[32,len(b)-1]
    return b

def tiempo_irr_ante_parada(act_deseada,trf,t_parada,flujo):
    result=np.zeros(1)
    a=familia_curvas1(trf,t_parada,flujo)
    b=act_max_luego_reirradiar(a)
    found=0
    for i in range (0,len(b)):
        if(b[i]>=act_deseada):
            found=1
            result=i+1
            break
        else:
            result=np.where(b==np.amax(b))
            result=int(result[0])+1
    return result

def tiempo_irrad_s_interrupcion(act_deseada,flujo):# Funci n que calcula el tiempo de irradiacion en caso que no
     haya interrupcion
    t=np.arange(0,14.5,0.5)
    found=0
    a=[0]*len(t)
    ti=np.zeros(1)
    for i in range (0,len(t)):
        a[i]=Alu(t[i],flujo)
    for i in range (0,len(a)):
        if(a[i]>=act_deseada):
            ti=i/2
            found=1
            break
        else:
            ti=np.where(a==np.amax(a))
            ti=int(ti[0])/2
    return ti
def ocupar(pos_irr,a,b):# Funcion ocupar posici n de irradiaci n desde el momento a a b,me pone un 1 en el vector
      posicion en 1 desde a a b
    for i in range (a-1,b):
        pos_irr[i]=1
def check_libre(pos_irr,a,b):
    for i in range (a,b):
        z=pos_irr[i]
        if(z!=0):
            bool=False
            break
        else:
            bool=True
    return bool
#%%%%%%%Input de la matriz reactor

reactor=np.zeros((30,9))
#dia | demanda | parada | flujo1 | x 1| pos_irr1| flujo2 | x 2| pos_irr2
for i in range (0,len(reactor)):
    reactor[i][0]=i+1#col 0 son los dias
    reactor[i][3]=1.3E+14 #col2 son los flujos pos 1
    reactor[i][6]=1.3E+14 #col6 son los flujos pos 2
reactor[10][1]=400
#reactor[11][1]=800

reactor[8][2]=1#el dia 4 hay parada de un dia
reactor[0][4]=1#r=1
reactor[0][7]=1#r=1
for i in range (1,len(reactor)):
    reactor[i][4]=reactor[i][3]/reactor[i-1][3]
    reactor[i][7]=reactor[i][6]/reactor[i-1][6]

def momento_inicio_parada(reactor,a,b):
    momento_parada=0
    for i in range (a-1,b):
        if(reactor[i]==1):
            momento_parada=i+1
            break
    return momento_parada

def momento_fin_parada(reactor,a,b):
    momento_fin_parada=0
    for i in range (a,b):
        if(reactor[i]==0 and reactor[i-1]==1):
            momento_fin_parada=i+1
    return momento_fin_parada
```

```
def duracion_parada(reactor,a,b):
    duracion_parada=-momento_inicio_parada(reactor,a,b)+momento_fin_parada(reactor,a,b)
    return duracion_parada
#%%PLANIFICADOR PARA mas de una POSICION DE IRRADAICI N

cant_posiciones_irrad=2#la cantidad total en el reactor
for i in range (0,len(reactor)):
    if(reactor[i][1]!=0):#si hay demanda
        if(reactor[i][1]>Alu(30,reactor[i][3])):#si la actividadad deseada es mayor que la que puedo generar con
             ese flujo y un tiempo m ximo de irradiaci n, error
            print("␣la␣actividad␣es␣muy␣grande␣para␣una␣sola␣posicion␣irradiacion")
            cant_pos_irr_necesarias=int(np.ceil(reactor[i][1]/Alu(30,reactor[i][3])))
            if (cant_pos_irr_necesarias>cant_posiciones_irrad):
                print("error,␣no␣hay␣suficientes␣posiciones␣de␣irradiaci n")
                break
        else:#sisolo hay que ocupar una posici n de irrad
            cant_pos_irr_necesarias=1

        ti=tiempo_irrad_s_interrupcion(reactor[i][1]/cant_pos_irr_necesarias,reactor[i][3])

        if(check_libre(reactor[:,2],int(reactor[i][0]-ti),int(reactor[i][0]))==True):
                #si no hay parada ni cambio de potencia
            print("no␣hay␣nada")
            for k in range (0,cant_pos_irr_necesarias):
                if(check_libre(reactor[:,5+3*k],int(reactor[i][0]-ti),int(reactor[i][0]))):
                    ocupar(reactor[:,5+3*k],int(reactor[i][0]-ti),int(reactor[i][0]))
                else:
                    print("pasar␣a␣la␣siguiente␣posici n␣de␣irradiaci n")
                    for j in range (1,cant_posiciones_irrad+1):
                        if(j==cant_posiciones_irrad):
                            print("error,␣se␣acabaron␣las␣posiciones␣de␣irradiaci n")
                            break
                        ti=tiempo_irrad_s_interrupcion(reactor[i][1]/cant_pos_irr_necesarias,reactor[i][3+3*j])
                        if(check_libre(reactor[:,5+3*j],int(reactor[i][0]-ti),int(reactor[i][0]))):
                            ocupar(reactor[:,5+3*j],int(reactor[i][0]-ti),int(reactor[i][0]))
                            break

        if(check_libre(reactor[:,2],int(reactor[i][0]-ti),int(reactor[i][0]))==False):
                #si hay parada pero no cambio de potencia
            print("hay␣parada")
            trf=duracion_parada(reactor[:,2],int(reactor[i][0]-ti),int(reactor[i][0]))
            t_parada=reactor[i][0]-momento_fin_parada(reactor[:,2],int(reactor[i][0]-ti),int(reactor[i][0]))+1
            t_parada=int(t_parada)
            for k in range (0,cant_pos_irr_necesarias):
                ti=tiempo_irr_ante_parada(reactor[i][1]/cant_pos_irr_necesarias,trf,t_parada,reactor[i][3+3*k])
                if(check_libre(reactor[:,5+3*k],int(reactor[i][0]-(ti+trf+t_parada+1)),int(reactor[i][0]))):
                    ocupar(reactor[:,5+3*k],int(reactor[i][0]-(ti+trf+t_parada+1)),int(reactor[i][0]))
                else:
                    print("pasar␣a␣la␣siguiente␣posici n␣de␣irradiaci n")
                    for j in range (1,cant_posiciones_irrad+1):
                            if(j==cant_posiciones_irrad):
                                print("error,␣se␣acabaron␣las␣posiciones␣de␣irradiaci n")
                                break
                            ti=tiempo_irr_ante_parada(reactor[i][1]/cant_pos_irr_necesarias,trf,t_parada,reactor[i
                                 ][3+3*j])
                            if(check_libre(reactor[:,5+3*j],int(reactor[i][0]-(ti+trf+t_parada+1)),int(reactor[i
                                 ][0]))):
                                ocupar(reactor[:,5+3*j],int(reactor[i][0]-(ti+trf+t_parada+1)),int(reactor[i][0]))
                                break

    else:
        i=i+1
```

Bibliografía

[1] Radioisotopes, (accedido: 09.04.2020). URL https://www.iaea.org/topics/nuclear-science/isotopes/radioisotopes. 3

[2] Radioisotopes, (accedido: 09.04.2020). URL https://www.ansto.gov.au/education/nuclear-facts/what-are-radioisotopes. 3

[3] Stabin, M. G. Radiation Protection and Dosimetry. Berlin: Springer, 2007. 3

[4] R.B., H. Applications of radioisotopes. *Meyers R. (eds) Encyclopedia of Sustainability Science and Technology. Springer, New York, NY*, págs. 891–921, 2017. 4

[5] Radioisotopes & research, (accedido: 09.04.2020). URL https://www.world-nuclear.org/information-library/non-power-nuclear-applications/radioisotopes-research.aspx. 4

[6] Radioisotopes in medicine, (accedido: 09.04.2020). URL https://www.world-nuclear.org/information-library/non-power-nuclear-applications/radioisotopes-research/radioisotopes-in-medicine.aspx. 6

[7] Committee on Medical Isotope Production Without Highly Enriched Uranium, N. R. C. Medical isotope production without highly enriched uranium, 2009. 9, 10, 11

[8] National Academies of Sciences, E., Medicine. Molybdenum-99 for Medical Imaging. Washington: The National Academies Press, 2016. 9

[9] Hergenreder, D. Irradiation requirements for isotope production, 2014. 11, 12

[10] Comunicación oral del área radioquímica de invap, 2005. 11

[11] Rostelato, M. E. C. M. Development and production of radioactive sources used for cancer treatment in brazil, 2008. 11, 12

[12] Dvoráková, Z. Production and chemical processing of 177lu for nuclear medicine at the munich research reactor frm-ii, 2007. 13, 14, 15

[13] Cruz, L. Diseño y optimización de dispositivos de irradiación para la producción de nuevos radioiótopos en gran escala, 2019. 13

[14] Lieser, K. H. Nuclear and Radiochemistry. Alemania: WILEY-VCH, 2001. 17, 20

[15] AGENCY, N. E. Review of potential molybdenum-99/ technetium-99m production technologies, 2010. 21

[16] informe del proceso de demostración de molibdeno de fisión - demostración para invap/ansto, 2005. 24

[17] thermal neutron measurements in the ra-3 irradiation facility, 2005. 24

[18] Sar [safety analysis report] : Replacement research reactor project / prepared by invap for australian nuclear science and technology organisation, 6 May 2005. 44

[19] Reactor de investigación opal, lucas heights, australia, (accedido: 19.06.2020). URL http://www.invap.com.ar/es/area-nuclear-de-invap/proyectos/reactor-opal-de-australia/el-nucleo-del-reactor.html. 44