Problemas de optimización en aprendizaje automático con aplicaciones biomédicas

Trabajo Fin de Grado

FACULTAD DE CIENCIAS MATEMÁTICAS

UNIVERSIDAD
COMPLUTENSE
MADRID

GRADO EN INGENIERÍA MATEMÁTICA-TECNOMATEMÁTICA

Autor: **David Lozano Valle**
Tutora: **Ana Carpio**

Madrid, 17 de Junio de 2023

Resumen

En este trabajo vamos a abordar el diseño de marcos matemáticos que ayuden en el diagnóstico, tratamiento y posible prognóstico de enfermedades. Nos centraremos en un modelo para el **cáncer de próstata**. Primero, plantearemos el problema de optimización desde dos estrategias diferentes:

- **La estrategia determinista:** En ella plantearemos el problema de optimizacion que correspondería al tratamiento del paciente, utilizando como restricción un sistema de EDOS.
- **La estrategia de aprendizaje:** En ella hacemos el mismo planteamiento, pero esta vez, como restricción, utilizaremos una red neuronal previamente ajustada a una colección de datos preseleccionados.

Finalmente, procederemos a resolverlo utilizando los métodos correspondientes a cada uno de los planteamientos propuestos.

Palabras Clave:

Cáncer de Próstata, modelo matemático, optimización, antígeno prostático específico (**PSA**), red neuronal

Abstract

In this work we are going to dealt with the design of mathematical frameworks useful in the diagnosis, treatment and possible prognosis of diseases. We will focus on a model for **prostate cancer**. First, we will pose the optimization problem in two different ways:

- **The deterministic approach:** We formulate an optimization problem that would correspond to the patient's treatment, using an EDOS system as a constraint.
- **The learning approach:** We formulate the optimization problem where the constraint, in this case, will be a neural network previously fitted to a pre-selected data collection.

Finally, we will solve it using the methods corresponding to each of the proposed approaches.

Keywords:

Prostate cancer, mathematical model, optimization, Prostate-Specific Antigen (**PSA**), neural network

Índice de contenidos:

1. Introducción.

El cáncer es una patología producida por el crecimiento descontrolado de las células de un tejido, órgano o cualquier otra parte del cuerpo, recibiendo el nombre de la parte afectada y diferenciándose de un **tumor** por su **capacidad para extenderse hacia otras partes**.

En 2020, el cáncer fue responsable de casi 10 millones de muertes en todo el mundo, siendo una de las principales causas de mortalida

De entre los diversos tipos, vamos a centrar nuestro estudio en el cáncer de próstata, pues se trata de uno de los tipos de cánceres más comunes en todo el mundo, siendo en 2020 el 4º tipo con más casos detectados (1,41 mill) por detrás del cáncer de mama, pulmonar y colorrectal [TFG M. Morales (2017)]. Es la variedad más destacada entre la población masculina, sobre todo a partir de cierta edad, siendo en el 90 % de los casos en mayores de 65 años, y con una edad media de diagnóstico de 75 años [Ruiz et al. (2017)]. En España, estas estadísticas se corresponden con un 9 % de las muertes por cáncer en hombres y un 2.8 % del total de muertes masculinas.

El cáncer de próstata[1], es una patología sexual que afecta a la próstata, como indica su nombre, órgano del **aparato reproductor masculino**, y por tanto, se da únicamente en varones. Se diferencia del resto de cánceres por su lenta evolución, pudiendo pasar hasta 10 años desde que las células se transforman e incrementan hasta que se desarrollen los primeros síntomas, causa por la cual es considerada una **enfermedad silenciosa.**

Para determinar la gravedad y el estado en que se encuentra esta patología, se utiliza el **Sistema de Gleason**[2], que es una escala del 1 al 5 en la que se representa en orden ascendente el nivel de desarrollo del carcinoma [Ruiz et al. (2017)].

De entre las distintas técnicas de detección precoz existentes:

- **Prueba sérica del PSA**
- **Ecografía prostática transrectal**
- **Tacto rectal**

es en la prueba sérica del **PSA (antígeno prostático)** junto con la **terapia de supresión androgénica intermitente**[3], como tratamiento, en la que vamos a basar nuestro modelo matemático.

Para comprender mejor el contexto de nuestro trabajo, vamos a introducir los conceptos de **PSA** y **terapia de supresión androgénica**, así como algunos valores de referencia y otros términos y matices que van a ser de gran importancia durante el desarrollo de nuestro modelo.

[1]**Próstata**: Órgano situado entre la vejiga y el recto y responsable de la producción del líquido seminal.

[2]**Sistema de Gleason:** Se refiere a cómo se ven las células cancerosas de la próstata y qué tan probable es que el cáncer avance y se disemine.

[3]**Supresión androgénica:** El objetivo de este tratamiento hormonal es reducir los niveles de andrógenos en el cuerpo y evitar que estas hormonas estimulen el crecimiento de células cancerosas de la próstata.

El PSA es un biomarcador[4] cuya producción depende de la presencia de andrógenos[5] y del tamaño de la glándula prostática, se trata de una proteína de síntesis exclusiva en la próstata, de la que solo un mínimo porcentaje pasa a la sangre, **PSA sérico**, que es el que se tiene en cuenta durante el diagnóstico, pronóstico y seguimiento de esta enfermedad.

El nivel de PSA sérico es la **prueba más sensible para detectar precozmente el cáncer de próstata**, ya que se eleva en el 65 % de los casos aproximadamente.

En varones sanos, la concentración en sangre de PSA es muy baja, **elevándose durante la enfermedad prostática**. Los **valores de referencia** varían en función del laboratorio, oscilando en la mayoría de los casos entre los **4 ng/mL** [Wikipedia] . Otro factor de variación es la edad, condicionando el que unos niveles sean altos o no. Los niveles de PSA oscilan de forma aleatoria del orden de un 15 % en un mismo individuo, disminuyendo hasta un 50 % en un paciente hospitalizado.

En cuanto a la **supresión de andrógenos**, es importante saber que puede emplearse en el tratamiento durante todas las fases del cáncer de próstata, controlando la respuesta a éste mediante mediciones del PSA. Tiene la limitación de que si se hace forma muy continuada, puede dar paso a la evolución de la enfermedad con andrógeno-independencia[6] (AI), motivo por el cual se está estudiando la aplicación de la **supresión de andrógenos intermitente (IAS)** con el fin de retrasar la resistencia androgénica y mejorar la calidad de vida de los pacientes durante los períodos de no-tratamiento (reposo).

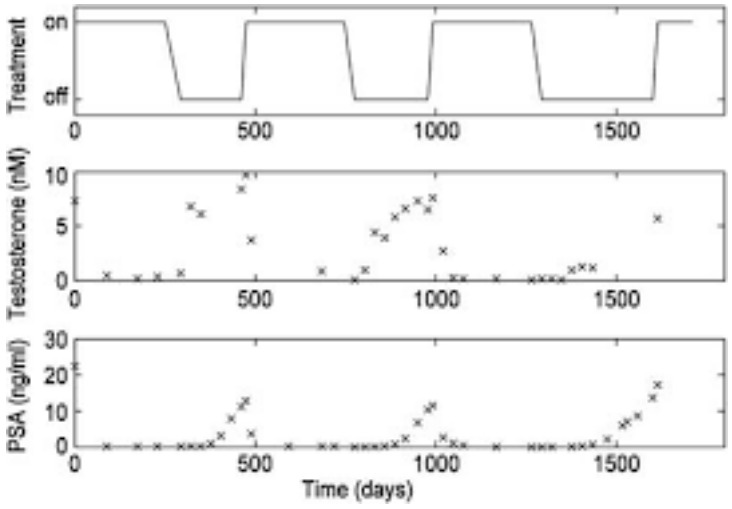

Figura 1: Ejemplo de terapia IAS en un paciente en el que se ve los niveles de PSA y testosterona durante las fases de este [Hirata et al. (2012)].

[4]**Biomarcador**: Molécula biológica que se encuentra en la sangre, otros líquidos o tejidos del cuerpo, y cuya presencia es un signo de un proceso normal o anormal, de una afección o de una enfermedad.

[5]**Andrógeno:** Tipo de hormona que promueve el desarrollo y el mantenimiento de las características sexuales masculinas, corresponden a la **testosterona, la androsterona y la androstenediona**.

[6]**Andrógeno-independencia:** Desarrollo de las células tumorales por la cual éstas pueden seguir creciendo sin necesidad de andrógenos. Los cánceres de próstata avanzados a menudo son andrógeno-independientes.

Durante un tratamiento usual de IAS, como el de la figura 1, ocurren las siguientes fases:

1. **Terapia hormonal:** Para reducir el nivel de testosterona, seguido de un descenso de PSA, con una duración específica de 8-9 meses o hasta que el nivel de PSA desciende hasta un valor por debajo de los 4 ng/ml.

2. **Reposo:** Durante el cual la terapia hormonal es interrumpida, permitiendo una recuperación de los niveles de testosterona y PSA, volviendo de nuevo a repetirse el ciclo cuando estos valores superen aproximadamente los 10 ng/ml.

3. Repetición de los pasos 1 y 2.

Pese a que existen regímenes de la terapia IAS ya establecidos de unos 6-9 meses de terapia hormonal (supresión) seguidos de un período variable de reposo en los que el nivel de PSA se reestablece hasta un límite ya fijado, en este trabajo vamos a optimizar los tiempos incial y final de cada una de las fases del tratamiento (supresión y reposo) considerando el modelo matemático dado en [Hirata et al. (2012)] y utilizando como parámetros los valores iniciales, el cual explicaremos en detalle más adelante, con el fin de mejorar no solo la terapia hormonal en hombres con recaída bioquímica, sino también identificar los pacientes en los que la (IAS) tendrá beneficios frente a los que no, desde dos puntos de vista distintos, que se corresponden con las dos estrategias mencionadas anteriormente.
Por un lado, como problema de optimización del tratamiento, por el otro, la de aprendizaje, por medio de la implementación de redes neuronales que aprendan en función de los parámetros iniciales dados, y den unos valores óptimos.

2. Modelo matemático

2.1. Introducción al concepto de modelización

Un modelo matemático consiste en explicar ciertos fenómenos naturales no matemáticos por medio de técnicas y objetos matemáticos, es decir, encontrar las ecuaciones que explican y describen ese fenómeno. [Morten et al]

La modelización consiste en desarrollar un modelo que se adapte y describa el fenómeno que queremos estudiar.

2.2. Descripción e introducción del modelo utilizado

El modelo sobre el que se basa nuestro trabajo, es el modelo lineal propuesto en [Hirata et al. (2012)], el cual describe cuantitativamente la dinámica del cáncer de próstata, bajo el tratamiento por supresión de andrógenos intermitente (IAS), basándose en el nivel de PSA sérico.

En el modelo, vamos primero a distinguir los 3 tipos distintos de células cancerígenas que aparecen durante el tratamiento:

- **Andrógeno - dependientes (AD)**: Se pueden transformar en los otros dos tipos de células durante la fase de tratamiento.
- **Andrógeno-independientes reversibles (AI tipo 1):** Durante la fase de reposo pueden transformarse de nuevo en andrógeno dependientes y son causa de cambios reversibles (adaptaciones).
- **Andrógeno-independientes irreversibles (AI tipo 2)**: Permanecen irreversibles durante ambas fases del tratamiento y vienen dadas por mutaciones.

Sean $\mathbf{x_0}$, $\mathbf{x_1}$ y $\mathbf{x_2}$ variables que expresan el número de células AD, AI tipo 1 y AI de tipo 2, respectivamente.

Sea $\mathbf{w^m_{i\longrightarrow j}}$ la variable que representa el número de células que se transforman de un tipo a otro (de i a j) durante la fase m de tratamiento, con $i, j \in \{0, 1, 2\}$ y $m \in \{\text{on, off}\}$.

Tal y como explica [Aihara et al. (2010)], vamos a escribir las ecuaciones que describen el modelo en forma de un sistema híbrido, donde durante los períodos de terapia hormonal, tenemos:

$$\frac{d}{dt}\begin{pmatrix} x_0 \\ x_1 \\ x_2 \end{pmatrix} = \begin{pmatrix} w^{\text{off}}_{0\longrightarrow 0} & 0 & 0 \\ w^{\text{off}}_{0\longrightarrow 1} & w^{\text{off}}_{1\longrightarrow 1} & 0 \\ w^{\text{off}}_{0\longrightarrow 2} & w^{\text{off}}_{1\longrightarrow 2} & w^{\text{off}}_{2\longrightarrow 2} \end{pmatrix} \begin{pmatrix} x_0 \\ x_1 \\ x_2 \end{pmatrix} = W^{\text{on}} \begin{pmatrix} x_0 \\ x_1 \\ x_2 \end{pmatrix} \tag{1}$$

y durante la fase de reposo:

$$\frac{d}{dt}\begin{pmatrix} x_0 \\ x_1 \\ x_2 \end{pmatrix} = \begin{pmatrix} w^{\text{off}}_{0\longrightarrow 0} & w^{\text{off}}_{1\longrightarrow 0} & 0 \\ 0 & w^{\text{off}}_{1\longrightarrow 1} & 0 \\ 0 & 0 & w^{\text{off}}_{2\longrightarrow 2} \end{pmatrix} \begin{pmatrix} x_0 \\ x_1 \\ x_2 \end{pmatrix} = W^{\text{off}} \begin{pmatrix} x_0 \\ x_1 \\ x_2 \end{pmatrix} \tag{2}$$

Además, el nivel de PSA lo vamos a calcular con la siguiente función lineal:

$$\begin{pmatrix} c_0 & c_1 & c_2 \end{pmatrix} \begin{pmatrix} x_0 \\ x_1 \\ x_2 \end{pmatrix}$$

Donde por simplicidad hemos establecido que $c_0 = c_1 = c_2 = 1$, para así obtener el nivel de PSA dado por la siguiente fórmula:

$$\textbf{Nivel de PSA} = x_0 + x_1 + x_2 \tag{3}$$

Aunque más adelante vamos a definir y establecer otra función de costes que depende directamente de esta (ver: 3.1):

$$x_0^2 + x_1^2 + x_2^2$$

.

Antes de comenzar, vamos a establecer las siguientes hipótesis de nuestro modelo:

- Todas las ecuaciones del modelo son lineales durante ambas etapas del tratamiento.
- Los niveles de PSA no vienen dados explícitamente por el modelo.
- Las células están sometidas tanto a cambios reversibles como irreversibles.

Así, ya podemos continuar con nuestro modelo, que según [Hirata et al. (2012)], éste puede tener complicaciones cara a casos clínicos, como es el nuestro, debido a que necesitamos predecir si el paciente va a recaer o no sin contar con esta información previa entre nuestros datos. Para evitar este problema, imponemos las siguientes restricciones:

1. Los parámetros que representan las transiciones de un tipo de célula a otra son no negativos:
$$w^{\text{off}}_{0\to 1} \geqslant 0, w^{\text{off}}_{1\to 2} \geqslant 0, w^{\text{off}}_{0\to 2} \geqslant 0, w^{\text{off}}_{1\to 0} \geqslant 0$$
2. Cada tipo de célula puede variar como mucho un 20 % de su volúmen al día, es decir:
$$\left| \sum_{\substack{i,j\in\{0,1,2\} \\ m\in\{\text{on, off}\}}} w^{m}_{i\to j} \right| < 0{,}2$$
3. Los niveles de PSA se reestablecen con el tiempo, si se aplica la terapia de forma continuada (**CAS: Supresión de andrógenos continua**):
$$w^{\text{off}}_{2\to 2} > 0 \tag{4}$$
4. La mayoría de pacientes tratados con CAS, recaen con el tiempo.

A partir de esto, podemos retrasar la recaída minimizando el máximo valor propio de la matriz, es decir, minimizando la tasa de crecimiento más rápida de uno de los tipos de células cancerígenas, sin embargo y para minimizar el gran coste computacional que requiere, [Hirata et al. (2010b)] propone considerar un periodo de tratamiento programado de duración T, en el que se sucederán varios ciclos de duración $t_1 + t_2$, cada uno, y buscar el ratio óptimo entre ambas fases.

Nota 1 *A partir de ahora denominamos* ***ciclo*** *a un periodo de longitud "$t_1 + t_2$", que comprende ambas fases, la fase de terapia hormonal y la de reposo, con su duración "t_1" y "t_2", respectivamente.*

Como resultado, de este modelo obtenemos soluciones para un instante t del tratamiento, de la forma:

$$\begin{pmatrix} x_0(t) \\ x_1(t) \\ x_2(t) \end{pmatrix} = W(S_{0,\ T}) \begin{pmatrix} x_0(0) \\ x_1(0) \\ x_2(0) \end{pmatrix} \tag{5}$$

donde $W(S_{0,T})$ indica la matriz dependiente del tratamiento durante un periodo de tiempo programado que comienza en un instante "a" y termina en un instante "b", denotado por $\mathbf{S_{a,b}}$, en nuestro caso 0 y T, respectivamente, con T el tiempo total de duración del tratamiento de IAS, incluyendo sus ciclos correspondientes, y, por tanto, $W(S_{0,T})$ puede expresarse como:

$$W(S_{0,\ T}) = W(S_{N\cdot(t_1+t_2),\ T})(W(S_{0,\ t_1+t_2}))^N$$

con N el número de ciclos completos que se suceden durante un tratamiento de duración T, y donde para un N suficientemente grande, vemos que $W(S_{0,T})$ viene dominada por:

$$W(S_{0,\ t_1+t_2}) = W^{\text{off}}(S_{t_1,\ t_1+t_2})W^{\text{on}}(S_{0,\ t_1})$$

W^{on} y W^{off} son los de las ecuaciones (1) y (2) respectivamente.

2.3. Clasificación de los pacientes

Continuando con la elaboración del modelo, lo siguiente es clasificar a los pacientes [Hirata et al. (2012)] y [Hirata et al. (2010a)] ignorando los elementos no diagonales y siguiendo dos criterios:

1. $\mathbf{w^{\text{off}}_{2\to 2} < 0}$: Teniendo en cuenta que los términos no diagonales se ignoran, entonces:

   $$\frac{d}{dt}x_2 \approx w^{\text{on}}_{2\to 2}x_2$$

   durante la fase de tratamiento, y que

   $$\frac{d}{dt}x_2 = w^{\text{off}}_{2\to 2}x_2,$$

   durante la de reposo, y como $w^{\text{on}}_{2\to 2} > 0$, por hipótesis (4),entonces

   $$\frac{d}{dt}x_2 \downarrow \ si\ w^{\text{off}}_{2\to 2} < 0$$

   obteniendo así el **grupo de pacientes de tipo 1**, es decir en los que no habrá recaída.

   En el resto de pacientes no se evitará la recaída:

2. Si $w^{\text{off}}_{0\to 0} < w^{\text{on}}_{0\to 0}$ *o* $w^{\text{off}}_{1\to 1} < w^{\text{on}}_{1\to 1}$ *o* $w^{\text{off}}_{2\to 2} < w^{\text{on}}_{2\to 2}$, entonces al menos uno de los tres tipos de células disminuye durante el tratamiento, retrasando así la recaída, siendo estos los **pacientes de tipo 2.**

3. Si 2 no se cumple, significa que al menos un tipo de células crece más lento durante los periodos de tratamiento, siendo entonces más efectivo la modalidad de tratamiento continuada (CAS), **pacientes de tipo 3**

Finalmente concluimos esta sección mencionando que nos vamos a centrar únicamente en los pacientes de **tipo 1**, por simplicidad al no tener que contar con la recaída del paciente, lo que nos va a permitir simplificar los cálculos. A continuación, vamos a desarrollar la solución analítica del modelo (**Sección: 2.4**), que usaremos como restricción para nuestro problema de optimización, seguida de varias simulaciones de Matlab (**Sección: 2.5**).

2.4. Solución analítica del modelo como restricción

Para comenzar, antes del planteamiento del problema de optimización, vamos a introducir las soluciones del modelo que vamos a dar como solución de un sistema de ecuaciones en diferencias, como indica [Hirata et al. (2010a)], que representa los niveles de PSA para cada instante t durante el tratamiento:

$$\begin{pmatrix} x_0(t+\Delta t) \\ x_1(t+\Delta t) \\ x_2(t+\Delta t) \end{pmatrix} = \begin{pmatrix} d_{0,0}^{\text{on}} & 0 & 0 \\ d_{0,1}^{\text{on}} & d_{1,1}^{\text{on}} & 0 \\ d_{0,2}^{\text{on}} & d_{1,2}^{\text{on}} & d_{2,2}^{\text{on}} \end{pmatrix} \begin{pmatrix} x_0(t) \\ x_1(t) \\ x_2(t) \end{pmatrix} = D^{\text{on}} \begin{pmatrix} x_0(t) \\ x_1(t) \\ x_2(t) \end{pmatrix} \tag{6}$$

durante el periodo de terapia hormonal, y

$$\begin{pmatrix} x_0(t+\Delta t) \\ x_1(t+\Delta t) \\ x_2(t+\Delta t) \end{pmatrix} \begin{pmatrix} d_{0,0}^{\text{off}} & d_{1,0}^{\text{off}} & 0 \\ 0 & d_{1,1}^{\text{off}} & 0 \\ 0 & 0 & d_{2,2}^{\text{off}} \end{pmatrix} \begin{pmatrix} x_0(t) \\ x_1(t) \\ x_2(t) \end{pmatrix} = D^{\text{off}} \begin{pmatrix} x_0(t) \\ x_1(t) \\ x_2(t) \end{pmatrix} \tag{7}$$

durante los periodos de reposo.

Con:

- $d_{i,j}^m = W_{i\to j}^m \Delta t \qquad si\ i \neq j$
- $d_{i,j}^m = 1 + W_{i\to j}^m \Delta t \qquad si\ i = j$

Así, por [Hirata et al. (2010b), III. Analytical Solution], y asumiendo que:

$$d_{0,0}^{\text{on}} \neq d_{1,1}^{\text{on}},\ d_{1,1}^{\text{on}} \neq d_{2,2}^{\text{on}}\ y\ d_{0,0}^{\text{on}} \neq d_{1,1}^{\text{on}}$$

$$d_{0,0}^{\text{off}} \neq d_{1,1}^{\text{off}}$$

con $\Delta t = 1$ obtenemos la solución para los periodos de terapia hormonal y de reposo, respectivamente, en función a los valores iniciales ($\mathbf{X_0}$).

Como esta parte ya se realizó en [TFG M. Carnicero (2021)], y para avanzar más rápidamente, vamos a resumir el proceso para llegar a las soluciones:

Durante la fase de tratamiento, por (6), sabemos que:

$$X(1) = D^{\text{on}} X_0$$
$$X(2) = D^{\text{on}} X(1) = (D^{\text{on}})^2 X_0$$
$$\vdots$$
$$X(t) = (D^{\text{on}})^t X_0$$

Y así llegamos a:

$$\underbrace{\begin{pmatrix} x_0(t) \\ x_1(t) \\ x_2(t) \end{pmatrix}}_{X(t)} = \underbrace{\begin{pmatrix} (d_{0,0}^{\text{on}})^t & 0 & 0 \\ d_{0,1}^{\text{on}} \frac{(d_{0,0}^{\text{onon}})^t - (d_{1,1}^{\text{on}})^t}{d_{0,0}^{\text{on}} - d_{1,1}^{\text{on}}} & (d_{1,1}^{\text{on}})^t & 0 \\ \frac{(d_{0,0}^{\text{on}})^t - (d_{2,2}^{\text{on}})^t}{d_{0,0}^{\text{on}} - d_{2,2}^{\text{on}}} (d_{0,2}^{\text{on}} + \frac{d_{1,2}^{\text{on}} d_{0,1}^{\text{on}}}{d_{0,0}^{\text{on}} - d_{1,1}^{\text{on}}}) + \frac{d_{(1,1)}^{\text{on}})^t - (d_{2,2}^{\text{on}})^t}{d_{1,1}^{\text{on}} - d_{2,2}^{\text{on}}} d_{0,1}^{\text{on}} d_{1,2}^{\text{on}} \frac{-1}{d_{0,0}^{\text{on}} - d_{1,1}^{\text{on}}} & \frac{d_{(1,1)}^{\text{on}})^t - (d_{2,2}^{\text{on}})^t}{d_{1,1}^{\text{on}} - d_{2,2}^{\text{on}}} d_{1,2}^{\text{on}} & (d_{2,2}^{\text{on}})^t \end{pmatrix}}_{(D^{\text{on}})^t} \cdot \underbrace{\begin{pmatrix} x_0(0) \\ x_1(0) \\ x_2(0) \end{pmatrix}}_{X_0} \tag{8}$$

Finalmente, para la fase de tratamiento, como tenemos que su duración va a ser t_1, entonces:

$$X(t_1) = (D^{\text{on}})^{t_1} X_0$$

Aplicando el mismo razonamiento, para la fase de reposo obtenemos que:

$$\underbrace{\begin{pmatrix} x_0(t) \\ x_1(t) \\ x_2(t) \end{pmatrix}}_{X(t)} = \underbrace{\begin{pmatrix} (d_{0,0}^{\text{off}})^t & d_{1,0}^{\text{off}} \frac{(d_{1,1}^{\text{off}})^t - (d_{0,0}^{\text{off}})^t}{d_{1,1}^{\text{off}} - d_{0,0}^{\text{off}}} & 0 \\ 0 & (d_{1,1}^{\text{off}})^t & 0 \\ 0 & 0 & (d_{2,2}^{\text{off}})^t \end{pmatrix}}_{(D^{\text{off}})^t} \underbrace{\begin{pmatrix} x_0(t_1) \\ x_1(t_1) \\ x_2(t_1) \end{pmatrix}}_{X_{t_1}} \tag{9}$$

Y como esta fase está comprendida entre $(t_1, t_1 + t_2]$, su duración va a ser t_2, así:

$$X(t_1 + t_2) = (D^{\text{off}})^{t_2} X_{t_1}$$

Por tanto, teniendo en cuenta (8) y (9), y por (5), sabemos que:

$$W(S_{0,\ T}) = W(S_{N(t_1+t_2),\ T})(W(S_{0,\ t_1+t_2}))^N$$

donde para un N suficientemente grande, tenemos que:

$$W(S_{0,\ t_1+t_2}) = W^{\text{off}}(S_{t_1,\ t_1+t_2}) W^{\text{on}}(S_{0,\ t_1})$$

Finalmente vamos a concluir esta sección con el desarrollo de $x_0,\ x_1\ y\ x_2$ en función de $t_1\ y\ t_2$ variables que queremos optimizar.

En la siguiente sección vamos a implementar las simulaciones mencionadas anteriormente, y analizar los resultados obtenidos.

2.5. Simulaciones numéricas del modelo de tratamiento para pacientes de tipo 1.

Utilizando los parámetros obtenidos de [Hirata et al. (2010a)] para el tipo de pacientes 1, el modelo quedaría tal que:

$$\begin{pmatrix} x_0(t) \\ x_1(t) \\ x_2(t) \end{pmatrix} = \begin{pmatrix} 0{,}956156 & 0 & 0 \\ 0{,}00130716 & 0{,}995718 & 0 \\ 5{,}8852 \cdot 10^{-17} & 9{,}01608 \cdot 10^{-17} & 1{,}00181 \end{pmatrix} \begin{pmatrix} x_0(t-1) \\ x_1(t-1) \\ x_2(t-1) \end{pmatrix} \tag{10}$$

para el periodo de tratamiento, y para el de reposo:

$$\begin{pmatrix} x_0(t) \\ x_1(t) \\ x_2(t) \end{pmatrix} = \begin{pmatrix} 1{,}00223 & 0{,}1000 & 0 \\ 0 & 1{,}00392 & 0 \\ 0 & 0 & 0{,}8000 \end{pmatrix} \begin{pmatrix} x_0(t-1) \\ x_1(t-1) \\ x_2(t-1) \end{pmatrix} \tag{11}$$

$$\begin{pmatrix} x_0(t) \\ x_1(t) \\ x_2(t) \end{pmatrix} = \begin{pmatrix} 17{,}8395 \\ 0{,}77803 \\ 0{,}382485 \end{pmatrix}$$

Por **(3)**, sabemos que el nivel de PSA inicial viene dado por:

$$PSA_{inicial} = x_0(0) + x_1(0) + x_2(0) = 17,8395 + 0,77803 + 0,382485 = 19,000015$$

Como podemos ver, se cumplen las hipótesis y restricciones, además de los criterios de los pacientes de tipo 1, pues $w_{2,2}^{\text{off}} < 0 \Rightarrow d_{2,2}^{\text{off}} < 1$ y $d_{2,2}^{\text{off}} = 0,8 < 1$)

En primer lugar, hemos hecho una simulación de los niveles de PSA durante un tratamiento de **(CAS)**, obteniendo los siguientes resultados:

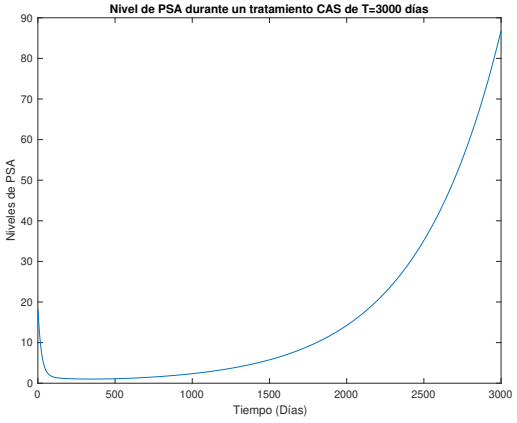

Figura 2: Evolución de un tratamiento continuado (CAS) durante un periodo de T=3000 días, con un nivel de PSA dado por (**3**). Durante el cual los niveles de PSA acaban aumentando hasta niveles muy altos, a medida que va avanzando el tratamiento, por lo que es evidente que la aplicación del tratamiento sin un periodo de descanso no es muy eficaz en este tipo de pacientes, siendo incluso contraproducente

Finalmente hemos realizado varias simulaciones numéricas con matlab, en las que vamos a reproducir los niveles de PSA durante un tratamiento de IAS para tiempos distintos de los periodos de tratamiento (t_1) y reposo (t_2) a lo largo de un periodo T fijado, en nuestro caso de $\mathbf{T = 3000}$ **días**, como vemos a continuación:

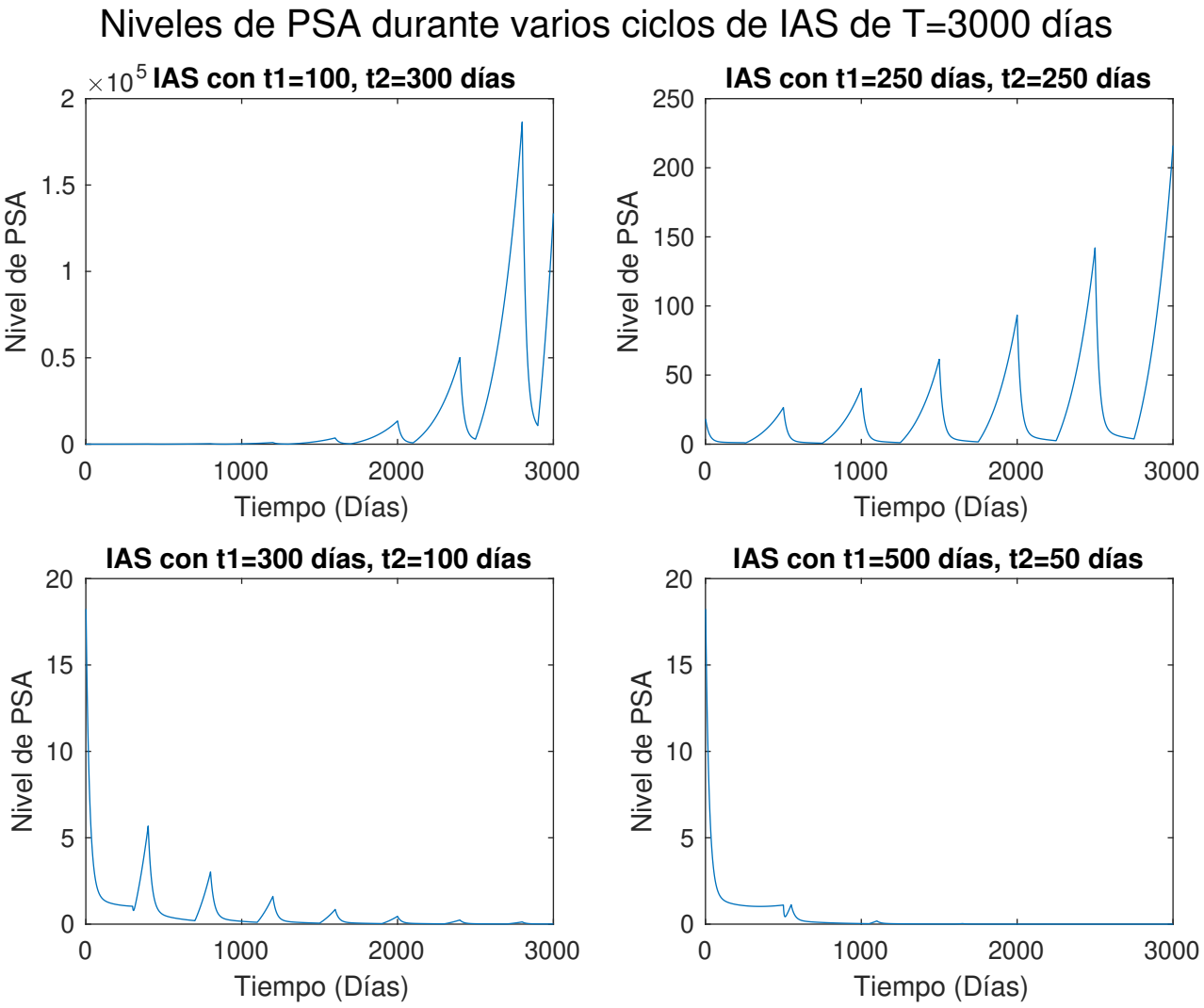

Figura 3: Como podemos observar, según la duración de cada fase durante un ciclo, los resultados del tratamiento varían. En las simulaciones superiores, vemos que no se logra disminuir el nivel de PSA, sino que éste aumenta llegando a adoptar niveles de PSA muy superiores a los iniciales. Por otro lado en las inferiores, el tratamiento es bastante óptimo, disminuyendo e incluso terminando con unos niveles de PSA prácticamente nulos, frenando la enfermedad. En estos casos, el paciente llega a superar el cáncer sin riesgo de recaída.

Por tanto, al concluir esta sección hemos visto el modelo que aproxima al tratamiento, su solución desde el punto de vista analítico y unas simulaciones numéricas que permiten reflejar cuán importante es fijar unos tiempos adecuados, dando paso al siguiente apartado.

3. Optimización del tratamiento

Teniendo ya definidos el modelo cuantitativo y la clasificación de pacientes y en vista a la importancia que tiene elegir unos tiempos adecuados, en esta sección, nuestro objetivo va a ser optimizar los parámetros t_1 *y* t_2 que indican la duración del período de terapia y reposo, respectivamente, considerando un programa de tratamiento periódico con ciclos de duración $(t_1 + t_2)$ para un tiempo total fijado (T).

3.1. Elección del funcional de costes a optimizar

Como lo que queremos optimizar realmente son los tiempos de duración de cada fase de tratamiento, durante un tiempo T, en el que se van a transcurrir varios ciclos de tratamiento(terapia hormonal y reposo), esto puede traducirse como intentar minimizar el nivel de PSA total durante cada ciclo, en un intervalo T de tratamiento durante el que se sucederán varios ciclos, para unos tiempos t_1 *y* t_2 (tiempos de terapia y reposo respectivamente), lo que matemáticamente se puede expresar por (3) como:

$$F(t_1, t_2) = \sum_{j=1}^{T} \left(x_0(j) + x_1(j) + x_2(j)\right)$$

y también de la forma siguiente:

$$F(t_1, t_2) = \sum_{j=1}^{T} \left(x_0(j)^2 + x_1(j)^2 + x_2(j)^2\right)$$

donde cada $x_i(j), i \in \{0, 1, 2\}, \forall j \in [1, T]$ son las soluciones del problema en el instante $t = j$, dadas por (5).

Ambas pueden servirnos como función de costes de nuestro problema de optimización [Carpio et al. (2022)].

Sea $N \in \mathbb{Z}$ el mayor entero tal que $N(t_1 + t_2) \leq T$, entonces el funcional se puede ver como la suma de los niveles de PSA durante cada ciclo de tratamiento (n=1,...,N), hasta llegar al último ciclo, teniendo también en cuenta el intervalo $(N(t_1+t_2),\ T]$. Además, como cada ciclo de tratamiento viene dado por las ecuaciones del modelo presentadas en ((8) y (9)), entonces, nuestro coste viene dado por cualquiera de estas dos ecuaciones:

$$C(t_1, t_2) = \min_{t_1, t_2 \in \mathbb{R}^+} F(t_1, t_2) = \min_{t_1, t_2 \in \mathbb{R}^+} \left(\sum_{j=1}^{T} \left(x_0(j)^2 + x_1(j)^2 + x_2(j)^2\right) \right) \tag{12}$$

$$C(t_1, t_2) = \min_{t_1, t_2 \in \mathbb{R}^+} F(t_1, t_2) = \min_{t_1, t_2 \in \mathbb{R}^+} \left(\sum_{j=1}^{T} \left(x_0(j) + x_1(j) + x_2(j)\right) \right) \tag{13}$$

En ambos desarrollos, obtenemos unos tiempos óptimos más o menos similares, aunque los niveles de PSA si varían de uno a otro por estar elevados a 2 en (13).

3.2. Técnicas de minimización del funcional

Para hallar la duración óptima de cada fase de tratamiento, así como la de un periodo completo(incluyendo las dos fases, cuya duración viene dada por (t_1+t_2) debemos minimizar el funcional introducido en la sección anterior. Como se trata de una función multivariable, vamos a introducir dos métodos que vamos a utilizar para optimizar el tratamiento:

- Método del descenso de gradiente
- Método multivariable de Newton-Raphson

3.2.1. Método de descenso de gradiente

Se trata de un método iterativo de primer orden, que se utiliza para encontrar un máximo o un mínimno local de una función dada, en nuestro caso (12) y/o (13).

Como requisitos previos, la función sobre la que se va a aplicar el método debe ser diferenciable y convexa, ya que si no, el algoritmo no garantiza que se encuentre el mínimo global de la función, sino que éste puede quedar "estancado" en un mínimo local.

Una vez visto esto, el método consiste en obtener el gradiente de la función y calcular iterativamente el siguiente punto usando el gradiente en la posición actual, lo escala (por una tasa de aprendizaje) y resta el valor obtenido de la posición actual (hace un paso). Resta el valor porque queremos minimizar la función (maximizarla sería sumar). Este valor de aprendizaje, que debe ser fijado por el usuario es el que nosotros denotaremos por α.

El proceso aplicado a nuestro caso concreto puede definirse de la forma:

$$\begin{pmatrix} t_{1_{n+1}} \\ t_{2_{n+1}} \end{pmatrix} = \begin{pmatrix} t_{1_n} \\ t_{2_n} \end{pmatrix} - \alpha \cdot \nabla C(t_1, t_2) \tag{14}$$

Con t_1 y t_2 tiempos de tratamiento y reposo definidos anteriormente, $C(t_1, t_2)$ la función de costes que se encarga de optimizar los tiempos de nuestro tratamiento, y su gradiente que hemos calculado utilizando la librería (**Symbolic matlab**) que permite hacer cálculos de gradiente con ecuaciones complejas como es nuestro caso. Por último en (A.2) tenemos el código que nos ha permitido optimizar el tratamiento.

En nuestro caso, los parámetros iniciales que hemos fijado son:

- Un tamaño de paso $\alpha = 0{,}5$.
- Un nº máximo de iteraciones: $max\_iter = 30000$.
- Un valor inicial de $t_0 = \begin{pmatrix} 0 \\ 0 \end{pmatrix}$.
- Una tolerancia: $tol = 1{,}0\exp -12$.

Finalizamos esta sección introduciendo los resultados obtenidos, con los que hemos visto, que en función de unos parámetros u otros elegidos, nuestro resultado va a variar, sin embargo, en todos los casos, hemos obtenido que t_1 debe ser bastante grande, mientras que t_2 debe ser en comparación muy pequeño.

Según varias de las simulaciones realizadas nuestros resultados fueron:

$$t = \binom{177}{3}, \text{con un nivel de PSA} = 986{,}15$$

Sin embargo, lo que nos hace plantearnos que la función no sea convexa, es que al probar algunos intervalos próximos al obtenido, obtenemos algún valor más pequeño del nivel de PSA, por lo que el mínimo obtenido no será un mínimo global, sino más bien un mínimo local.

Nota 2 *Por último, recalcar que estas simulaciones las hemos realizado sobre el funcional* ***(13)****, ya que era el único que admitía el valor inicial de ambos valores igual a 0.*

Sin embargo, si aplicamos el algoritmo a (12), obtenemos un resultado más óptimo, en términos de los niveles de PSA.

$$t = \binom{139}{5}, \textit{con un nivel de PSA} = 846{,}32$$

3.2.2. Método de Newton-Raphson

El método Newton-Raphson, o método Newton, es una potente técnica para resolver ecuaciones numéricamente. Como gran parte del cálculo diferencial, se basa en la simple idea de la aproximación lineal. El método de Newton, utilizado correctamente, suele llegar a una raíz con una eficacia devastadora.

Sea $f(x)$ una función bien definida y r una solución de la ecuación $f(x) = 0$. Sea x_0 una buena estimación de r, y sea $r = x_0 + h$, ya que la solución real es r y $h = r - x_0$, por tanto como h mide el error de la estimación y por hipótesis ésta era buena, entonces h es 'pequeña' pudiendo aproximarla por:

$$0 = f(r) = f(x_0 + h) \approx f(x_0) + hf'(x_0)$$

con

$$h \approx -\frac{f(x_0)}{f'(x_0)} \qquad si f'(x_0) \ncong 0$$

y por tanto llegamos a que:

$$r \approx x_0 - \frac{f(x_0)}{f'(x_0)}$$

Sea $x_1 = x_0 - \frac{f(x_0)}{f'(x_0)}$ una mejor estimación de r y siguiendo este razonamiento, llegamos a que:

$$x_{n+1} = x_n - \frac{f(x_n)}{f'(x_n)}$$

Y ésta es la fórmula en la que nos vamos a centrar, aunque por querer minimizar, necesitamos llegar hasta la $2^{\underline{a}}$ derivada, es decir vamos a alargar el proceso tal que:

$$f(x_{n+1}) \approx f(x_n) + f'(x_n)(x_{n+1} - x_n) + f''(x_n)(x_{n+1} - x_n)^2$$

que por el mismo razonamiento de antes, acaba siendo:

$$x_{n+1} = x_n - \frac{f'(x)}{f''(x)} \tag{15}$$

Y así por (15), en nuestro funcional al ser bidimensional obtenemos:

$$x_{n+1} = x_n - (H)^{-1}(\nabla F(t_1, t_2)) \tag{16}$$

donde ∇F es el gradiente del funcional F:

$$\nabla F = \left(\frac{\partial F}{\partial t_1}(t_1, t_2), \frac{\partial F}{\partial t_2}(t_1, t_2) \right)$$

y H es la matriz Hessiana:

$$H = \begin{pmatrix} \frac{d^2F}{dt_1^2}(t_1, t_2) & \frac{d^2F}{dt_1 dt_2}(t_1, t_2) \\ \frac{d^2F}{dt_2 dt_1}(t_1, t_2) & \frac{d^2F}{dt_2^2}(t_1, t_2) \end{pmatrix}$$

Sin embargo, como no está claro que nuestra función tenga definidas las segundas derivadas respecto de t_1 y t_2, entonces no vamos a aplicar éste método a nuestro funcional. Sin embargo, esta sección la incluimos, ya que en caso de sí existir éstas derivadas, es un método que bien se podría aplicar, y viene bien conocer.

4. Problema de optimización con redes neuronales

En esta sección vamos a dar un contexto sobre las redes neuronales, así como a mostrar el mismo problema de optimización del tratamiento de IAS, pero esta vez desde el punto de vista de una red neuronal que lo resuelva mediante el método de descenso de gradiente.

4.1. Contexto histórico de las redes neuronales.

En el 1943, Warren McCulloch y Walter Pitts crearon uno de los primeros modelos informáticos de redes neuronales artificiales conocido como logic umbral. Esto fue originando poco a poco la aplicación de estas redes neuronales en la Inteligencia Artificial.

De esta manera, una red neuronal artificial, consiste en un conjunto de unidades, que se llaman neuronas artificiales, conectadas entre sí para transmitirse señales. La información de entrada atraviesa la red neuronal, donde trabajan bajo ciertas operaciones lógicas, produciendo unos valores de salida.

Una importante ventaja de estos sistemas se debe a que aprenden y se forman a sí mismos, en lugar de ser programados de forma explícita. Con el fin de conseguir este aprendizaje automático, normalmente, se intenta minimizar una función de pérdida que evalúa la red en su total.

Una aplicación de estas tecnologías de aprendizaje automático, corresponde a la capacidad de resolución de ciertos problemas matemáticos, como puede ser problemas de optimización.

4.2. Aplicación a nuestro problema de optimización.

En la anterior sección, habíamos tratado de optimizar el tratamiento, una vez definidos el modelo cuantitativo y la clasificación de pacientes. Así, tratábamos de optimizar los parámetros t1 y t2 que indican la duración del período de terapia y reposo, respectivamente, considerando un programa de tratamiento periódico con ciclos de duración $(t_1 + t_2)$ dado tiempo total fijado T.

Entonces, definíamos correctamente, bajo ciertos argumentos, el funcional de costes a optimizar y, en último lugar, habíamos presentado dos técnicas matemáticas de resolución del problema de optimización.

A continuación vamos a tratar de resolver el problema de optimización propuesto, haciendo uso de técnicas de redes neuronales.

Para ello, vamos a seguir trabajando con una serie de datos iniciales asociados al paciente:

1. A_{on}: Matriz de transformación durante el tratamiento.
2. A_{off}: Matriz de transformación durante el reposo.
3. X_0: Vector de datos iniciales del paciente.
4. T: Período temporal máximo.

4.3. Arquitectura y desarrollo de la red neuronal

Como ya habíamos dicho, una red neuronal es un modelo computacional que utiliza una arquitectura de capas y algoritmos de aprendizaje para realizar predicciones a partir de una serie de datos de entrada, los cuales tenemos y con los cuales trabajaremos. La red neuronal se entrena ajustando los pesos y los sesgos en función del error entre las predicciones y los valores reales. Una vez entrenada, la red neuronal puede hacer predicciones sobre nuevos datos, los cuales generamos aleatoriamente y esto nos permite resolver el problema de optimización que planteamos.

4.3.1. Arquitectura

Nuestra red neuronal está basada en una arquitectura **feedforward** o alimentación directa. Siendo esta una de las arquitecturas más básicas de redes neuronales. En este tipo de red, la información se propaga en una sola dirección, desde la capa de entrada (Inputs) hacia la capa de salida (Outputs), sin poseer ciclos ni conexiones retroactivas. Cada neurona en una capa está directamente conectada a todas las de la capa anterior y no hay conexiones entre neurona de la misma capa ni de capas anteriores.

Esta arquitectura consta de tres tipos de capas:

- Capa de entrada: Recibe nuestros datos de entrada y se encarga de transmitir la información a la siguiente capa. Cada una de estas neuronas en esta capa representa una característica de los datos de entrada. Nuestra capa de entrada de la red neuronal tiene dos neuronas, que corresponden a los valores de t1 y t2.
- Capas ocultas: Estas capas están situadas entre la capa de entrada y la capa de salida. Cada neurona en estas capas realiza cálculos sobre los valores recibidos de las capas anteriores. Cada capa oculta puede contener múltiples neuronas, y la cantidad de estas puede ser ajustada. En nuestro caso, hemos tomado 10 neuronas.
- Capa de salida: Esta capa es la última capa de la red neuronal y produce las predicciones que estamos buscando. Cada neurona en esta capa representa una salida que estamos tratando de predecir. En nuestro caso hemos tomado una única neurona que representa la predicción del valor de PSA.

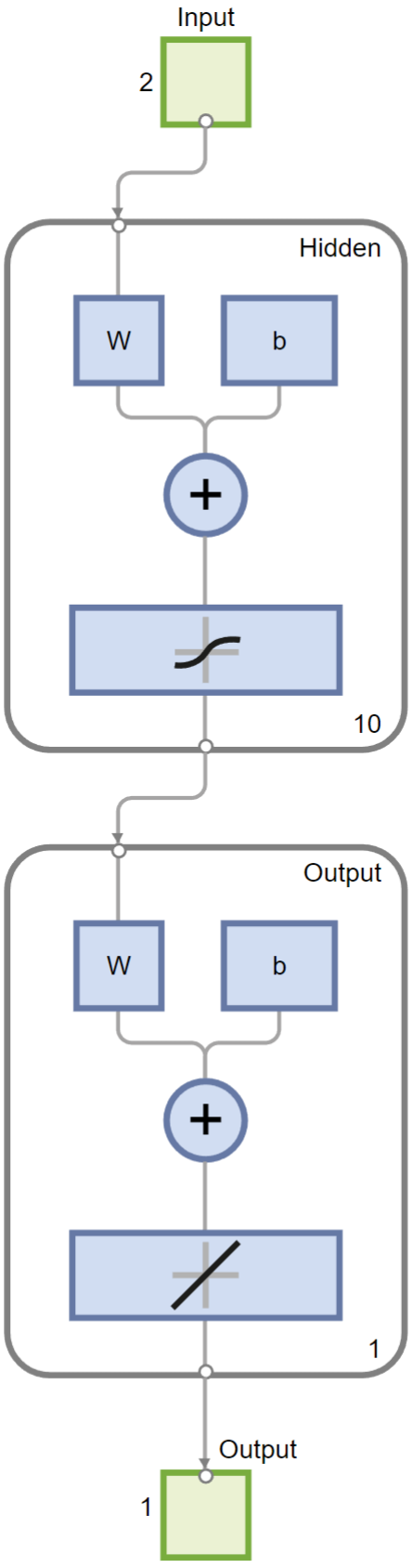

Figura 4: Arquitectura de la red neuronal feedforward, con una capa oculta.

Inicialización de pesos y sesgos

En la red neuronal, los pesos se inicializan aleatoriamente permitiendo que ésta comience el proceso de entrenamiento sin la existencia de sesgos.

Propagación hacia adelante (forward propagation)

Durante la etapa de entrenamiento la información fluye desde la capa de entrada hacia la capa oculta y, finalmente, hacia la capa de salida. Cada neurona en la capa oculta realiza una combinación lineal de las entradas y aplica una función de activación la cual es no lineal. La salida de la capa oculta se utiliza como entrada para la neurona de salida, que también aplica una función de activación no lineal. La salida final de la red neuronal es el valor de PSA que estamos tratando de predecir.

Cálculo del error y ajuste de los pesos

Después de la propagación hacia adelante, se compara la salida predicha de la red neuronal con el valor real de PSA. Se calcula un valor de error que representa la discrepancia entre la predicción y el valor real. El objetivo del entrenamiento es minimizar este error para que las predicciones de la red neuronal sean lo más precisas posible. El algoritmo de retropropagación (**backpropagation**) se utiliza para calcular cómo los pesos y los sesgos de la red deben ajustarse para reducir el error.

Visualizando las distintas etapas de la red neuronal, obtenemos:

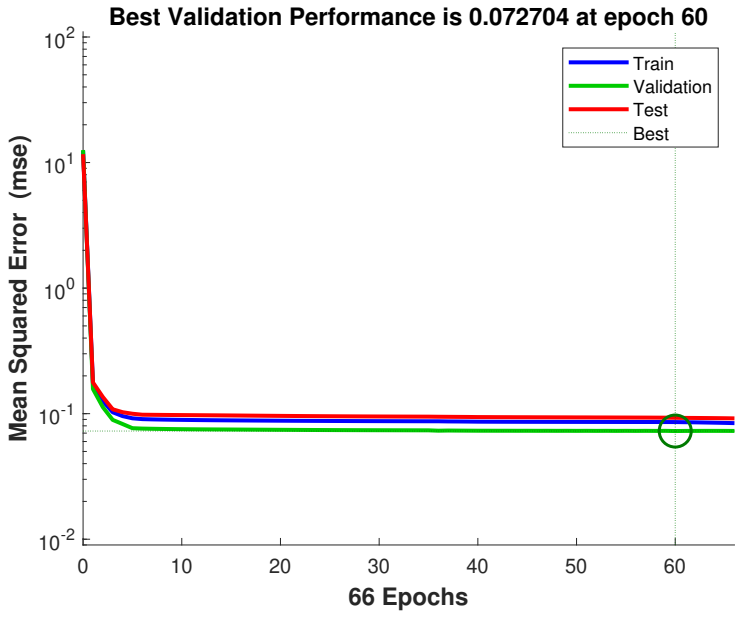

Figura 5: Rendimiento de la red neuronal, durante su periodo de entrenamiento, validación y test, utilizando el error cuadrático medio.

Predicción y evaluación

Después de completar el entrenamiento, la red neuronal se puede utilizar para hacer predicciones sobre nuevos datos. En nuestro caso, como ya se ha comentado, generamos un conjunto de valores de t1 y t2 para la predicción. Estos valores se pasan a través de la

red neuronal entrenada utilizando la propagación hacia adelante. La red neuronal produce una predicción del nivel de PSA correspondiente a cada combinación de t1 y t2. Después de obtener las predicciones, podemos evaluar la calidad de la red neuronal comparando las predicciones con los valores reales de PSA. En nuestro caso, calculamos el valor óptimo de t1 y t2 correspondiente a la predicción que posee menor nivel de PSA.

4.3.2. Desarrollo técnico

Generación de valores de t_1 y t_2:
Dado que no tenemos un conjunto de datos organizados en una base de datos con los que trabajar, a partir de las técnicas de redes neuronales, podemos generar nosotros los datos y trabajar con ellos, con el fin de resolver muchas veces el problema y permitir ese aprendizaje de la red neuronal. Así, generaremos valores equidistantes de t_1 y t_2 en el rango de 0 a T. Esto lo conseguimos mediante la función linspace y se utiliza la raíz cuadrada de T más 1 para obtener una distribución uniforme.

Generación de combinaciones posibles de t_1 y t_2:
Utilizamos la función meshgrid para generar todas las combinaciones posibles de t1 y t2. Esto nos permite generar dos matrices de idéntica dimensión, donde cada elemento representa una combinación única de t1 y t2.

Cálculo del valor del funcional PSA correspondiente a cada combinación:
Como ya hemos definido antes, trabajaremos con la función IAS para calcular el valor absoluto del funcional PSA para cada combinación de t_1 y t_2. Con esos cálculos, almacenamos los resultados en la matriz PSA.

Trabajo con los datos de entrenamiento:
Los valores de t_1 y t_2 los concatenamos en la matriz inputs para ser utilizados como entradas de la red neuronal. Los valores de PSA son almacenados en el vector targets para ser utilizados como salidas de la red neuronal.

Construcción de la red neuronal:
Creamos una red neuronal *feedforward* utilizando la función *feedforwardnet*. El número de neuronas en la capa oculta es definido mediante la variable hidden-units. Vamos a tomar el comando net.trainParam.showWindow = true. Con ello, hacemos que se muestre la ventana de entrenamiento durante el proceso, lo que nos permite visualizar la estructura así como otras opciones para ver el entrenamiento, los errores y la regresión lineal correspondiente a los procesos de entrenamiento, validación y test.

Entrenamiento de la red neuronal:
Vamos a utilizar la función train para entrenar la red neuronal utilizando los datos de entrada inputs y las salidas esperadas que las habíamos tomado como targets. Entonces, almacenamos la red neuronal en la variable *net*.

Proceso de predicción del valor del funcional PSA para diferentes combinaciones de t_1 y t_2:

Generamos unos nuevos valores de t1 y t2 para realizar la predicción, generándolos como hemos hecho antes. Estos nuevos valores son combinados en una matriz; inputs-test y se utilizan como entrada para la red neuronal. Entonces, utilizamos la red neuronal ya entrenada con el fin de conseguir la predicción de los valores del funcional PSA para cada combinación de t_1 y t_2. Los resultados de estos cálculos los almacenamos posteriormente en un vector llamado *PSA-predicted*.

Búsqueda de los valores óptimos de t_1 y t_2:

Buscamos el valor mínimo del valor absoluto del funcional PSA predicho utilizando la función de MATLAB *min*.

Con todo esto, obtenemos el índice correspondiente al valor mínimo, lo cual es utilizado para obtener los valores óptimos de t_1 y t_2 de las respectivas matrices $t_1 - test$ y $t2 - test$.

El número de neuronas en la capa oculta es establecido mediante la variable *hidden-units*, que se puede ajustar según las necesidades del problema. Con ello, se consigue aumentar la capacidad de representación de la red neuronal, lo que podría conducir a un mejor rendimiento en la predicción de los valores óptimos.

Cabe destacar que se ha realizado un proceso de normalización y desnormalización que se utilizan para ajustar los valores de entrada y salida de una red neuronal dentro de un rango específico. Esto es beneficioso ya que mejora la convergencia y estabilidad del entrenamiento de la red neuronal.

Primero, normalizamos los datos de entrada y salida antes de ser utilizados para entrenar la red neuronal. En este caso, se utiliza la función *normalize* para normalizar los valores de entrada y salida en un rango común. Esta función escala los datos para que tengan una media de cero y una desviación estándar de uno. El fin de este proceso se debe a asegurar que los datos estén en una escala similar, evitando diferencias extremas en los valores que pueden dificultar el aprendizaje de la red neuronal. Posteriormente, los resultados normalizados se desnormalizan para obtener los valores originales. En el código, se utiliza la fórmula de desnormalización manualmente para revertir la normalización y obtener los valores desnormalizados. La desnormalización es necesaria para interpretar correctamente los resultados de la red neuronal en la escala original.

5. Interpretación de los resutados.

Tras ejecutar el problema de optimización tanto con la red neuronal como desde un problema de ecuaciones en diferencias, podemos observar, como ya habíamos comentado, que debido al carácter asintótico del modelo, en cada ejecución nos va a devolver valores distintos, pero siempre el que obtiene el mínimo nivel de PSA, y esto siempre se corresponde con períodos muy largos de reposo, acompañados con ciertos periodos de tratamiento que oscilan entre los 200 y los 300 días. Por ello con el fin de obtener siempre el mismo resultado, hemos añadido una random seed.

Finalmente, si ejecutamos la función que simula el tratamiento de IAS para los tiempos $\mathbf{t_1}$ **y** $\mathbf{t_2}$ obtenidos por la red neuronal, podemos comprobar si se tratan de valores óptimos que realmente minimicen el nivel de PSA para un intervalo fijo de T=3000 días. Para ello hemos visualizado una gráfica con los valores de PSA a medida que transcurren los 3000 días.

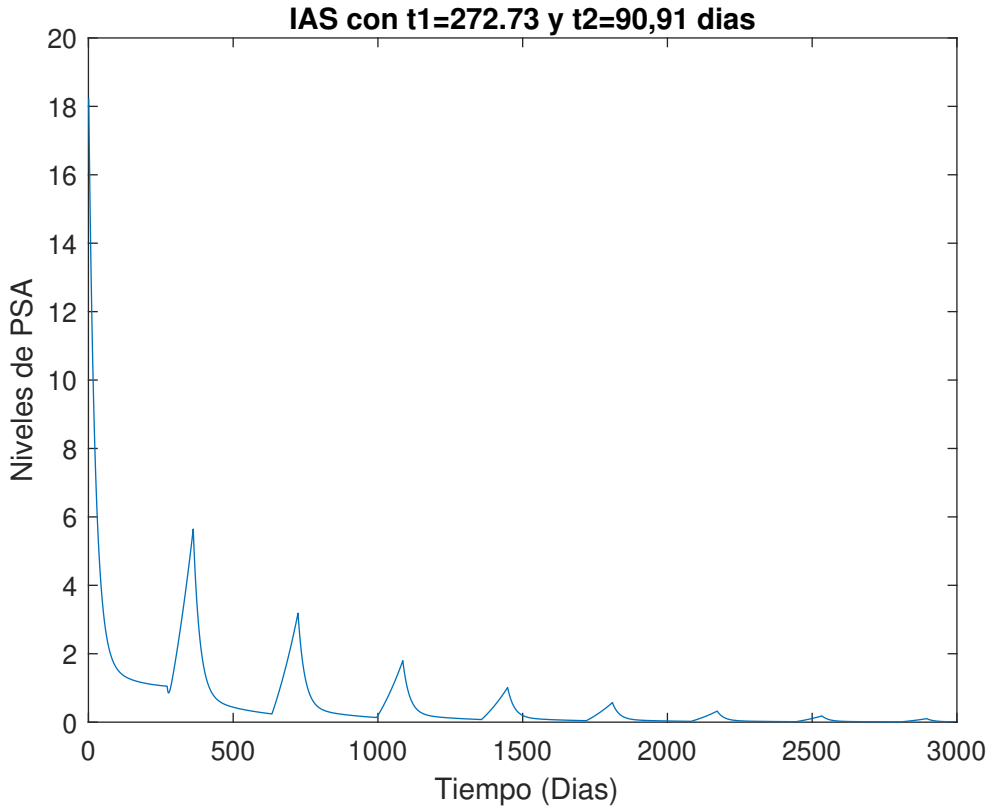

Figura 6: Se puede observar como para estos tiempos, los niveles de PSA van descendiendo, además se aprecia claramente cada intervalo de tratamiento y de reposo, pero al final de los 3000 dias podemos ver como los niveles son prácticamente nulos.

A. Códigos de Matlab

A.1. Simulaciones iniciales de terapia IAS y CAS

```
clear all;
clc;
clf;
%Datos paciente de tipo 1:

% Matriz durante el tratamiento:
format long
A_on=[0.956156 0 0;0.00130716 0.995718 0;5.8852*10^-17 9.01608*10^-17 1.00181];

% Matriz durante el reposo:
A_off=[1.00223 0.1 0;0 1.00392 0;0 0 0.8];

% Datos iniciales paciente:

X_0=[17.8395; 0.77803; 0.382485];

T=3000; %Por ejemplo

%Para esta simulacion usamos t1 y t2 del articulo de hirata (Hybrid
%optimal)
PSA1=IAS(A_on,A_off,X_0,T,300,100);
PSA2=IAS(A_on,A_off,X_0,T,250,250);
PSA3=IAS(A_on,A_off,X_0,T,500,50);
PSA4=IAS(A_on,A_off,X_0,T,100,300);
%Funcion para el tratamiento continuo:
for t=1:T
    x=A_on*X_0;
    X_0=x;
    CAS(t)=x(1)+x(2)+x(3);
end
hold on
sgtitle('Nivel de PSA durante un periodo de tratamiento con IAS de T=3000 dias')
subplot(4,1,1)
plot(1:T,PSA4)
xlabel("Tiempo (Dias)")
ylabel("Nivel de PSA")
title("Nivel de PSA bajo tratamiento de IAS con t1=100 dias, t2=300 dias")

subplot(4,1,4)
plot(1:T,PSA3)
xlabel("Tiempo (Dias)")
ylabel("Nivel de PSA")
title("Nivel de PSA bajo tratamiento de IAS con t1=500 dias, t2=50 dias")

subplot(4,1,3)
plot(1:T,PSA1)
xlabel("Tiempo (Dias)")
ylabel("Nivel de PSA")
title("Nivel de PSA bajo tratamiento de IAS con t1=300 dias, t2=100 dias")
```

```
subplot(4,1,2)
plot(1:T,PSA2)
xlabel("Tiempo (Dias)")
ylabel("Nivel de PSA")
title("Nivel de PSA bajo tratamiento de IAS con t1=250 dias, t2=250 dias")
close
hold off
%print -depsc simsIAS.eps
plot(1:T,CAS)
title("Nivel de PSA durante un tratamiento CAS de T=3000 dias")
xlabel("Tiempo (Dias)")
ylabel("Niveles de PSA")
%print -depsc simCAS.eps
%Funcion que calcula el tratamiento intermitente
function [PSA]=IAS(A_on,A_off,X_0,T,t1,t2)
N=floor(T/(t1+t2));
modul=mod(T,t1+t2);
t=1;
if modul==0
    for n=1:N
        for i=1:t1
            x=A_on*X_0;
            X_0=x;
            PSA(t)=x(1)+x(2)+x(3);
            t=t+1;
        end
        for i=1:t2
            x=A_off*X_0;
            X_0=x;
            PSA(t)=x(1)+x(2)+x(3);
            t=t+1;
        end
    end
else
    for n=1:N
        for i=1:t1
            x=A_on*X_0;
            X_0=x;
            PSA(t)=x(1)+x(2)+x(3);
            t=t+1;
        end
        for i=1:t2
            x=A_off*X_0;
            X_0=x;
            PSA(t)=x(1)+x(2)+x(3);
            t=t+1;
        end
    end
    if modul<=t1
        for i=1:modul
            x=A_on*X_0;
            X_0=x;
            PSA(t)=x(1)+x(2)+x(3);
            t=t+1;
        end
```

```
    else
        for i=1:t1
            x=A_on*X_0;
            X_0=x;
            PSA(t)=x(1)+x(2)+x(3);
            t=t+1;
        end
        for i=1:modul-t1
            x=A_off*X_0;
            X_0=x;
            PSA(t)=x(1)+x(2)+x(3);
            t=t+1;
        end
    end
end
end
```

A.2. Método de gradiente descendiente aplicado:

```
clear all;
clc;
%Datos paciente de tipo 1:

% Matriz durante el tratamiento:
format long
A_on=[0.956156 0 0;0.00130716 0.995718 0;5.8852*10^-17 9.01608*10^-17 1.00181];

% Matriz durante el reposo:
A_off=[1.00223 0.1 0;0 1.00392 0;0 0 0.8];

% Datos iniciales paciente:

X_0=[17.8395; 0.77803; 0.382485];

%Empezamos el programa para optimizar t1 y t2, con [0,t1] el primer
%intervalo de tratamiento, [t1+t2] el de reposo, ... hasta T periodo
%maximo establecido, que vamos a fijar en 300 dias, por ejemplo.

T=3000;

%ahora vamos a definir nuestras variables t1 y t2 de forma simbolica

syms t1 t2

%asi como otros parametros conocidos, o que dependen de nuestras variables:

%N=floor(T/(t1+t2)); %Dado en forma simbolica de t1 y t2
W_on=A_on^t1;

W_off=A_off^t2;

%W_sol=(W_off*W_on)^(T/(t1+t2));

X_t1=(W_on*X_0);
X_t2=(W_off*X_t1);
```

```
%--------------------------------J y grad para J =   x0  +x1+x2:

mixing = sum(X_t1+X_t2);
J=symfun(mixing,[t1,t2]);
grad = gradient(J,[t1,t2]);

%-----------------J y grad para J =   x0  ^2+x1^2+x2^2:----------------------

%mixing = sum(X_t1.^2+X_t2.^2);
%J=symfun(mixing,[t1,t2]);
%grad = symfun(gradient(J),[t1,t2]);

t0=[0,0];

[topt, fopt, niter, gnorm]=grad_descent(J, grad, t0, A_on, A_off, X_0, T)

function [topt,fopt,niter,gnorm,dt] = grad_descent(J, grad, t0, A_on, A_off, X_0, T)
% termination tolerance
tol = 1e-3;
% minimum allowed perturbation
dtmin = 1e-3;
% step size ( )
alpha = 0.1;
% initialize gradient norm, optimization vector, iteration counter, perturbation
gnorm = inf; t=t0;niter = 0; dt = inf; PSA = inf; t1s=[]; t2s=[];
% gradient descent algorithm:
while and(gnorm>=tol, dt >= dtmin)
    % calculate gradient:
    t1=t(1);
    t2=t(2);
    g = double(grad(t1,t2))
    gnorm = norm(g);
    % take step:
    tnew = t - alpha*g;
    % check step
    if ~isfinite(tnew)
        display(['Number of iterations: ' num2str(niter)])
        error('t is inf or NaN')
    end
    % update termination metrics
    dt = norm(tnew-t);
    niter = niter + 1;
    t = tnew;
    t1s(end+1)=t(1);
    t2s(end+1)=t(2);
end

for i=1:length(t1s)
     PSAs(i)=sum(IAS(A_on,A_off,X_0,T,round(t1s(i)),round(t2s(i))));
end
topt = round(t);
topt=round([t1s(find(PSAs==min(PSAs), 1))
```

```
        t2s(find(PSAs==min(PSAs), 1))]);
%fopt = J(topt(1), topt(2));
fopt = min(PSAs)
niter = niter - 1;
end


function [PSA]=IAS(A_on,A_off,X_0,T,t1,t2)
N=floor(T/(t1+t2));
modul=mod(T,t1+t2);
t=1;

for n=1:N
    for i=1:t1
        x=A_on*X_0;
        X_0=x;
        %PSA(t)=x(1)^2+x(2)^2+x(3)^2;
        PSA(t)=sum(x);
        t=t+1;
    end
    for i=1:t2
        x=A_off*X_0;
        X_0=x;
        %PSA(t)=x(1)^2+x(2)^2+x(3)^2;
        PSA(t)=sum(x);
        t=t+1;
    end
end

if modul~=0
    if modul<=t1
        for i=1:modul
            x=A_on*X_0;
            X_0=x;
            PSA(t)=sum(x);
            %PSA(t)=x(1)^2+x(2)^2+x(3)^2;
            t=t+1;
        end
    else
        for i=1:t1
            x=A_on*X_0;
            X_0=x;
            PSA(t)=sum(x);
            %PSA(t)=x(1)^2+x(2)^2+x(3)^2;
            t=t+1;
        end
        for i=1:modul-t1
            x=A_off*X_0;
            X_0=x;
            %PSA(t)=x(1)^2+x(2)^2+x(3)^2;
            PSA(t)=sum(x);
            t=t+1;
        end
    end
end
```

```
end
```

A.3. Red neuronal para optimización de tiempos (t_1 y t_2) de IAS.

```
clear all
clc
clf
close all

% Datos iniciales del problema
format long
A_on = [0.956156 0 0; 0.00130716 0.995718 0; 5.8852e-17 9.01608e-17 1.00181];
A_off = [1.00223 0.1 0; 0 1.00392 0; 0 0 0.8];
X_0 = [17.8395; 0.77803; 0.382485];
T = 3000;

% Genera los valores de t1 y t2 para el entrenamiento
t1_values_train = linspace(0, T, sqrt(T)+1);
t2_values_train = linspace(0, T, sqrt(T)+1);
[t1_train, t2_train] = meshgrid(t1_values_train, t2_values_train);
inputs_train = [t1_train(:), t2_train(:)];

% Calcula el valor del funcional PSA correspondiente a cada combinacion
PSA_train = abs(IAS(A_on, A_off, X_0, T, t1_train(:), t2_train(:)));

% Preparacion de los datos de entrenamiento
inputs_normalized_train = normalize(inputs_train);
targets_normalized_train = normalize(PSA_train);

% Construccion de la red neuronal
hidden_units = 10;  % Numero de neuronas en la capa oculta
net = feedforwardnet(hidden_units);
net.trainParam.showWindow = true;  % No mostrar la ventana de entrenamiento durante

% Semilla laeatoria
rng(7)

% Entrenamiento de la red neuronal
net = train(net, inputs_normalized_train.', targets_normalized_train.');

% Genera los valores de t1 y t2 para la prediccion
t1_values_pred = linspace(0, T, 100);
t2_values_pred = linspace(0, T, 100);
[t1_pred, t2_pred] = meshgrid(t1_values_pred, t2_values_pred);
inputs_pred = [t1_pred(:), t2_pred(:)];

% Normalizacion de los datos de prediccion
inputs_normalized_pred = normalize(inputs_pred);

% Prediccion del valor del funcional PSA para diferentes combinaciones de t1 y t2
PSA_predicted_normalized = net(inputs_normalized_pred.').';

% Desnormalizacion de los resultados de prediccion
min_targets_normalized = min(targets_normalized_train);
max_targets_normalized = max(targets_normalized_train);
```

```
min_targets = min(PSA_train);
max_targets = max(PSA_train);
PSA_predicted = abs((PSA_predicted_normalized - min_targets_normalized) * (max_targe

% Busqueda de los valores optimos de t1 y t2
[min_PSA, min_index] = min(PSA_predicted);
optimal_t1 = t1_pred(min_index);
optimal_t2 = t2_pred(min_index);
optimal_PSA = min_PSA;

% Mostrar resultados
fprintf('Valores optimos encontrados:\n');
fprintf('t1: %.2f\n', optimal_t1);
fprintf('t2: %.2f\n', optimal_t2);
fprintf('optimo: %.2f\n', optimal_PSA);


valores = IAS(A_on, A_off, X_0, T, optimal_t1, optimal_t2);
plot(1:length(valores) , valores)


function [PSA] = IAS(A_on, A_off, X_0, T, t1, t2)
    N = floor(T / (max(t1) + max(t2)));
    modul = mod(T, max(t1) + max(t2));
    t = 1;
    PSA = zeros(size(t1));

    if modul == 0
        for n = 1:N
            for i = 1:numel(t1)
                for j = 1:t1(i)
                    x = A_on * X_0;
                    X_0 = x;
                    PSA(t) = x(1) + x(2) + x(3);
                    t = t + 1;
                end
                for j = 1:t2(i)
                    x = A_off * X_0;
                    X_0 = x;
                    PSA(t) = x(1) + x(2) + x(3);
                    t = t + 1;
                end
            end
        end
    else
        for n = 1:N
            for i = 1:numel(t1)
                for j = 1:t1(i)
                    x = A_on * X_0;
                    X_0 = x;
                    PSA(t) = x(1) + x(2) + x(3);
                    t = t + 1;
                end
                for j = 1:t2(i)
                    x = A_off * X_0;
```

```
                    X_0 = x;
                    PSA(t) = x(1) + x(2) + x(3);
                    t = t + 1;
                end
            end
        end
        if modul <= max(t1)
            for i = 1:modul
                x = A_on * X_0;
                X_0 = x;
                PSA(t) = x(1) + x(2) + x(3);
                t = t + 1;
            end
        else
            for i = 1:max(t1)
                x = A_on * X_0;
                X_0 = x;
                PSA(t) = x(1) + x(2) + x(3);
                t = t + 1;
            end
            for i = 1:modul - max(t1)
                x = A_off * X_0;
                X_0 = x;
                PSA(t) = x(1) + x(2) + x(3);
                t = t + 1;
            end
        end
    end
end


% Esta es la funcion de mi codigo, sin ningun cambio.
% function [ PSA ]= IAS ( A_on , A_off , X_0 ,T , t1 , t2 )
%     N = floor ( T /( t1 + t2 ));
%     modul = mod (T , t1 + t2 );
%     t =1;
%     if modul ==0
%         for n =1: N
%             for i =1: t1
%                 x = A_on * X_0 ;
%                 X_0 = x ;
%                 PSA(t)= x (1)+ x (2)+ x (3);
%                 t = t +1;
%             end
%             for i =1: t2
%                 x = A_off * X_0 ;
%                 X_0 = x ;
%                 PSA(t)= x (1)+ x (2)+ x (3);
%                 t = t +1;
%             end
%         end
%     else
%         for n =1: N
%             for i =1: t1
%                 x = A_on * X_0 ;
```

```
%                 X_0 = x ;
%                 PSA(t)= x (1)+ x (2)+ x (3);
%                 t = t +1;
%             end
%             for i =1: t2
%                 x = A_off * X_0 ;
%                 X_0 = x ;
%                 PSA(t)= x (1)+ x (2)+ x (3);
%                 t = t +1;
%             end
%         end
%         if modul <= t1
%             for i =1: modul
%                 x = A_on * X_0 ;
%                 X_0 = x ;
%                 PSA(t)= x (1)+ x (2)+ x (3);
%                 t = t +1;
%             end
%         else
%             for i =1: t1
%                 x = A_on * X_0 ;
%                 X_0 = x ;
%                 PSA(t)= x (1)+ x (2)+ x (3);
%                 t = t +1;
%             end
%             for i =1: modul - t1
%                 x = A_off * X_0 ;
%                 X_0 = x ;
%                 PSA(t)= x (1)+ x (2)+ x (3);
%                 t = t +1;
%             end
%         end
%     end
% end
%
```

Referencias

[Aihara et al. (2010)] Aihara K. & Suzuki H. (2010). *Theory of hybrid dynamical systems and its applications to biological and medical systems.* Phil. Trans. R. Soc. A.3684893–4914. http://doi.org/10.1098/rsta.2010.0237

[TFG M. Carnicero (2021)] Carnicero M., Trabajo Fin de Grado. *Modelado del tratamiento de problemas de próstata con terapias hormonales.* UCM 2021

[Carpio et al. (2022)] Carpio A. & Pierret E. (2022). *Parameter identification in epidemiological models.* In Mathematical Analysis of Infectious Diseases (MAID 2020) 103-124, Praveen Agarwal, Juan J. Nieto and Delfim F. M. Torres Eds., Elsevier https://doi.org/10.1016/B978-0-32-390504-6.00012-7

[Hirata et al. (2010a)] Hirata, Y., Bruchovsky, N., & Aihara, K. (2010). *Development of a mathematical model that predicts the outcome of hormone therapy for prostate cancer.* Journal of theoretical biology, 264(2), 517–527. https://doi.org/10.1016/j.jtbi.2010.02.027

[Hirata et al. (2010b)] Hirata, Y., di Bernardo, M., Bruchovsky, N., & Aihara, K. (2010). *Hybrid optimal scheduling for intermittent androgen suppression of prostate cancer.* Chaos, 20(4), 045125. https://doi.org/10.1063/1.3526968

[Hirata et al. (2012)] Hirata Y., Akakura K., Higano C. S., Bruchovsky N., & Aihara K. (2012). *Quantitative mathematical modeling of PSA dynamics of prostate cancer patients treated with intermittent androgen suppression.* Journal of molecular cell biology, 4(3), 127–132.

[TFG M. Morales (2017)] María Morales. TFG: *Ocurrencia de cáncer de prostata y factores predictores en un cohorte de nuevos usuarios de aspirina y un cohorte de comparación*. UCM 2017.

[Morten et al] Morten Blomhøj. *Modelización Matemática - Una Teoría para la Práctica.* Traducción autorizada de Blomhoj, M. (2004) Mathematical modelling - A theory for practice. En Clarke, B.; Clarke, D. Emanuelsson, G.; Johnansson, B.; Lambdin, D.; Lester, F. Walby, A. Walby, K. (Eds.) International Perspectives on Learning and Teaching Mathematics. National Center for Mathematics Education. Suecia, p. 145-159. http://funes.uniandes.edu.co/19254/1/Biomhaj2008Modelizacion.pdf

[Ruiz et al. (2017)] Ruiz López A.I., Pérez Mesa J.C., Cruz Batista Y., González Lorenzo L.E, *Actualización sobre cáncer de próstata.*

[Wikipedia] *Cáncer de próstata, Detección precoz, Antígeno prostático específico(PSA).*