Trabajo Fin de Grado
Grado en Ingeniería de las Tecnologías Industriales

Optimización del tratamiento Denosumab para la osteoporosis

Autor: Rocío Ruiz Lozano

Tutor: Jose Luis Calvo Gallego

Dep. de Ingeniería Mecánica y Fabricación
Escuela Técnica Superior de Ingeniería
Universidad de Sevilla

Sevilla, 2020

Trabajo Fin de Grado
Grado en Ingeniería de las Tecnologías Industriales

Optimización del tratamiento Denosumab para la osteoporosis

Autor:
Rocío Ruiz Lozano

Tutor:
Jose Luis Calvo Gallego
Profesor Ayudante Doctor

Dep. de Ingeniería Mecánica y Fabricación
Escuela Técnica Superior de Ingeniería
Universidad de Sevilla

Sevilla, 2020

Trabajo Fin de Grado: Optimización del tratamiento Denosumab para la osteoporosis

Autor: Rocío Ruiz Lozano
Tutor: Jose Luis Calvo Gallego

El tribunal nombrado para juzgar el trabajo arriba indicado, compuesto por los siguientes profesores:

Presidente:

Vocal/es:

Secretario:

acuerdan otorgarle la calificación de:

El Secretario del Tribunal

Fecha:

Agradecimientos

En primer lugar, me gustaría agradecer a mi familia todo el apoyo y ánimos que me han regalado todos estos años de estudio, en particular a mis padres, Elena y Patricio. También me gustaría agradecer, aunque solo fueran unas pequeñas líneas a mis profesores, que tanto me han enseñado en mi trayectoria académica, tanto personalmente como profesionalmente.
A mis amigos, que han estado conmigo estos cuatro años, con sus alegrías y disgustos.
Por último, en especial a Dani, por haber sido un gran apoyo y ejemplo durante todo el grado.

Rocío Ruiz Lozano
Sevilla, 2020

Resumen

En la actualidad, el incremento de personas de la tercera edad, ha llevado consigo un aumento de mujeres que padecen la osteoporosis postmenopáusica, una enfermedad que produce un gran impacto, aumentando el número de fracturas y el gasto sanitario. Esta enfermedad puede ser tratada con fármacos anabólicos (e.g. teriparatida) o anticatabólicos (e.g. los bifosfonatos o el Denosumab). Este trabajo se centra en el estudio de este último.

El Denosumab es un fármaco antirresortivo que se inyecta en la paciente cada 6 meses, y una cantidad de 60 mg. Pero esto no es suficiente para algunos pacientes, o es demasiado para otros. Por eso se ha desarrollado un programa, que calcularía el tratamiento óptimo para cada tipo de persona. Se parte de un modelo de poblaciones celulares, que modela el mecanismo de remodelación ósea en un hueso enfermo y la adición del tratamiento. Con los resultados obtenidos de este modelo se han usado herramientas de uso cada vez más frecuente en la actualidad, como son las redes neuronales y los algoritmos genéticos.

Abstract

Currently, the increase in the elderly has led to an increase in women suffering from postmenopausal osteoporosis, a disease that has a great impact, increasing the number of fractures and healthcare costs. This disease can be treated with anabolic (e.g. teriparatide) or anticatabolic (e.g. bisphosphonates or Denosumab) drugs. This work focuses on the study of the latter.
Denosumab is an antiresorptive drug that is injected at a dose of 60 mg SC administered every 6 months into the patient. But this is not enough for some patients, or it is too much for others. That is why a program has been developed, which would calculate the optimal treatment for each type of person. It is based on a model of cell populations, which models the mechanism of bone remodeling in a diseased bone and the addition of treatment. With the results obtained from this model, tools have been used that are increasingly frequent today, such as neural networks and genetic algorithms.

Índice

1 Introducción

Actualmente, el incremento de las personas de tercera edad, ha llevado consigo un aumento de mujeres que padecen la osteoporosis postmenopáusica. Una enfermedad que afecta a los huesos y a la calidad de vida de estas personas.
En el siguiente trabajo, se presenta un programa que logra predecir el tratamiento óptimo para una persona que sufra osteoporosis postmenopáusica buscando un equilibrio entre el efecto antireabsortivo del Denosumab, pero sin que la falta de reabsorción produzca una acumulación excesiva de daño microestructural y de mineral. Dicha aplicación necesita de una serie de datos de entrada, que son característicos del paciente y con ellos, tras una serie de cálculos, se obtiene el tratamiento. La aplicación final que se obtiene de todos estos estudios será muy sencilla de utilizar y, con la introducción de unos parámetetros se obtiene de forma directa el tratamiento, sin la necesidad de conocer la programación interna y por lo tanto, no hay que cambiar los códigos para cada paciente. Esto se va a llevar a cabo con la ayuda de una red neuronal, que ayudará a precedir las salidas que queremos optimizar y con un algoritmo genético, que gracias a sus características, nos permitirán conocer los parámetros del tratamiento óptimo.

1.1 Justificación del trabajo

En la actualidad, lo normal es que una persona que se esté tratando con Denosumab lo reciba cada 6 meses y una dosis de 60 mg, aunque a veces, existe la posibilidad de hacer un descanso entre periodos de tratamiento, y sería objeto de otro estudio ver cómo afecta la duración de esas "vacaciones terapéuticas",al haberse demostrado que en algunas ocasiones se puede producir un efecto rebote. Sin embargo, esta cantidad puede no ser suficiente para una determinada paciente, o ser demasiado para otra. Por eso surge la necesidad de personalizar dicho tratamiento para cada persona.
Las dosis y frecuencias de los tratamientos y el tiempo de intercambio de un tratamiento a otro, son las variables a optimizar en el diseño de un tratamiento específico de un paciente.

1.2 Estructura del trabajo

Este trabajo de fin de grado está dividido en distintos capítulos: en primer lugar, en el capítulo 2 se expone brevemente la biología ósea, es decir, qué tipos de huesos hay, sus funciones y composición. También se tratará la remodelación ósea, un tema muy importante ya que influye enormemente en la enfermedad que vamos a estudiar, la osteoporosis postmenopáusica, que también se explica en este capítulo.
Más adelante, en el capítulo 3, se expondrá el modelo matemático usado para modelar un hueso, teniendo en cuenta la remodelación ósea, la enfermedad y el tratamiento.

Posteriormente, en los capítulos 4 y 5, se van a explicar las dos herramientas usadas para llevar a cabo el proceso de optimización, las redes neuronales y los algoritmos genéticos, respectivamente. En el capítulo 6, se muestran los distintos procedimientos que se han utilizado y las justificaciones de los mismos para llegar al objetivo final, mientras que en el capítulo 7, se muestran las soluciones obtenidas en cada una de las fases del trabajo.
Los capítulos 8 y 9 son manuales para el programador y para el usuario, respectivamente. En el primero se explican los programas que han sido utilizados para llevar a cabo la aplicación, mientras que en el segundo, se muestra al usuario final de la aplicación, cómo se usaría el programa.
Por último, en el capítulo 10, se manifiestan las conclusiones obtenidas durante la realización de este trabajo y también se exponen los posibles futuros trabajos que partirían de este.

2 Biología ósea. Osteoporosis

El ser humano está compuesto de una serie de estructuras rígidas que lo sostienen, ayudando además al movimiento de los músculos: los huesos, que, además, tienen la capacidad de renovarse o autorrepararse continuamente mediante el mecanismo de Remodelación Ósea. Cuando hay un desequilibrio en este proceso, se produce una de las enfermedades más comunes en la actualidad: la osteoporosis, que también se desarrollará en este capítulo.

2.1 Hueso: tipos, funciones y composición

El esqueleto humano está formado por 206 huesos, que tienen numerosas funciones como son la de dar soporte a los tejidos blandos del cuerpo y fijarlos a los músculos. Éstos son insertados proximalmente, produciendo una pérdida mecánica al necesitar una mayor fuerza para levantar objetos, pero agilizan el movimiento dando una función locomotora al hueso. También tienen funciones de protección de órganos internos, así como de reserva de minerales como el calcio, el sodio y el fósforo que, gracias al sistema vascular, son transportados por todo el cuerpo y cuando disminuye el nivel de calcio en sangre, se toman las reservas de calcio de los huesos para suplir la deficiencia, además de que son fuentes de células sanguíneas (eritrocitos, leucocitos, plaquetas) a partir de células madre.
Dependiendo de su morfología se pueden clasificar en (ver figura 2.2):

- Largos: son aquellos que se caracterizan porque una dimensión predomina sobre las otras dos. Su principal función es de soporte del cuerpo ya que pueden aguantar cargas elevadas y también facilita el movimiento, un ejemplo de este tipo de hueso es el húmero. Los huesos largos están diferenciados en distintas regiones anatómicas: la diáfisis, que es la parte central, formada por hueso cortical y tiene una estructura tubular hueca en la que se encuentra la médula ósea, por otro lado está la epífisis que se corresponde con los estremos, está formada por hueso trabecular y cubierto por cortical, sirviendo de unión con otros huesos y, por último, la metáfisis, que es la zona intermedia entre estos dos (Figura 2.1).

- Cortos: Las tres dimensiones características del hueso son similares, por ejemplo, los huesos del carpo. Su función es permitir algunos movimientos y dar estabilidad.

- Planos: son los huesos que tienen dos dimensiones parecidas y la tercera es significativamente más pequeña, es decir, tienen un aspecto aplanado, la función principal es de protección de órganos internos, por ejemplo, los huesos del cráneo protegen al cerebro, o las costillas junto con el esternón (en su conjunto son consideradas un hueso plano) protegen a los pulmones, corazón, etc.

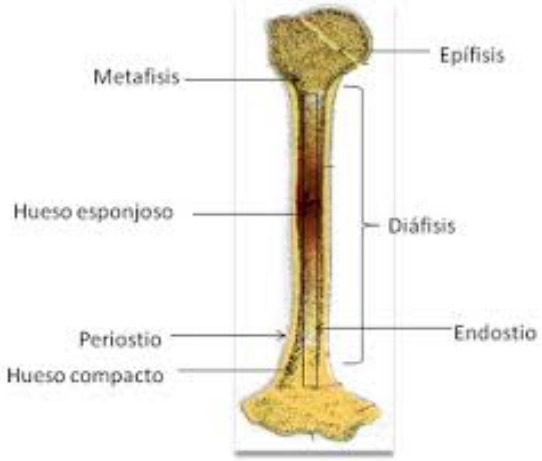
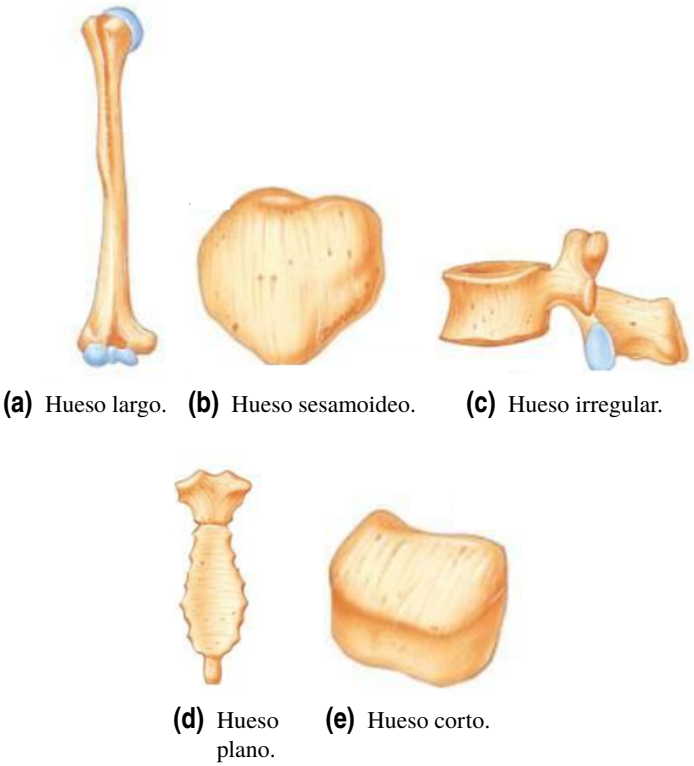


Figura 2.1 Partes del hueso largo [1].

- Irregulares: son huesos que no se pueden encontrar en las clasificaciones anteriores debido a la complejidad de su forma, ayudan a la protección de órganos internos, como es el caso de las vértebras con la médula espinal.
- Sesamoideos: son aquellos huesos que están incluidos en los tendones, son pequeños y redondos como la rótula, teniendo la función de proteger a los tendones frente al estrés y al deterioro.



(a) Hueso largo. **(b)** Hueso sesamoideo. **(c)** Hueso irregular.

(d) Hueso plano. **(e)** Hueso corto.

Figura 2.2 Tipos de huesos según su forma [2].

Si clasificamos a los huesos según su tejido óseo, se tiene:

- Cortical: también es llamado compacto debido a su alta densidad y forma aproximadamente el 80 % de la masa del esqueleto. Las lamelas que componen el hueso son caracterizadas

por formar círculos concéntricos, dando lugar a las osteonas, que se enrollan alrededor de unos canales conocidos como canales de Havers. Éstos contienen vasos sanguíneos y nervios, aunque también se cree que hay vasos linfáticos, así resisten de forma eficiente a la fuerza de compresión. Estos huesos se encuentran en la parte central de los huesos largos, la diáfisis, y en las capas más externas del hueso rodeando al trabecular [3].

- Trabecular: está caracterizado por tener un tejido óseo esponjoso, al tener una porosidad entre un 50 % y un 90 %. El hueso está formado por una serie de tubos unidos y entrelazados llamados trabéculas, en cuyos espacios se encuentra la médula, colocadas de forma irregular para que soporte diferentes cargas, aunque la resistencia del trabecular es menor que en el cortical en cualquiera de las direcciones y sentidos. Además, en este sistema hay zonas de vacío, como ocurre en el cuello del fémur (Figura 2.4), donde se forma el triángulo de Ward, que es un añadido a la fragilidad del hueso [3].

Figura 2.3 Triángulo de Ward [3].

En cuanto a la composición del hueso, éste está formado por matriz ósea y por poros. La matriz ósea, a su vez está compuesta por una fase orgánica, que ocupa el 30 % de la matriz y está formada principalmente por colágeno I, aunque también se puede encontrar de tipo III y V. Esta fase, está formada por fibras colágenas que están compuestas por cadenas de triple hélice enrolladas a izquierdas dando lugar a los polipéptidos. A continuación, estas cadenas se unen de tres en tres a derechas formando las móleculas de colágeno (tropocolágeno) y por último, al unirse las moléculas, dan lugar a las fibras. Al estar enrrolladas en distintos sentidos, hace que aumente la resistencia a tracción y que la fuerza de compresión no las deshaga. Además de colágeno, la fase orgánica está formada por proteínas no colágenas como la osteonectina y la osteopontina, que sirven de marcadores bioquímicos de la actividad del hueso y de la homeostasis del calcio. Por otro lado, está la fase inorgánica o mineral, que es la más presente en la matriz ósea, ya que se encuentra en torno al 70 %. Es la responsable de que los huesos tengan gran resistencia a la compresión y está formada por cristales de hidroxiapatita, que se encuentra insertados entre las fibras de colágeno. Por último, la matriz está compuesta también por agua, aunque es la fase más pequeña [3].

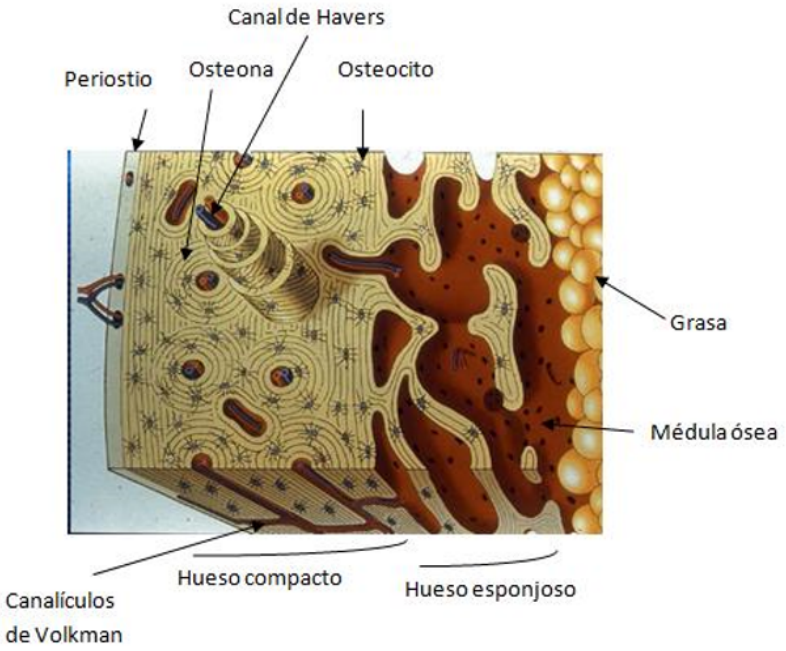


Figura 2.4 Partes del hueso: cortical y trabecular [3].

2.2 Remodelación ósea

El hueso es un órgano que está contínuamente regenerándose, gracias al mecanismo de remodelación ósea, por el que el tejido óseo es reparado debido a la renovación periódica o por alguna fisura. Dependiendo de cómo de grande sea el daño, se diferenciará en la remodelación ósea interna, ROI, que trabaja los cambios en la microestructura interna del hueso, teniendo en cuenta la porosidad y la orientación de la microestructura. Por otro lado, se encuentra la remodelación ósea externa, ROE, que trata los cambios externos del hueso, es decir, tanto de la forma como de las dimensiones.

La ROI es la que se va a analizar a continuación, y es llevada a cabo mediante la acción conjunta de los osteoclastos (OC) y los osteoblastos (OB), que actúan de forma secuencial en una asociación temporal de células llamada BMU (unidades multicelulares básicas) [4], que se describe en la figura 2.5, y junto con los osteocitos se lleva a cabo la remodelación.

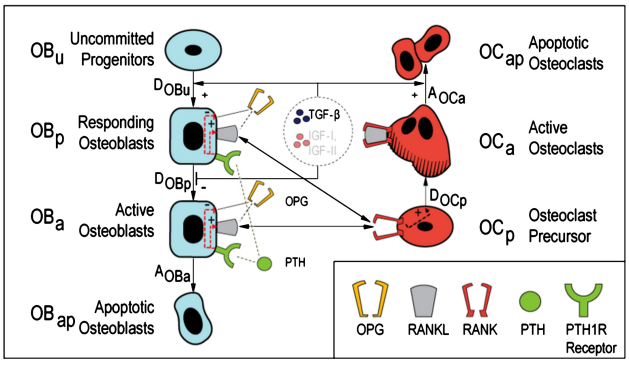


Figura 2.5 Esquema del modelo de población de células ósea donde se puede ver las diferentes fases de la activación de la remodelación ósea. En ella se aprecian las células que intervienen en las distintas etapas de su vida junto con los elementos bioquímicos [5].

2.2.1 Células óseas y funciones

Hay cuatro tipos de células óseas que tienen una función bien definida: osteoclastos, osteoblastos, osteocitos y células borde. Las primeras proceden de las células madre hematopoyéticas, más en concreto de las monocito-macrófago [6], mientras que las otras tres pertenecen a otro linaje, y es el de las células madre mesenquimales [5].

- Osteoclastos: son células multinucleadas ya que proceden de la unión de osteoclastos precursores, y su principal función es la de la reabsorción ósea (Figura 2.6). La reabsorción se lleva a cabo mediante la segregación de ácidos que disuelven la parte mineral de la matriz ósea, mientras que la parte orgánica se deshace mediante la actuación de una enzima. La actuación de los osteoclastos termina cuando han reabsorvido la zona dañada, y entonces es cuando sufren apoptosis o suicidio programado al ser fagocitados por macrófagos.

  Cuando los osteoclastos disuelven parte de la matriz ósea, se devuelve el calcio y fósforo que estaban almacenados, además de que desaparece la zona dañada [6].

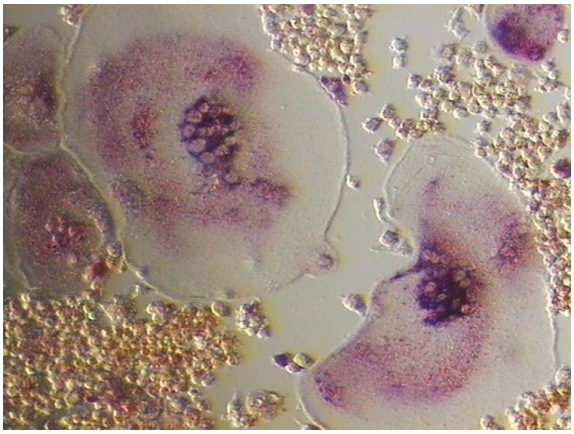


Figura 2.6 Imagen tomada con un microscopio óptico en la que se observan osteoclastos positivos a la fosfatasa ácida tartrato resistente [7].

- Osteoblastos: son pequeñas células mononucleadas y de forma cuboidal que se encargan de la formación del nuevo tejido óseo (Figura 2.7). Solo segregan osteoide, es decir, la fase orgánica y agua, que más tarde se irá mineralizando desplazando al agua. Los osteoblastos se diferencian de las células mesenquimales a los 2-3 días gracias a un estímulo mecánico y su actuación es significativamente más lenta que la de los osteoclastos al ser del orden de decenas de micras al día estos últimos.

  Los osteoblastos empiezan a actuar al cabo de 8 días desde que estuvieron los osteoclastos ya que, como éstos segregaron ácidos, podría dañarlos, así que se espera el tiempo suficiente para que el medio esté en condiciones de depositar el osteoide.

  Cuando han completado su función, una parte de ellos sufre apoptosis, sin embargo, la mayoría se diferencia en osteocitos (los que quedan atrapados en la matriz ósea) o en células borde [5].

- Osteocitos: son las células más numerosas que forman el tejido óseo, se encuentran atrapadas en la matriz ósea ya que cuando acabaron su misión como osteoblastos, pusieron otro tejido encima, impidiéndoles salir. Cuando esto ocurre, la forma del osteoblasto precursor cambia volviéndose más alargada, y se coloca en unos lugares llamados lagunas conectándose con otros osteocitos y con las células borde mediante prolongaciones que se encuentran en los

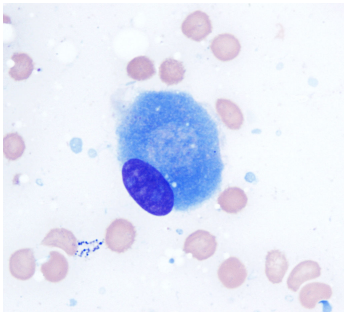


Figura 2.7 Imagen tomada con un microscopio óptico de un osteoblasto teñido con Giemsa [8].

canalículos (Figura 2.8). Estas células se caracterizan por "sentir" las cargas, ya que su función es el mantenimiento del hueso y asegurar que la carga aplicada en el hueso es la adecuada, es el fenómeno denominado mecanotransducción [9]. Entonces, si los osteocitos dejan de sentir la carga, o notan que la carga es demasiado elevada (lo notan al dejar de sentir el flujo de fluidos a través del sistema lacuno-canalicular), segregan una sustancia bioquímica que llega a las células borde y éstas ponen en marcha la remodelación ósea [10].

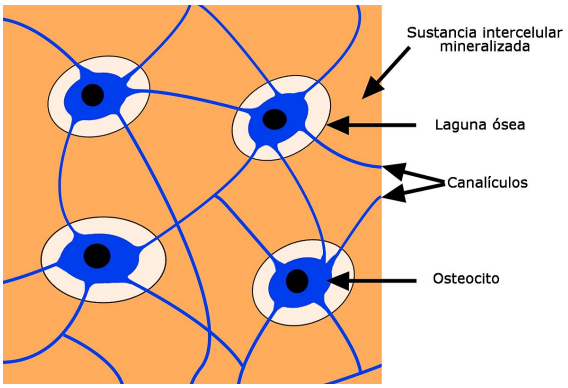


Figura 2.8 Esquema histológico del tejido óseo en el que son visibles los osteocitos [11].

- Células borde: Son la diferenciación de los últimos osteoblastos de la remodelación ósea, que en lugar de sufir apoptosis se diferencian en células borde. Se denominan así debido a que se depositan en la superficie [9], pero se transforman alargándose y cubriendo la matriz ósea, de forma que la protege de agentes químicos externos. Están en contacto con los osteocitos, mediante las prolongaciones de éstos y, cuando la remodelación se activa, se apartan para que actúen los osteoclastos, formando un conducto con los vasos cercanos.

2.2.2 Fases de la remodelación ósea

En el mecanismo de remodelación ósea hay una serie de fases que la componen: la activación, la reabsorción y la formación de tejido óseo seguido de una fase de reposo (Figura 2.9). Cualquier desequilibrio en estas etapas da lugar a las enfermedades óseas como el cáncer, enfermedades de Paget, osteopetrosis u osteoporosis, que se tratará más adelante.
En la remodelación influyen una serie de factores bioquímicos como son el RANKL, RANK, OPG, PTH y TGF-β.
La remodelación ósea da comienzo cuando hay un desequilibrio en las cargas aplicadas, entonces los osteocitos mediante un estímulo hormonal u otros factores, avisan a las células borde. Éstas segregan RANKL (aunque se encuentra en mayor medida en los osteoblastos precursores), un ligando del receptor activador de NF-κB [6] que activa el receptor RANK, presente en los preosteoclastos, y

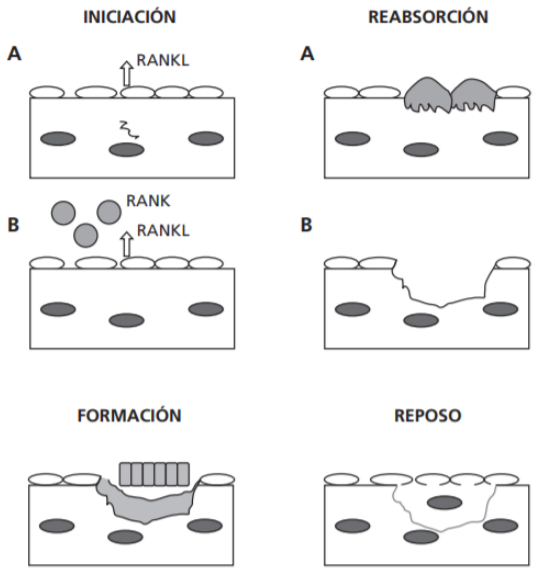


Figura 2.9 Esquema de las distintas fases de la remodelación ósea [4].

su interacción con el RANKL es la que permite la diferenciación de los osteoclastos precursores a osteoclastos activos capaces de reabsorver la superficie ósea [4]. Por otro lado, los osteoblastos sintetizan OPG, una proteína llamada osteoprotegrina, que también es producida por células estromales. Ésta se encarga de limitar la formación de osteoclastos activos al unirse al RANKL y cuando este sistema falla, es cuando se produce la osteoporosis [6], al reabsorverse más hueso del que debería.

Por otra parte, se encuentran unas hormonas sistémicas: el PTH y, junto con la vitamina D_3, regulan la actuación de los osteoclastos. Esta regulación se hace de forma indirecta al encontrarse los receptores de estas hormonas únicamente en los osteoblastos, de forma que la homeostasis mineral es regulada a la vez que se controlan a los osteoclastos a través de los osteoblastos. Otra hormona sistémica el la caltinonina, que actúa como antagonista del PTH y de la vitamina D_3 [5].

El factor de crecimiento TGF-β fomenta la diferenciación de osteoblastos no comprometidos, pero impide que éstos se diferencien en osteoblastos precursores, por lo que una inhibición del TGF-β aumentaría la cantidad de osteoclastos precursores. También se ha descubierto que este factor de crecimiento interviene en la apoptosis de los osteoclastos [5].

Otros factores que influyen en la remodelación ósea son los de la tabla 2.1:

Tabla 2.1 Factores mecánicos y no mecánicos que influyen en el proceso de la remodelación ósea [12].

Mecánicos	No mecánicos
• Fuerza de la gravedad	• Hormonas
• Peso del individuo	• Factores locales (autocrinos/paracrinos)
• Actividad física-sedentarismo	• Edad/sexo/genética
• Contractilidad muscular	• Dieta (calcio, vitaminas, minerales,...)
• Esfuerzo	• Ocupación laboral/ergonomía
• Ingravidez-reposo prolongado	• Algunas enfermedades (p. ej. artritis reumatoide)

2.3 Osteoporosis

La osteoporosis es una enfermedad muy común hoy en día y afecta a un gran número de personas. Según la OMS, en españa afecta a 3,5 millones de personas, siendo más común en mujeres [13] y su nombre significa "hueso poroso". Está caracterizada por causar pérdida ósea tanto en masa, como en resistencia y microestructura aumentando la fragilidad del hueso. Es muy frecuente que las personas que padecen esta enfermedad, al menor esfuerzo se rompan un hueso, ocurriendo en mayor frecuencia en la muñeca, la cadera y la columna (Figura 2.10), aunque también pueden aparecer en las costillas, en el trocánter o el húmero [14].

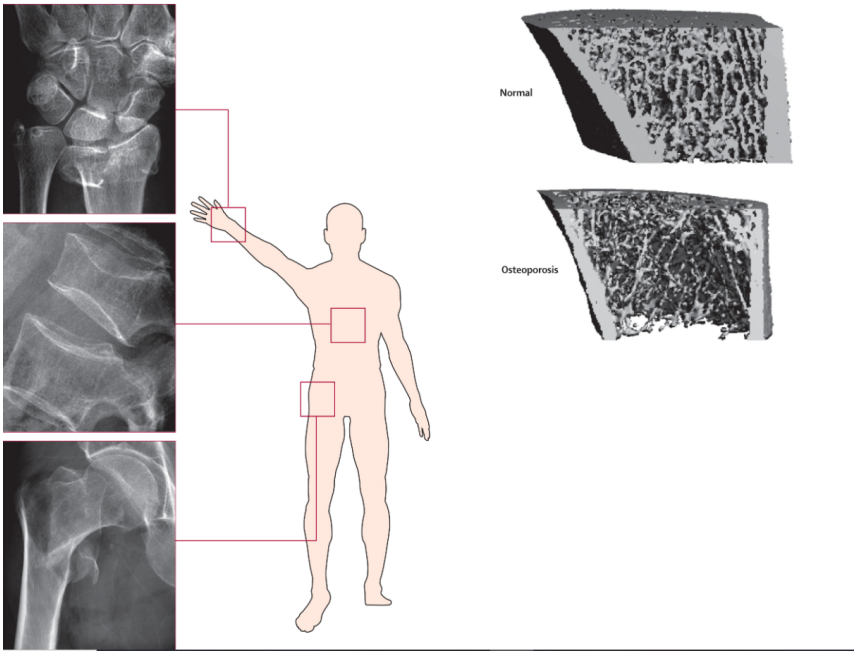


Figura 2.10 De izquierda a derecha: las fracturas por fragilidad típicamente involucran muñeca, vértebras y cadera, comparación de un hueso sano con un hueso con osteoporosis [14].

La osteoporosis se puede clasificar en dos grupos, en primer lugar, está la primaria, que se divide a su vez en otros tres subgrupos [15]:

- Idiopática juvenil o de adulto joven: esta no es muy común y suele aparecer a una edad temprana o en los embarazos. Remite de forma espontánea en los niños, mientras que la evolución en los adultos es más variable.
- Osteoporosis postmenopáusica. Tipo I: en este tipo de osteoporosis afecta a las mujeres que han tenido la menopausia y están dentro del rango de edad 51-71 años. Aquí los osteoclastos intervienen demasiado, reabsorviendo mucho más hueso del que se forma y se ve una pérdida del PTH que compensa el efecto de la remodelación ósea.
- Osteoporosis postmenopáusica. Tipo II: ocurre en mujeres y hombres de más de 70 años debido a la poca actuación de los osteoblastos.

Por otro lado se encuentra la osteoporosis secundaria, que es debida a efectos secundarios de algunos fármacos u otras enfermedades.
Los factores más comunes para la aparición de la osteoporosis son el sexo, ya que predomina significativamente en las mujeres frente a los hombres. Según la OMS, en la Unión Europea afecta a 22 millones de mujeres frente a 5,5 millones de hombres [13]. Otro factor es la raza, ya que afecta más a la blanca que a la negra. La menopausia, ya que al disminuir las hormonas estrógeno, no se frena la acción de los osteoclastos como se puede ver en la figura 2.11 al no disminuir la cantidad de RANKL ni aumentar OPG. La edad es otro factor de riesgo, la constitución corporal, si es ligera aumenta la posibilidad de contraerla. La falta de actividad física, el consumo de alcohol y tabaco, así como los hábitos alimenticios (si se ha consumido el suficiente calcio durante la infancia y adolescencia) etc [16].

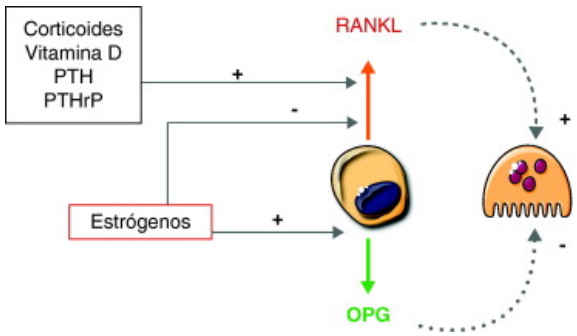


Figura 2.11 Efectos de los factores hormonales sobre RANKL y osteoprotegerina [6].

En cuanto a la osteoporosis postmenopáusica, que es la que va a tratarse en este trabajo, es frecuentemente tratada con Denosumab, un fármaco antirresortivo que inhibe la formación y la actividad de los osteoclastos al tener gran afinidad por el RANKL, un ligando del activador del receptor del factor nuclear [17], que es esencial para el desarrollo de osteoclastos activos, aumentando de forma directa la densidad ósea y disminuyendo las fracturas. Sin embargo, este tratamiento tiene el inconveniente de que al haber mucha matriz ósea antigua, ésta está más mineralizada y por ende es más frágil [18]. El suministro del tratamiento está en 60 mg cada 6 meses, pero en este trabajo se va a buscar una forma de optimizar el tratamiento personalizado a cada persona.

3 Modelo matemático

Para llegar a conseguir el objetivo final de este trabajo, es necesario conocer el modelo matemático que permite simular el mecanismo de remodelación ósea en un paciente enfermo con osteoporosis posmenopáusica, y se ha basado principalmente en los artículos de (Pivonka et al., 2013) [19], (Pivonka et al., 2008) [5], (Marínez et al., 2019) [18]. Además, también se incluye el efecto del Denosumab (Shneiner et al., 2014) [17]. En este apartado se van a describir primeramente los parámteros usados en el modelo, para posteriormente entender mejor las ecuaciones que lo rigen.

3.1 Parámetros que describen el modelo

En primer lugar, se van a exponer las constantes del modelo para su mayor comprensión a la hora de ver las ecuaciones. Se han puesto primero las cantidades dinámicas que gobiernan el problema [19] (Tabla 3.1) y a continuación, las constantes del modelo junto con una breve descripción (tabla 3.3).

Tabla 3.1 Cantidades dinámicas en las ecuaciones de gobierno.

Símbolo	Descripción
OC_a	Densidad de osteoclastos activos, en pM
OB_p	Densidad de pre-osteoblastos, en pM
OB_a	Densidad de osteoblastos activos, en pM
$TGF\beta$	Concentración de $TGF\beta$, en pM
RANK	Concentración de RANK, en pM
RANKL	Concentración de RANKL, en pM
OPG	Concentración de OPG, en pM
PTH	Concentración de PTH, en pM
f_{vas}	Porosidad vascular
S_V	Superficie específica
ψ_{bm}	Densidad de energía de deformación, en MPa

Tabla 3.2 Constantes del modelo, sus valores, unidades y descripción.

Parámetro	Valor	Unidad	Descripción
D_{OB_u}	6,30E-04	día^{-1}	Tasa de diferenciación de OB_u a OB_p
D_{OB_p}	1,66E-01	día^{-1}	Tasa de diferenciación de OB_p a OB_a
A_{OB_a}	2,11E-01	día^{-1}	Tasa de apoptosis de OB_a

P_{OB_p}	2,11E-02	día^{-1}	Ratio de proliferación de OB_p
D_{OC_u}	4,20E-01	día^{-1}	Tasa de diferenciación de OC_u a OC_p
D_{OC_p}	2,10E+00	día^{-1}	Tasa de diferenciación de OC_p a Oc_a
A_{OC_a}	5,65E+00	día^{-1}	Tasa de apoptosis de OC_a
$K_{act}^{TGF\beta}$	5,63E-04	pM	Coeficiente de activación de TGF-relacionado con la diferenciación de OB_u a OB_p
$K_{rep}^{TGF\beta}$	1,75E-04	pM	Coeficiente de represión de TGF-relacionado con la diferenciación de OB_p a OB_a
$K_{[RANKL-PTH],act}$	1,50E+02	pM	Coeficiente de activación para la producción de RANKL debida a la unión de PTH
$K_{[RANKL-PTH],rep}$	2,23E-01	pM	Coeficiente de represión para la producción de RANKL debida a la unión de PTH
$K_{[RANKL-RANK],act}$	5,68E+00	pM	Coeficiente de activación relacionado con la unión de RANKL y RANK
$N_{OC_p}^{RANK}$	1,00E+04	pM RANK / pM cells	Coeficiente de represión relacionado con la unión de RANKL y RANK
K_{act}^{MCSF}	1,00E-03	pM	Coeficiente de activación relacionado con la unión de MCSF y OC_u
$P_{RANKL-d}$	0,00E+00	pM/día	Inyección externa de RANKL
$P_{RANKL-dlc}$	2,00E+05	pM/día	Modificación de la inyección externa de RANKL para otros casos de carga
P_{RANKL}^{PMO}	Variable según paciente	pM/día	Tasa de producción de RANKL relacionada con PMO
ξ	6,50E+01	-	Parámetro para ϕ_{PMO}^{RANKL} en PMO
ϕ_{PMO}^{RANKL}	1,00E+00	-	Factor de atenuación de RANKL con PMO
τ_{PMO}^{RANKL}	1,00E+01	días	Constante de tiempo en el factor de atenuación de RANKL con PMO
P_{OPG-d}	0,00E+00	pM/día	Inyección externa de OPG
P_{PTH-d}	0,00E+00	pM/día	Inyección externa de PTH
$P_{TGF\beta-d}$	0,00E+00	pM/día	Inyección externa de $TFG\beta$
pd_{OB_a}	0,00E+00	pM cell/día	Producción adicional de OB_a
pd_{OB_p}	0,00E+00	pM cell/día	Producción adicional de OB_p
pd_{OC_Y}	0,00E+00	pM cell/día	Producción adicional de OC_Y
pd_{OC_p}	0,00E+00	pM cell/día	Producción adicional de OC_p
pd_{OC_a}	0,00E+00	pM cell/día	Producción adicional de OC_a
$\widetilde{D}_{PTH}$	8,60E+01	día^{-1}	Tasa de degradación de PTH
β_{PTH}	2,50E+02	pM PTH/día	Tasa de síntesis de PTH sistémico
β_{OPG}	1,63E+08	pM OPG/pM cell/día	Tasa mínima de producción de OPG por célula
$\widetilde{D}_{OPG}$	3,50E-01	día^{-1}	Tasa de degradación de OPG
OPG_{max}	2,00E+08	pM OPG	Máxima concentración posible de OPG
$\beta 1_{OB_p}^{OPG}$	0,00E+00	-	Factor para la expresión de OPG sobre OB_p
$\beta 2_{OB_a}^{OPG}$	1,00E+00	-	Factor para la expresión de OPG sobre OB_a
β_{RANKL}	1.684.200	pM RANKL/día	Tasa de producción de RANKL por célula
R_{RANKL}	2,70E+06	pM RANKL/pM cell	Máxima concentración de RANKL sobre cada superficie celular

$\widetilde{D}_{RANKL}$	1,10E+01	día^{-1}	Tasa de degradación de RANKL
$K_{A1[RANKL-OPG]}$	1,00E-03	(pM OPG)$^{-1}$	Constante de asociación RANKL - OPG
$K_{A2[RANKL-RANK]}$	3,41E-02	(pM RANKL)$^{-1}$	Constante de asociación RANKL - RANK
$R1_{OB_p}^{RANKL}$	1,00E+00	-	Factor para la expresión de RANKL sobre OB_p
$R2_{OB_a}^{RANKL}$	0,00E+00	-	Factor para la expresión de RANKL sobre OB_a
$\widetilde{D}_{TGF\beta}$	2,00E+00	día^{-1}	Tasa de degradación de $TGF\beta$
t_R	3000	días	Tiempo de residencia del mineral
Dt	1,20E+01	días	Tiempo de retraso de mineralización
MINP	1,00E+01	días	Longitud de la fase primaria
XKAPPA	4,50E-03	día^{-1}	Tasa de mineralización
v_{prim}	1,21E-01	$\frac{V_{mineral}}{VRVE}$	Volumen específico de mineral al final de la fase primaria
v_{max}	4,20E-01	Vmineral / VRVE	Volumen específico de mineral correspondiente al máximo contenido de Ca
k_{OB_u}	0,00E+00	-	Exponente de la regulación geométrica. Diferenciación de OB_u a OB_p
k_{OB_p}	0,00E+00	-	Exponente de la regulación geométrica. Diferenciación de OB_p a OB_a
k_{OC_u}	0,00E+00	-	Exponente de la regulación geométrica. Diferenciación de OC_u a OC_p
k_{OC_p}	0,00E+00	-	Exponente de la regulación geométrica. Diferenciación de OC_p a OC_a
$K_{A3,Denosumab}$	0,067	(pM RANKL)$^{-1}$	Asociación vinculante constante RANKL-Denosumab Nominal
$C_{Denosumab}$	0,00E+00	pM Denos / pM cells	Concentración de Denosumab
N	1,00E+04	ciclos	Número de ciclos de deformación diarios
nivel	1,00E+00	-	Tensión constante
σ	5,00E-01	Mpa	Tensión aplicada (compresión)
$\Delta\sigma$	Variable según paciente	Mpa	Incremento de tensión aplicada
ε	2,80E+03	microdef	Deformación
$\Delta\varepsilon$	0,00E+00	microdef	Incremento de deformación
ϕ_{PMO}^{mech}	1,00E+00	-	Capacidad de mecanorespuesta de los osteocitos
τ_{PMO}^{mech}	9,00E+02	días	Constante de tiempo en la ecuación de la mecanorespuesta
MESr	8,00E-04	microdef	Máxima deformación provocada por la producción de RANKL
MES1	1,00E-09	microdef	Límite de modelo
MES2	8,00E-04	microdef	Deformación correspondiente al cambio de pendiente en la tasa de proliferación
MES3	1,80E-03	microdef	Deformación correspondiente a la tasa mínima de proliferación
K_{dam}	1,00E+04	pM/día	Influencia del factor de daño en el RANKL
N_f	1,00E+07	ciclos	Número de ciclos de fatiga

BW	6,00E+01	kg	Masa de la paciente
Δt	2,50E-01	días	Incremento de tiempo
k_{res}	Variable según paciente	%	Tasa relativa de reabsorción ósea normalizada con respecto a la reabsorción ósea normal
K_{form}	4,00E+01	%	Tasa relativa de formación ósea normalizada con respecto a la formación ósea normal
MCSF	1,00E-03	pM	Concentración de factor estimulante de la colonia de macrófagos
$n_{TGF\beta}^{bone}$	1,00E-02	pM	Densidad de $TGF\beta$ almacenado en la matriz ósea
$N_{OB_p}^{RANKL}$	2,70E+06	-	Número máximo de RANKL por OB_p
$N_{OC_p}^{RANK}$	1,00E+04	-	Número máximo de RANK por OC_p

3.2 Porosidad en la matriz ósea

Una vez que se tienen los parámetros principales del modelo, se ve que para representar la microestructura del hueso se usan unos parámetros geométricos donde uno de los más importantes es la porosidad vascular.

$$f_{vas} = \frac{V_{vas}}{V_T} \tag{3.1}$$

Donde V_{vas} es el volumen de poros vasculares.
V_T es el volumen de tejido.

$$f_{bm} = \frac{V_{bm}}{V_T} \tag{3.2}$$

Donde V_{bm} es el volumen de matriz ósea.
De estas ecuaciones se deduce que:

$$f_{vas} + f_{bm} = 1 \tag{3.3}$$

3.3 Funciones de Hill

La función de Hill es una función comúnmente empleada que describe muchas funciones reales de genes y de entrada molecular [5]. A la luz de si es un proceso de activación o uno de represión se tiene:

$$\beta\pi_{act} = \frac{\beta(X^*)^n}{K1 + (X^*)^n} \tag{3.4}$$

$$\beta\pi_{rep} = \frac{\beta}{1 + (\frac{X^*}{K2})^n} \tag{3.5}$$

Donde K1 y K2 [mol / l] son coeficientes de activación y represión, β es el nivel de expresión máximo del promotor, y n es el coeficiente de Hill. Se toma que n = 1.

Para las expresiones de las funciones de Hill hay que tener en cuenta lo que significa cada superíndice o subíndice. De forma general se escribe:

$$\pi^{molecula}_{act/rep,cell} \tag{3.6}$$

La "molécula" está referenciada al ligando involucrado, la "célula" es el tipo de célula a la que se une el ligando específico.
Los factores bioquímicos que influyen en la remodelación ósea se pueden modelar matemáticamente mediante las fuciones de Hill:
$TGF\beta$:

$$\pi^{TGF\beta}_{act,OB_u/OC_a} = \frac{TGF\beta}{TGF\beta + K^{TGF\beta}_{OB_u/OC_a,act}} \tag{3.7}$$

$$\pi^{TGF\beta}_{rep,OB_p} = \frac{TGF\beta}{1 + \frac{TGF\beta}{K^{TGF\beta}_{OB_p,rep}}} \tag{3.8}$$

RANKL:

$$\pi^{RANKL}_{act,OC_u/OC_p} = \frac{[RANKL - RANK]}{[RANKL - RANK] + K^{RANKL}_{OC_u/OC_p,act}} \tag{3.9}$$

$$\pi^{RANKL}_{act,OC_u} = \pi^{RANKL}_{act,OC_p} \tag{3.10}$$

Donde el coeficiente de activación K es igual a $K_{A2[RANKL-RANK]}$.
MCSF:

$$\pi^{MCSF}_{act,OC_p} = \frac{MCSF}{MCSF + K^{MCSF}_{OC_p}} \tag{3.11}$$

Donde $K^{MCSF}_{OC_p} = K^{MCSF}_{act}$.
PTH:

$$\pi^{PTH}_{act,OB_a/OB_p} = \frac{PTH}{PTH + K^{PTH}_{OB_a/OB_p,act}} \tag{3.12}$$

$$\pi^{PTH}_{act,OB_a} = \pi^{PTH}_{act,OB_p} \tag{3.13}$$

$$\pi^{PTH}_{rep,OB_a/OB_p} = \frac{PTH}{1 + \frac{PTH}{K^{PTH}_{OB_a/OB_p,act}}} \tag{3.14}$$

$$\pi^{PTH}_{rep,OB_a} = \pi^{PTH}_{rep,OB_p} \tag{3.15}$$

3.4 Regulación mecánica

En la regulación mecánica es muy importante la densidad de energía de deformación ψ_{bm}, al determinar la adaptación ósea a las distintas cargas mecánicas. También es una medida del estímulo mecánico detectado por las células óseas, que pondrán en marcha la adptación ósea [19].

La densidad de deformación ósea es una función que depende del tensor de estrés macroscópico, $\sum$, del tensor de rigidez de la matriz ósea, c_{bm} y de la porosidad vascular f_{vas}.

$$\psi_{bm} = \psi_{bm}(\sum, c_{bm}, f_{vas}) \tag{3.16}$$

La tasa de producción de RANKL en pre-osteoblastos viene dada por:

$$P^{mech}_{RANKL} = \begin{cases} P^{mech}_{RANKL,max}\left(1 - \frac{\psi_{bm}}{\psi_r}\right) & \psi_{bm} < \psi_r \\ P^{mech}_{RANKL,max} = 0 & \psi_{bm} \geq \psi_r \end{cases} \tag{3.17}$$

Donde ψ_r es el SED por debajo del cual, en el desuso se incrementa la prodccuión de RANKL y $P^{mech}_{RANKL,max}$ es la máxima producción de RANKL debido al desuso.
Se supone que la sobrecarga promueve la formación de hueso por la proliferación de precursores de osteoblastos a través de la función activadora $\Pi^{\psi_{bm}}_{act,OB_p}$, que se da por la función lineal por partes de SED (ver figura 3.1).

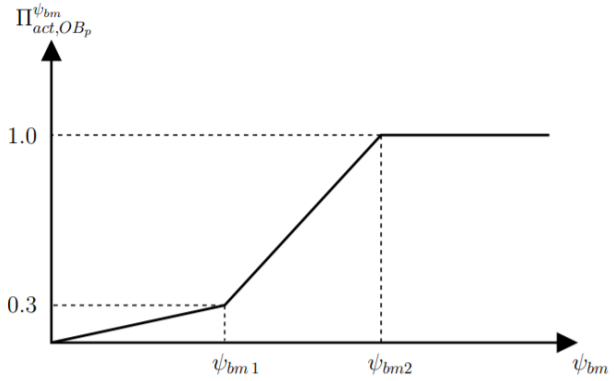


Figura 3.1 Curva de proliferación, ajustada para lograr la tería del mecanostato y el principio de alojamiento celular. [20].

3.5 Inclusión del daño

En este modelo se incluye el efecto del daño. Esta nueva variable está relacionada con la densidad de microgrietas que corresponde a la variable de daño continuo generalmente utilizada en la teoría isotrópica de la Mecánica de Daño Continuo [21]. La variable daño, está dentro del intervalo [0,1], donde 0 se corresponde con un hueso sin daño y 1 se corresponde con un estado de fractura o fallo del hueso.

$$d = 1 - \frac{E}{E_0} \tag{3.18}$$

Donde E y E_0 son el módulo de Young del hueso dañado y sano respectivamente.
La vida a fatiga viene dada por la expresión:

$$N_f = \frac{K_i}{\varepsilon^{\delta_i}} \qquad i = c(\text{compresión}), t(\text{tensión}) \tag{3.19}$$

De los test de fatiga experimentales de Patin et al. [22] obtuvieron que $K_c = 1.479 \cdot 10^{-21}$ y $\delta_c = 10.3$ en el caso de compresión, mientras que para la tensión obtuvieron $K_t = 3.630 \cdot 10^{-32}$ y $\delta_t = 14.1$.

$$d_c = -\frac{1}{\gamma_c}\left[\ln(1 - C_c\,\varepsilon^{\delta_c} N)\right] \tag{3.20a}$$

$$d_t = 1 - \left[\frac{1}{C_{t2}}\ln(e^{C_{t2}} - C_{t1}\,\varepsilon^{\delta_t} N)\right]^{\frac{1}{\gamma_t}} \tag{3.20b}$$

Donde los parámteros son:

$$\begin{aligned} &\gamma_c = -5.238(\varepsilon - 6100)10^{-3} + 7 \qquad C_c = \frac{1 - e^{-\gamma_c}}{K_c} \qquad \text{compresión} \\ &\gamma_t = -0.018(\varepsilon - 4100) + 12 \qquad C_{t1} = \frac{e^{C_{t2}} - 1}{K_t} \qquad C_{t2} = -20 \quad \text{tensión} \end{aligned} \tag{3.21}$$

En el modelo de daños propuesto por Martínez-Reina et al. [23] se supone que las grietas crecen normales a la dirección de deformación máxima y solo bajo deformaciones por tracción. Esto permite aplicar el modelo a un estado de deformación general, reemplazando ε con la deformación principal máxima, ε_{max}.
Lan ecuaciones se trataron como si el proceso se hiciese a tensión constante, sin embargo, se pueden aplicar a un historial de carga general utilizando el procedimiento descrito en [23] y explicado a continuación:

$$d = 1 - \left[\frac{1}{C_{t2}}\ln(e^{C_{t2}} - C_{t1}\,\varepsilon_{max}^{\delta_t}\tilde{N})\right]^{\frac{1}{\gamma_t}} \qquad \Longrightarrow \qquad \tilde{N} \tag{3.22}$$

El incremento del daño Δd_A se habría alcanzado con los N ciclos adicionales aplicados en el paso actual y puede evaluarse a partir de:

$$d + \Delta d_A = 1 - \left[\frac{1}{C_{t2}}\ln\left(e^{C_{t2}} - C_{t1}\,\varepsilon_{max}^{\delta_t}(\tilde{N} + N)\right)\right]^{\frac{1}{\gamma_t}} \tag{3.23}$$

En vista de las ecuaciones anteriores, se puede ver que el daño va aumentando con la fatiga, pero teniendo en cuenta el proceso de remodelación ósea, éste disminuye, ya que los osteoclastos reabsorven las microfracturas del tejido óseo, para que posteriormente, los osteoblastos depositen el osteoide, que está libre de daño. EL modelo supone que se realiza de forma uniforme en todo el hueso. Por lo tanto, el daño reparado por la remodelación debe ser proporcional al daño presente en ese volumen y al volumen de tejido que está siendo resorbido.

$$\Delta d_R = d\,\frac{\Delta V_r}{V_{bm}} = d\,\frac{k_{res} \cdot OC_a \cdot \Delta t}{V_{bm}} \tag{3.24}$$

Finalmente, el daño se actualiza teniendo en cuenta el daño acumulado por la fatiga y el reparado por la remodelación ósea, mediante la expresión:

$$d(t + \Delta t) = d(t) + \Delta d_A - \Delta d_R \tag{3.25}$$

3.6 Degradación de las propiedades de la fatiga por mineralización

Se conoce que cuanto más fase mineral hay, más rígido se vuelve el hueso, pero aumenta también la fragilidad. Se supone que, cuando mayor es el contenido de mineral, mayor es la degradación de las

propiedades de la fatiga y se han tomado las siguientes hipótesis:

- La forma de la curva d-N se mantiene independiente del contenido de mineral.
- Solo la vida de fatiga se ve afectada por el contenido de mineral, al cambiar K_t, manteniendo constante el exponente δ_t.
- El límite de fatiga es 10^7 ciclos. Se toma como valor más común $\beta = 2$.

K_t es obtenido mediante:

$$K_t([Ca]) = 10^7 \left(\frac{\varepsilon_u([Ca])}{\beta}\right)^{\delta_t} \tag{3.26}$$

ε_u depende de la concentración de [Ca] y se expresa de la siguiente forma [24]:

$$log\,\varepsilon_u = 25.452 - 11.341\ log\,[Ca] \tag{3.27}$$

3.7 Concentración de factores bioquímicos

Las ecuaciones que expresan las concentraciones de factores bioquímicos vienen dadas a partir de las ecuaciones de velocidad basadas en la cinética de acción de masas, pero se toman en su estado estable al ser los tiempos característicos de la respuesta celular mayores que el tiempo de reacción de la unión ligando-receptor [19].

$$TGF\beta(t) = \frac{n^{bone}_{TGF\beta} k_{res} OC_a(t)}{\widetilde{D}_{TGF\beta}} \tag{3.28}$$

$$RANK(t) = N^{RANK}_{OC_p} OC_p(t) \tag{3.29}$$

$$OPG(t) = \frac{\beta^{OPG}_{OB_a} OB_a(t) \pi^{PTH}_{rep,OB}}{\frac{\beta^{OPG}_{OB_a} OB_a(t) \pi^{PTH}_{rep,OB}}{OPG_{max} + \widetilde{D}_{OPG}}} \tag{3.30}$$

$$RANKL_{eff}(t) = N^{RANKL}_{OB_p} OB_p(t) \pi^{PTH}_{act,OB} \tag{3.31}$$

$$RANKL(t) = \frac{RANKL_{eff}\left(\frac{\beta_{RANKL} + P^{\psi_{bm}}_{RANKL}}{\beta_{RANKL} + \widetilde{D}_{RANKL} RANKL_{eff}}\right)}{1 + K_{A1[RANKL-OPG]} OPG + K_{A2[RANKL-RANK]} RANK} \tag{3.32}$$

$$RANKL_{tot} = RANKL(1 + K_{A1[RANKL-OPG]} OPG + K_{A2[RANKL-RANK]} RANK) \tag{3.33}$$

$$PTH(t) = \frac{P_{PTH}(t) + \beta_{PTH}}{\widetilde{D}_{PTH}} \tag{3.34}$$

3.8 Inclusión de la mineralización

Como se vio anteriormente en el apartado 2.1, los huesos están compuestos de una fase mineral, una orgánica y agua, por lo que tomando un volumen de referencia, $V_{RVE} = V_T$ que incluye además

los poros, se tiene:

$$V_T = V_{bm} + V_{vas} = V_m + V_o + V_w + V_{vas} \tag{3.35}$$

Los osteoblastos en la fase de formación depositan osteoide, que consiste solo en fase orgánica y agua, pero parte de este agua durante la mineralización pasa a ser fase inorgánica (mineral). Este proceso de transformación está compuesto de tres fases: una fase inicial en la que no se produce la mineralización y por eso se llama también: tiempo de retraso de la mineralización, durando de 6 a 22 días. Posteriormente hay una fase primaria, que es muy rápida (del orden de días), en la que se alcanza el 70 % de la mineralización máxima, y por último, se encuentra la fase secundaria, que se caracteriza porque el mineral se agrega a una tasa exponencial decreciente a medida que el tejido se satura con los minerales, es una fase muy lenta que puede ir desde los 6 meses a varios años [18]. El contenido de mineral en el hueso, normalmente se mide mediante la fracción de mineral, α, que es la relación entre la masa de mineral, m_m, y la masa seca, que es la suma de las fases orgánica y mineral [18].

$$\alpha = \frac{m_m}{m_o + m_m} = \frac{\rho_m V_m}{\rho_o V_o + \rho_m V_m} \tag{3.36}$$

Dando esa misma ecuación pero en volúmenes específicos, dividiendo todos los volúmenes entre V_b:

$$\alpha = \frac{\rho_m v_m}{\rho_o v_o + \rho_m v_m} \tag{3.37}$$

Manteniéndose que:

$$v_o + v_m + v_w = 1 \tag{3.38}$$

Como la mineralización está dividida en tres etapas, el volumen de mineralización está expresado como una función de tres partes dependiendo del tiempo en que se encuentre:

$$v_m(t) = \begin{cases} 0 & t \leq t_{mlt} \\ v_{prim}\frac{t - t_{mlt}}{t_{prim}} & t_{mlt} < t < t_{prim} + t_{mlt} \\ v_{max} - (v_{max} - v_{prim})e^{-\kappa(t - t_{prim} - t_{mlt})} & t_{prim} + t_{mlt} < t \end{cases} \tag{3.39}$$

Donde t_{mlt} y t_{prim} son respectivamente, la duración del tiempo de retraso de mineralización y la fase primaria. κ es un parámetro que mide la tasa de deposición mineral durante la fase secundaria, está tomado como 0,007.
La cantidad de volumen de mineral en un instante de tiempo viene dado por la siguiente expresión:

$$v_m(t+1) = \frac{\sum \bar{V}_{form}(t+1,i) v_m(i) + V_{res}(t+1) v_{max}}{V_b(t+1)} \tag{3.40}$$

Donde $V_{res}(t) = V_b(t) - \sum \bar{V}_{form}(t,i)$.
Para calcular el volumen de tejido óseo formado τ días atrás, pero todavía presente en el tiempo t, $\bar{V}_{form}(t,\tau)$, se calcula mediante un algoritmo de cola FIFO (First In First Out) (Figura 3.2) que se puede expresar según la siguiente ecuación:

$$\bar{V}_{form}(t+1,i+1) = \bar{V}_{form}(t,i) - V_{res}(t)\frac{\bar{V}_{form}(t,i)}{V_b(t)} \tag{3.41}$$

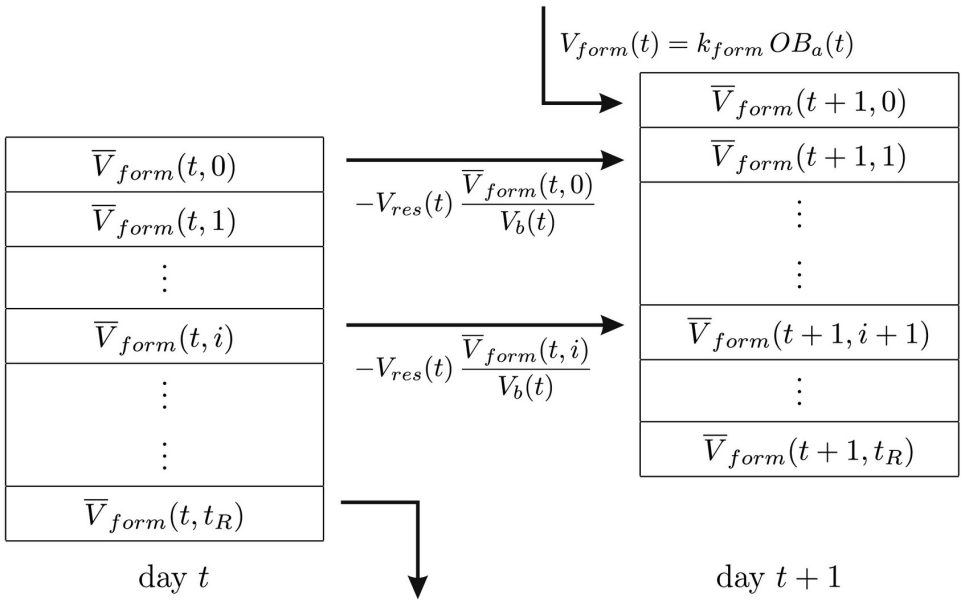


Figura 3.2 Algoritmo de cola FIFO utilizado para actualizar la distribución de parches de tejido de diferentes edades dentro del RVE [18].

3.9 Inclusión de Denosumab y osteoporosis (PMO)

En este apartado se va a ver cómo se ha incluido la enfermedad de la osteoporosis postmenopausica (PMO) y la adición de Denosumab en el modelo matemático [17].
En comparación con un hueso normal, uno con PMO tiene mayor cantidad de osteoclastos que osteoblastos provocando una mayor resorción frente a la formación de tejido óseo, provocando una pérdida ósea, además, está caracterizado por tener una producción de RANKL mayor, fomentando la aparición de osteoclastos activos, por lo que para tener en cuenta la enfermedad en el modelo, se va a suponer que el exceso de porducción de RANKL se va reduciendo con el tiempo.

$$P_{PMO}^{RANKL} = P_{RANKL}^{PMO,ini}\,\phi_{PMO}^{RANKL} \tag{3.42}$$

$$\phi_{PMO}^{RANKL} = \frac{\xi^2}{\xi^2 + (\frac{t - t_{PMO,ini}}{\tau_{PMO}^{RANKL}})^2} \tag{3.43}$$

Donde $P_{RANKL}^{PMO,ini}$ es el índice de de producción de exceso de RANKL que inicia la enfermedad. ϕ_{PMO}^{RANKL} es un factor de reducción, $t_{PMO,ini}$ es el tiempo de inicio de la enfermedad y ξ, es una constante de asociación RANKL - Denosumab.
Por último, se puede ver cómo además de que la falta de estrógenos aumenta la cantidad de osteoclastos producidos, la PMO influye también en la capacidad de mecanorespuesta, disminuyéndola al aumentar la apoptosis de osteocitos.

$$\phi_{PMO}^{mech} = exp(\frac{t - t_{PMO,ini}}{\tau_{PMO}^{RANKL}}) \tag{3.44}$$

Por otro lado, para ver la influencia del Denosumab en el modelo, hay que tener en cuenta de que es un fármaco monoclonal que tiene gran afinidad por el RANKL, por lo que cuanto mayor es su concentración, hay menos complejos RANKL-RANK que logran una menor $\pi_{OC_p,act}^{RANKL}$ y provoca una menor población de osteoclasatos precursores.

La concentración de RANKL sigue la siguiente expresión:

$$RANKL = \frac{RANKL_{eff}}{1 + K_{A1,[RANKL-OPG]}OPG + K_{A2,[RANKL-RANK]}RANK + \xi K_{A3,Denosumab}C_{Denosumab}} \left(\frac{\beta_{RANKL} + P_{RANKL-d}P_{RANKL}^{\psi_{bm}}}{\beta_{RANKL} + RANKL_{eff}\widetilde{D}_{RANKL}}\right) \tag{3.45}$$

$$[RANKL - RANK] = K_{A2,[RANKL-RANK]}RANKLRANK \tag{3.46}$$

$$\pi_{OC_p,act}^{RANKL} = \frac{[RANKL - RANK]}{[RANKL - RANK] + K_{A3,Denosumab}} \tag{3.47}$$

3.10 Ecuaciones diferenciales

Para terminar, se expone el sistema de ecuaciones diferenciales que rigen densidades de células óseas en el modelo conociendo con anterioridad los parámetros necesarios para resolverlo, se han considerado constantes OB_u y OC_p:

$$\frac{\partial OB_p}{\partial t} = D_{OB_u} \cdot \pi_{act,OBu}^{TGF-\beta} \cdot OB_u + P_{OB_p} \cdot \Pi_{act,OB_p}^{\psi_{bm}} \cdot OB_p - D_{OB_p} \cdot \pi_{rep,OB_p}^{TGF-\beta} \cdot OB_p \tag{3.48}$$

$$\frac{\partial OB_a}{\partial t} = D_{OB_p} \cdot \pi_{rep,OB_p}^{TGF-\beta} \cdot OB_p - A_{OB_a} \cdot OB_a \tag{3.49}$$

$$\frac{\partial OC_a}{\partial t} = D_{OC_p} \cdot \pi_{act,OC_p}^{RANKL} \cdot OC_p - A_{OC_a} \cdot \pi_{act,OC_p}^{TGF-\beta} \cdot OC_a \tag{3.50}$$

Por otro lado, la evolución de la porosidad vascular y la fracción de volumen de la matriz ósea vienen dadas por:

$$\frac{\partial f_{vas}}{\partial t} = -\frac{\partial f_{bm}}{\partial t} = -k_{form}OB_a + k_{res}OC_a \tag{3.51}$$

Se ha tenido en cuenta que la porosidad y la fracción de volumen matriz ósea cambian al actuar tanto los osteoclastos y los osteoblastos. Aunque como la capa de osteoide que depositan los osteoblastos mineraliza muy rápido en comparación con el proceso de remodelación en sí, se ha supuesto que la matriz ósea está mineralizada desde el principio y que, tanto la tasa de reabsorción de la matriz ósea y la tasa de nueva deposición de matriz ósea son constantes [19].

4 Redes neuronales

En este capítulo se van a explicar los fundamentos de las redes neuronales, que han sido utilizadas en nuestro trabajo, para su mayor compresión. También hay una sección con las redes neuronales densas, en la que se explica cómo son y cómo están formadas.
La red neuronal es una herramienta predictiva muy útil para este trabajo, ya que usando directamente el modelo matemático en la optimización, el coste computacional sería excesivo. Las redes están compuestas por una serie de entradas y salidas que se obtienen del modelo matemático y, de esta forma, se entrena la red. Estas herramientas proporcionan una función que para las entradas dadas, simula el modelo y predice las salidas de forma más rápida.

4.1 La neurona biológica

Las redes neuronales están inspiradas en el funcionamiento de las neuronas biológicas, que componen el sistema nervioso y reciben una serie de señales eléctricas de entrada y producen una señal de salida combinándolas.
Las neuronas están formadas por tres partes bien diferenciadas [25]: el soma o cuerpo de la célula, que es donde se aloja el núcleo y orgánulos, el axón, una fibra larga que sale del soma, y es el camino de salida de la señal generada por la neurona. Por último, se encuentran las dendritas, una serie de fibras más cortas y son la vía de entrada de las señales que se combinan en el cuerpo de la neurona (ver figura 4.1).

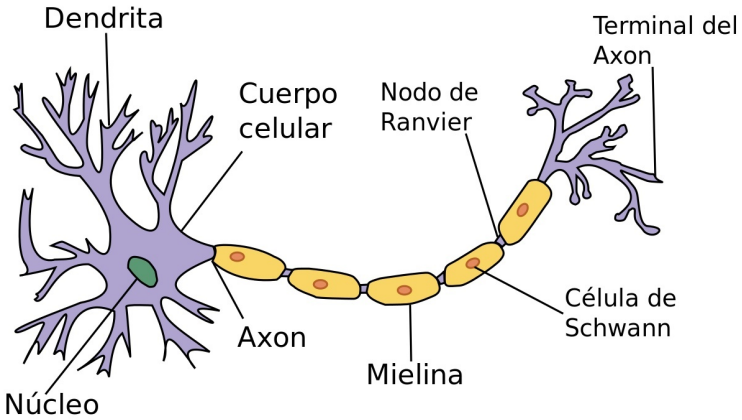


Figura 4.1 Neurona y partes principales [25].

La principal función de la neurona es la transmisión de impulsos eléctricos hacia otras células. Este proceso de procesamiento transmisión y recepción de impulsos se denomina sinapsis.

4.2 Fundamentos de las redes neuronales

Las redes neuronales, de la misma forma que el sistema nervioso, están formadas por neuronas artificiales. Estas neuronas artificiales están compuestas de una serie de elementos [26]:

- Un conjunto de entradas $x_j(t)$.
- Unos pesos sinápticos asociados a las entradas, w_{ij}.
- Una regla de propagación $h_i(t) = \sigma(w_{ij}, x_j(t))$. La más común es $h_i(t) = \sum w_{ij} x_j(t)$.
- Una función de activación $y_i(t) = f_i(h_i(t))$, que representa tanto al estado de activación como a la salida de la neurona.
- El umbral θ_i que se resta del potencial postsináptico.

Estos elementos vienen recogidos en la figura 4.2

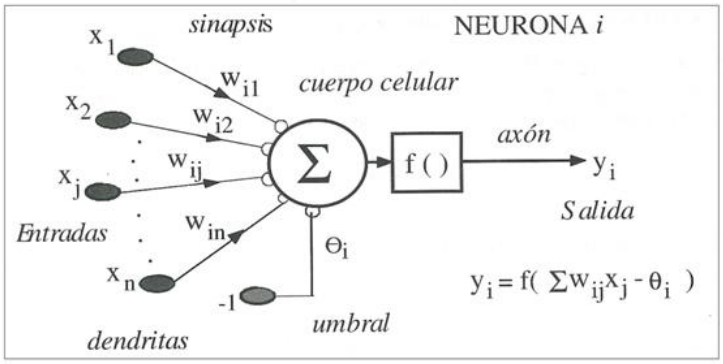


Figura 4.2 Neurona artificial y partes principales [26].

La expresión que refleja el comportamiento de una neurona artificial es la siguiente:

$$y_i(t) = f_i(\sum w_{ij} x_j(t)) \tag{4.1}$$

Hay distintas funciones de activación, pero aquí se van a detallar solamente las más usadas. En particular son: las escalón, lineal a trozos y sigmoidal [27].

- La función escalón viene definida en la siguiente expresión, y se utiliza principalmente para la clasificación, al ser la salida binaria muy útil en la selección de clases.

$$f_i(v) = \begin{cases} 1 & v \geq 0 \\ 0 & v < 0 \end{cases} \tag{4.2}$$

- La función lineal por partes viene dada por la siguiente ecuación, y su pendiente se asume que es la unidad. La forma de esta función se asemeja a la de un amplificador no lineal.

$$f_i(v) = \begin{cases} 1 & v \geq \frac{1}{2} \\ v & -\frac{1}{2} < v < \frac{1}{2} \\ 0 & v \leq -\frac{1}{2} \end{cases} \tag{4.3}$$

- La función sigmoidal tiene forma de "s" y es estrictamente creciente. Es una función muy utilizada en los algoritmos de aprendizaje debido a la derivavilidad en toda su curva, a

diferencia de las anteriores. Además, su derivada es positiva en toda la función. La expresión que define este tipo de función es:

$$f_i(v) = \frac{1}{1 + e^{-av}} \tag{4.4}$$

Donde a es el parámetro de la pendiente y siempre es mayor que 0.

Las redes neuronales son modelos predictivos que reciben como entrada un tensor (generalización de un vector con una cantidad arbitraria de dimensiones) y devuelve otro tensor correspondiente a la predicción. La predicción puede tener cualquier cantidad de dimensiones y tamaño, por lo que se puede predecir un único número (un tensor de una dimensión y longitud 1), o varios.
Para realizar una predicción, una red neuronal realiza operaciones sobre el tensor de entrada, como multiplicarlo por matrices, realizar sumas, etc. Estas operaciones involucran variables conocidas como parámetros que deben ajustarse para que la red funcione y la salida sea una predicción precisa. Por ejemplo, si multiplicamos el tensor de entrada por una matriz, los valores de dicha matriz podrían ser parámetros.
Los parámetros (pesos sinápticos) son normalmente inicializados con valores arbitrarios, y su ajuste se produce en un proceso de entrenamiento en el que la red "aprende" a partir de ejemplos en forma de tuplas (tensor de entrada, tensor de salida). Para ello, lo primero es definir una función de pérdida. Una función de pérdida recibe como entrada un conjunto de predicciones y con conjunto de salidas esperadas (la verdad que se intentaba predecir), y devuelve una medida del error de las predicciones. Por tanto, el objetivo del entrenamiento es reducir tanto como sea posible esa función de pérdida.
Para ajustar los parámetros de forma que se reduzca la pérdida, se suele llevar a cabo un procedimiento conocido como descenso por gradiente. Para ello, se realizan los siguientes pasos de forma iterativa:

- Se hace pasar un subconjunto de los ejemplos de entrenamiento, conocido como batch y con tamaño dado por el usuario, por la red.
- Se calcula la pérdida para las predicciones obtenidas.
- Se calcula la derivada de la pérdida en función de cada parámetro. Estas derivadas se calculan siguiendo un proceso conocido como "backpropagation" que calcula primero las derivadas de los parámetros al final de la red, y usa estas para calcular la de los anteriores.
- Según la derivada, ajusta cada parámetro en la "dirección" (incremento o decremento) que disminuiría la pérdida.

Este proceso, en un principio terminaría una vez se han recorrido todos los datos de enternamiento. Sin embargo, se pueden realizar varios recorridos de todos los datos. Cada recorrido se conoce como un epoch. Realizar más epochs involucra ajustar más los parámetros, lo que es beneficioso hasta cierto punto, ya que si el número de epochs es demasiado alto, la red se ajustará demasiado a los datos concretos que se usan como entrenamiento y no realizará predicciones precisas para otros datos. Este fenómeno se conoce como "overfitting" o sobreentrenamiento.
Una vez una red ha sido entrenada, se puede aplicar a tensores de entrada para obtener una salida sin modificar los parámetros.
Las redes neuronales han ganado muchísima popularidad en los últimos años, ya que, al contrario que otros modelos predictivos, son capaces de aproximar casi cualquier función, es decir, son capaces de realizar predicciones muy complejas, incluso no-lineales. Esto es posible gracias a la introducción en la red de operaciones no-lineales que permiten ir más allá de resultados obtenidos mediante multiplicaciones y sumas.

4.2.1 Tipos de aprendizajes

Las redes usan distintos métodos de aprendizaje, pero en general se basan en ir ajustando los pesos sinápticos en función de las entradas disponibles de forma que minimicen la función de pérdida, optimizando la respuesta de la red a las salidas esperadas. Hay tres tipos de aprendizaje en las redes [28]:

- Supervisado: la red dispone de los patrones de entrada y de salida (que se desean para esa entrada), de forma que en función de ellos se cambian los pesos sinápticos para ajustar las entradas a las salidas. Este tipo de aprendizaje es en el que se basa el backpropagation, el algoritmo que ha sido utilizado en la red desarrollada en este trabajo.
- No supervisado: este tipo de aprendizaje consiste en que no se presentan patrones objetivo, sino que solo patrones de entrada, de forma que la red es la que clasifica dichos patrones en función de sus características comunes.
- Reforzado: es un híbrido entre los dos últimos y se basa en que a partir de un patrón de entrada, el supervisor no enseña patrones objetivos, si no que únicamente dice si en su respuesta ha acertado o fallado.

4.3 Redes neuronales densas

Este tipo de red es el que se ha utilizado en este trabajo. Se caracteriza por estar dividida en componentes conocidos como capas. Cada capa recibe un tensor de entrada y devuelve un tensor de salida. Existen muchos tipos de capas, como capas convolucionales, recurrentes, etc. Sin embargo, la mayoría de estas solo son aplicables a casos específios, como imágenes o secuencias de datos. El tipo de capa más popular es la capa densa. Una capa densa devuelve como salida un tensor de tamaño especificado por el usuario, donde cada componente del vector suele representarse por una "neurona". Una neurona devuelve la suma de cada componente del tensor de entrada multiplicada por un peso, sumando un último parámetro conocido como "bias". Cada neurona tiene sus propios parámetros, por lo que el número de éstos es alto. El nombre de "capa densa" viene de cómo existen todas las conexiones posibles (multiplicaciones por parámetros) entre las componentes del tensor de entrada y las del tensor de salida (ver figura 4.3).

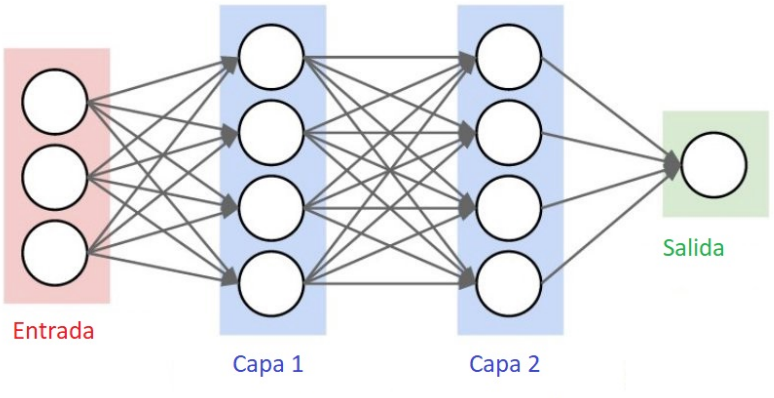


Figura 4.3 Red neuronal densa. Adaptada de [29].

Una red neuronal densa es, por tanto, aquella que consiste en una concatenación de capas densas, solo añadiendo capas auxiliares para normalización de datos e introducción de no-linealidad. Estas

redes son apropiadas para predicciones a partir de datos numéricos que no requieren arquitecturas complejas para su explotación. El número de capas y su tamaño puede ajustarse por el usuario, siendo lo típico empezar con capas más grandes y reducir su tamaño progresivamente de forma que la red neuronal se vea forzada a combinar la información sin tener que ajustar demasiados parámetros.
La arquitectura usada ha sido la de feedforward, que se caracteriza por tener una o más capas de neuronas con la función sigmoide seguidas de capas con funciones lineales (ver figura 4.4). La no linealidad que da la función sigmoide permite que la red aprenda tanto las relaciones lineales como no lineales de los vectores de entrada y salida. La capa de salida lineal permite que la red produzca valores fuera del rango de -1 a +1 [30].

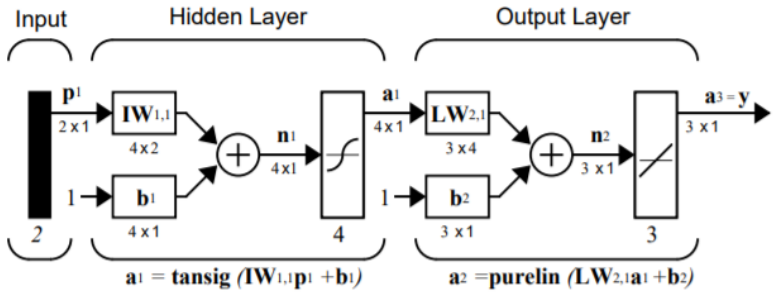


Figura 4.4 Red neuronal densa que se ha utlizado en este trabajo [30].

4.3.1 Creación de la red neuronal

El primer paso para entrenar una red feedforward es crear el objeto de red. Para ello, en Matlab se llama a la función feedforwardnet, que devuelve el objeto y tiene como argumentos la matriz de entrada, la de salida, los parámetros correspondientes al número de capas ocultas así como el número de neuronas, y por último, está la función de entrenamiento [30].
Las funciones de entrenamiento que tiene Matlab son varias, pero en este trabajo se va a centrar en tres de ellas, ya que han sido utilizadas en él.

- Trainlm: es la función que utiliza por defecto Matlab y generalmente es de las más rápidas rápidas. Sin embargo, utiliza bastante memoria. Esta función va actualizando los pesos y las bias de acuerdo a la optimización de Levenberg-Marquardt.
- Trainbr: esta función generalmente tarda más tiempo en entrenar la red. Actualiza los pesos y las bias de acuerdo a la optimización de Levenberg-Marquardt. Minimiza una combinación de errores cuadráticos y de los pesos, y luego determina la combinación correcta para producir una red que generalice bien. Este proceso se llama regularización bayesiana.
- Trainscg: es una función de entrenamiento en red que actualiza los valores de peso y bias de acuerdo con el método de gradiente conjugado escalado. Utiliza muy poca memoria.

Como se comentó anteriormente, este proceso de entrenamiento va recorriendo varias veces los datos (cada vuelta es un epoch), pero como esto no se puede realizar indefinidamente porque se sobreentrenaría la red, se toman del total de datos que se introducen a la red, un porcentaje destinado a la validación y otro al test. La validación consiste en ir evaluando el error de la red tras finalizar cada epoch, y ver cunándo empieza a aumentar. Momento en el que se para el entrenamiento. Como el conjunto de validación, al igual que el del test, están a parte del de entrenamiento (además de que cada vez que se crea una red toma valores diferentes al cogerlos de forma aleatoria del conjunto de datos inicial), el error de red que va evaluando la validación es un buen indicativo para ver cómo

será el del test. El test es una forma de ver cómo está prediciendo la red, y si los errores que se comenten son despreciables, se considera que la red está lista para su utilización.

Datos necesarios

En este caso, la red neuronal necesita de una serie de datos de entrada y de salida. Por un lado, son necesarios los datos de entrada a la red, los cuales coinciden con los datos de entrada al modelo y por otro lado, hacen falta los datos objetivo de la red que coinciden con la salida del modelo. Ambos conjuntos de datos se encuentran agrupados en matrices:
Matriz de entrada a la red:

- Filas: representan las distintas combinaciones numéricas de los parámetros.
- Columnas: representan los distintos parámetros de entrada del modelo.

Matriz de salida de la red (BDG, daño o fracción de ceniza):

- Filas: representan la evaluación en los distintos instantes de tiempo las diferentes combinaciones de parámetros.
- Columnas: representan los distintos instantes de tiempo donde se evalúa el BDG, daño o fracción de ceniza.

Para que la red neuronal se pueda entrenar correctamente, es necesario que tanto el tamaño de las filas de las matrices de entrada, como las de la salida sean iguales.
Para la configuración de la red, ha sido importante tener en cuenta el reparto de datos para el entrenamiento, testeo y validación. En todos los casos se ha tomado como el 70 % de los datos para el entrenamiento, el 15 % para la validación, mientras que el 15 % restante fue para el testeo.

5 Optimización

Los algoritmos de optimización son algoritmos cuyo fin es, dada una función con parámetros variables, encontrar los valores que maximizan o minimizan el resultado de la función. Por ejemplo, dada la función $y = (x-3)^2$, se podría optimizar x para que el valor de y sea mínimo, algo que ocurre cuando $x = 3$. A una asignación de valores se la conoce como "solución".
Este tipo de algoritmo ha sido necesario en este trabajo para calcular el tratamiento óptimo de Denosumab, partiendo de las funciones creadas en las redes neuronales, descritas anteriormente en el capítulo 4.
El objetivo de este capítulo es ilustrar de forma general qué tipos de métodos de optimización existen para luego profundizar más en los algoritmos genéticos. Se ha usado este algoritmo optimizador ya que, como se verá en el capítulo 7, la función tiene más de un mínimo, por lo que como se aprecia en este capítulo, esa es una de las características de este tipo de algoritmo que lo diferencian del resto.

5.1 Algoritmos de optimización

Los algoritmos de optimización son una transformación de un problema de optimización en una función de probabilidad del problema. En esta transformación se llevan a cabo un número de iteraciones (en el bucle de optimización del algoritmo) de forma que, se encuentra al menos un óptimo local.
En cuanto a la optimización se pueden dar dos tipos de búsqueda, optimización local u optimización global. Por un lado, la optimización local trata de acceder a los extremos de una función dentro de una región de las posibles soluciones candidatas, de manera que el extremo que se halle no tiene por qué ser el absoluto. Por otro lado, se encuentra la optimización global, que se basa en la búsqueda de los extremos de la función sin limitarse solo a una posible solución. Esta idea hace que los algoritmos de optimización global sean más complejos que los locales y tengan distintos métodos de optimización [31]:

- Deterministas: son aquellos que se basan en la garantía de encontrar un óptimo global sin componentes probabilísticas. Es una técnica exacta.
- Estocásticos: estos evalúan la función a optimizar en una serie de puntos escogidos al azar continuando el proceso de forma iterativa. Salvo el caso en que el número de puntos que vayan a ser evaluados tiendan al infinito, no se asegura de que el óptimo que se ha alcanzado es el global. Es una técnica probabilística en la que cada vez que se le introduce la misma entrada, da resultados diferentes, al tener la capacidad de buscar en regiones diferentes cada vez. Este resultado podría mejorarse aumentando esa capacidad de búsqueda del algoritmo.

5.2 Metaheurísticos

Para comprender mejor la optimización metaheurística, basta con fijarse en su denominación: Meta, es decir, que va más allá, y heurístico, que en griego significa encontrar, aunque también se puede entender como la utilización del conocimiento para llevar a cabo una determinada tarea. Otra definición dada por Glover es: "*métodos que integran de diversas maneras, procedimientos de mejora local y estrategias de alto nivel para crear un proceso capaz de escapar de óptimos locales y realizar una búsqueda robusta en el espacio de búsqueda. En su evolución, estos métodos han incorporado diferentes estrategias para evitar la convergencia a óptimos locales, especialmente a espacios de búsqueda complejos*" [32]. Es un método de optimización global probabilístico y sus principales características son [31]:

- Son de uso general y aplicables a gran número de problemas, ya que no se restringen a problemas específicos.
- En general son fácilmente implementables (utilizan pocas líneas de código y pueden ser reutilizables).
- Es sencilla la definición de implementaciones paralelas de las metaheurísticas existentes, debido a la estructura común que tienen todas ellas.
- No cuenta con mucha base teórica, ya que está todavía en desarrollo. Lo único que se puede afirmar es que si hay un número que tiende a infinito de puntos de evaluación, la solución encontrada es el óptimo global, debido a su naturaleza estocástica.
- Los resultados que encuentran son aproximados, y se desconoce cómo de cerca están del óptimo global.

La figura 5.1 reúne las principales metaheurísticas existentes, dentro de las cuales se encuentran los algoritmos genéticos, de los que se va a profundizar más en la siguiente sección.

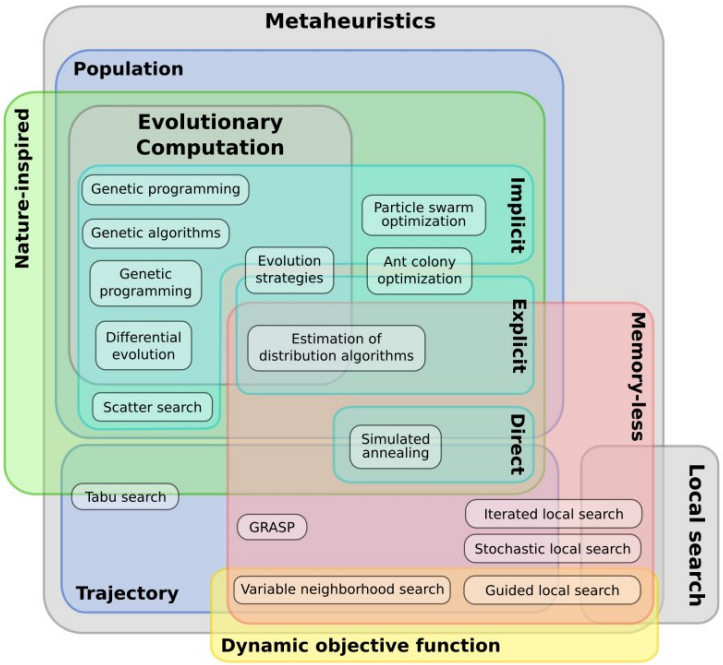


Figura 5.1 Esquema que reúne las metaheurísticas existentes [33].

5.3 Algoritmos genéticos

En esta sección se va a ver en más profundidad cómo son los algoritmos genéticos, partiendo de su inspiración en la teoría de Darwin, también se explicarán las definiciones más comunes en este tipo de algoritmo, además de su funcionamiento.

5.3.1 Bases biológicas

El origen de los algoritmos genéticos está en la evolución biológica de Darwin, en la que partiendo de una población inicial, se seleccionan los individuos más cualificados para posteriormente reproducirlos entre ellos y mutarlos para así obtener la siguiente generación de individuos que estarán más adaptados que la anterior. Esto se ve en la propia naturaleza, en la cual los individuos que está menos capacitados tienen menor número de descendientes, y por tanto, la supervivencia de ese gen está destinada al fracaso. Este proceso es estocástico, ya que cuanto mayor adaptación tiene un individuo, es más probable que obtenga recursos y descendencia, sin embargo, esto no está asegurado en ningún caso.

5.3.2 Principales definiciones de la computación evolutiva

Basándose en la teoría de Darwin, los algoritmos evolutivos están compuestos de una serie de elementos cuyos conceptos generales se van a exponer a continuación [31].

- Individuo: esta denominación parte de que los algoritmos genéticos están basados en la evolución de las especies, por ello, esta componente principal se denomina así, y corresponde con una posible solución del problema. Generalmente se parte de individuos aleatorios que van mejorando según las iteraciones del algoritmo hasta alcanzar un criterio de parada definido.
- Población: esto, al igual que en biología, corresponde a un conjunto de individuos, pero en el caso de la heurística, es un conjunto de individuos pertenecientes a una iteración del

algoritmo.

- Función de aptitud: es la función matemática que va a utilizarse para la optimización, de forma que devuelve un valor numérico para cada punto del espacio del problema.
- Criterios de terminación: como en los problemas heurísticos el criterio de optimización es probabilístico y no se suele hallar una solución exacta, hay que definir cuándo se quiere finalizar las iteraciones del algoritmo. Para ello, estos se basan en una serie de criterios, donde los más comunes son:
  - Cuando se establece un número máximo de iteraciones (este es el criterio que se ha utilizado en este trabajo, imponiendo 15 iteraciones) o también, se puede fijar un tiempo máximo de ejecución. Este método es muy útil ya que permite acotar el tiempo de ejecución y cuanto mayor sea, las soluciones obtenidas estarán más ajustadas y refinadas. Por eso hay que buscar un equilibrio entre el valor buscado y el tiempo de ejecución.
  - Imponiendo un número máximo de iteraciones a partir de la cual, ya no hay cambio en las soluciones. Esto ocurre cuando el problema se ha quedado atrapado en un mínimo local y le es imposible encontrar otro.
  - Mirando la calidad de una solución obtenida, es decir, el algoritmo cesa de realizar iteraciones cuando el resultado tiene un grado de optimalidad definido a priori. Este criterio permite la comparación de distintos métodos de optimización, al alcanzarse soluciones de la misma calidad.
- Convergencia: este concepto se utiliza tanto para referirse a la capacidad que tiene el algoritmo para alcanzar el óptimo global, o el hecho de concentración de los individuos en una pequeña porción del espacio de búsqueda, pero no tiene que estar cercano a un óptimo. Esto generalmente no es deseable, ya que la falta de diversidad favorece la aparición de un óptimo local, con la imposibilidad de encontrar el global.
- Penalidad de muerte: consiste en que aquellos individuos que, inicialmente no cumplan con las restricciones del problema, se descartan directamente. Esto es de gran utilidad si el tamaño de la región válida es grande en comparación con la región no factible del problema.
- Funciones de penalización: éstas agregan un término a la función objetivo, de forma que si hay determinadas soluciones que se encuentran en una región no factible las penaliza, por ejemplo, incrementa su valor un determinado porcentaje para que de siempre un valor de aptitud peor que otros y se descarte en las siguientes iteraciones del algoritmo.

5.3.3 Fundamentos de los algoritmos genéticos

Las opciones a optimizar pueden tener cientos de parámetros usados de forma compleja y no-lineal, lo que hace que no sea posible optimizar fácilmente con, por ejemplo, cálculos de derivadas, o probando todas las soluciones posibles.
Para poder realizar optimización en estas circunstancias, se idearon los conocidos como algoritmos genéticos o evolutivos. Son algoritmos inspirados en los procesos de selección natural que buscan una solución que puede no ser la estrictamente óptima, pero que intenta obtener una aproximación razonable. Para ello comienzan con un conjunto de soluciones aleatorias, llamado población, en la que cada solución se la llama genoma. Cada genoma debe estar representado por partes alterables a las que se llama genes. Por ejemplo, en la función anterior, se podría representar el valor de x como una cadena de números binarios, cada uno de ellos un gen.
La representación de soluciones mediante genomas formados por genes permite combinarlos (descendencia), seleccionar los mejores de la población (selección natural) o introducir modificaciones aleatorias (mutaciones). De esta manera, para obtener mejores genomas (es decir, que optimizan

más la función), se actualiza iterativamente la población, algo que puede ocurrir de varias maneras, por ejemplo:

- Obteniendo combinaciones de los mejores genomas de la población actual.
- Pasando a la nueva población algunos de los mejores genomas de la población actual.
- Alterando aleatoriamente uno de los genomas de la población actual.

Este proceso busca un equilibrio entre mantener aquello que hace a los mejores genoma buenos, e introducir cierta variabilidad para que el proceso no se estanque y se exploren nuevas soluciones. Para combinar genomas, es frecuente combinar porciones aleatorias de sus genes, aunque algunos problemas con soluciones más complejas requieren representaciones y estrategias de combinación más elaboradas. Es destacable cómo, mientras sea posible representar las soluciones como genomas, es posible aplicar algoritmos evolutivos a cualquier función.
Las iteraciones se repiten un número fijo de iteraciones, o hasta que la función alcance un umbral de optimización. Llegado este punto, la solución final es el mejor genoma de la población.
El esquema de la figura 5.2 muestra el funcionamiento básico de estos algoritmos de optimización.

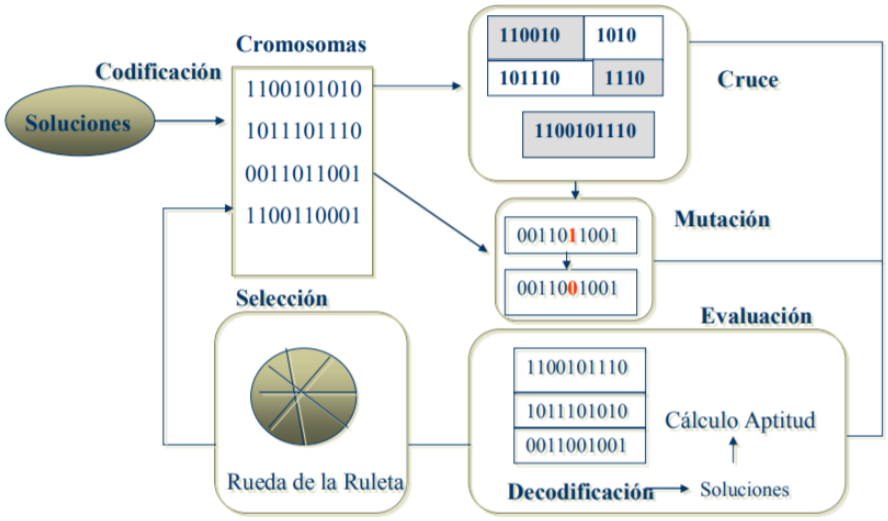


Figura 5.2 Esquema de un algoritmo genético simple. [34].

5.3.4 Ventajas e inconvenientes de los algoritmos genéticos

Antes de usar el algoritmo genético en un problema hay que conocer cuáles son sus puntos fuertes y cuáles los débiles. Por ello, basándonos en el artículo [34], se muestran las principales ventajas e inconvenientes de los mismos.
Los algoritmos genéticos son una buena herramienta al tener las siguientes ventajas:

- Operan con varias soluciones a la vez (son intrínsecamente paralelos), de forma que si siguiendo una dirección en la búsqueda de la solcución óptima, ésta no se hallara, simplemente se desecha esta solución subóptima y se buscan otros caminos, en lugar de empezar de nuevo como harían otras técnicas.
- Trabajan muy bien con problemas formados por más de un mínimo.
- Son hábiles para manipular varios parámtetros de forma simultánea.

- No es necesario un conocimiento específico sobre el problema que se está intentando resolver, sino que simplemente realizan cambios aleatorios en las soluciones candidatas a óptimas y ven si esos cambios han producido mejoras.
- Resulta fácil ejecutarlos en las modernas arquitecturas masivas en paralelo.
- Usan operadores probabilísticos en lugar de determinísticos, que usan otras técnicas de optimización.

Sin embargo, esta herramienta tiene una serie de dificultades en:

- La definición de una representación del problema.
- Dependiendo de los parámetros usados para la configuración del algoritmo, como son el tamaño de la población, número de generaciones etc. el problema puede llegar a no converger nunca.
- Puede converger demasiado temprano si un individuo que es más apto que la mayoría de sus competidores emerge muy pronto en el curso de la ejecución, entonces se reproduciría tanto que mermaría la diversidad de la población demasiado pronto, provocando que el algoritmo converja hacia el óptimo local que representa ese individuo, en lugar de rastrear el paisaje adaptativo lo bastante a fondo para encontrar el óptimo global.

6 Métodos

El objetivo de este capítulo es exponer lo que se ha hecho y justificar las tomas de decisiones hechas a lo largo de todo el trabajo. Para ello, se va a comenzar con la muestra de un diagrama que resume las actividades principales y las relaciones entre ellas 6.1.

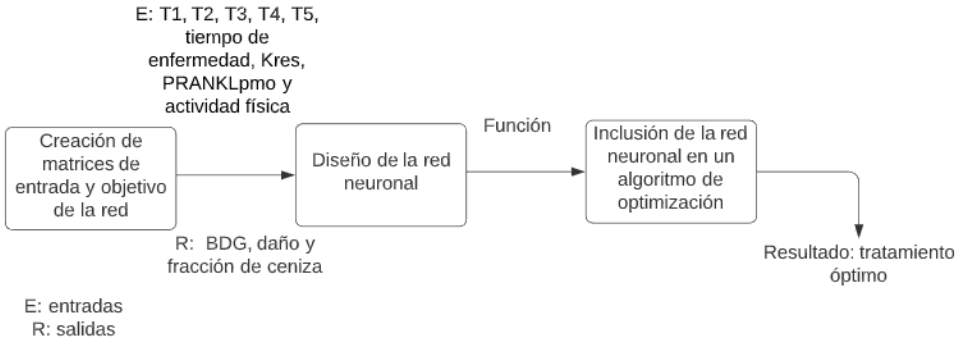


Figura 6.1 Esquema de las actividades principales.

6.1 Obtención del conjunto de datos

El propósito de la creación de las matrices de entrada y objetivo de la red, es para tener un conjunto de datos con los que entrenar la red neuronal. En primer lugar, se parte de un modelo matemático que determina mediante ecuaciones diferenciales la variación de las variables dinámicas del problema, como se vió de forma más exhaustiva en el capítulo 3. En dicho modelo, se fijan la mayoría de los parámetros de los que dependen, salvo las variables T1, T2, T3, T4, T5, k_{res}, P^{PMO}_{RANKL}, $\Delta\sigma$ y el tiempo de enfermedad, que son las que van a servir de entrada a la red, y por tanto hay que tomar varias combinaciones de ellas de forma aleatoria. Del modelo se obtienen una serie de salidas que son la ganancia de densidad ósea (BDG), el daño y la fracción de ceniza. Posteriormente, estos datos son llevados a una red neuronal que producirá una función. Esta función tendrá como datos de entrada los correspondientes a los parámetros de entrada de la red, mientras que su salida serán los correspondientes a la los valores de la matriz objetivo de la red. Esta función, se incluirá en un algoritmo de optimización que dará finalmente, el tratamiento óptimo.
El proceso de obtención de datos viene explicado en los diagramas de flujo 6.2, 6.3 y 6.4.
Partiendo de las constantes mencionadas anteriormente, se han definido unos parámetros que son los característicos del tratamiento:

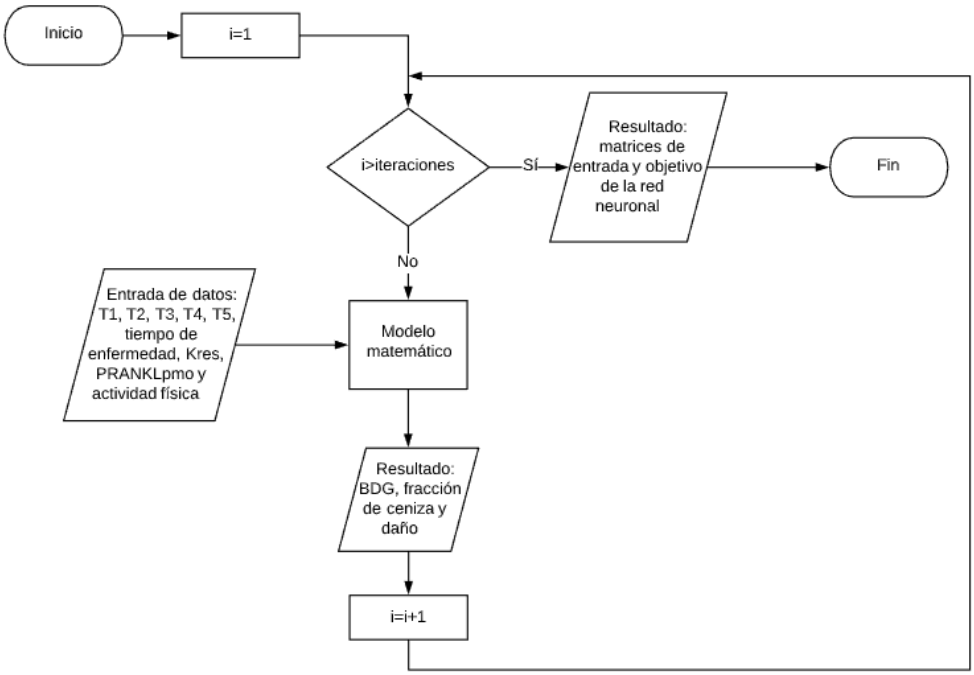


Figura 6.2 Diagrama de la obtención de las matrices objetivo y de entrada de la red.

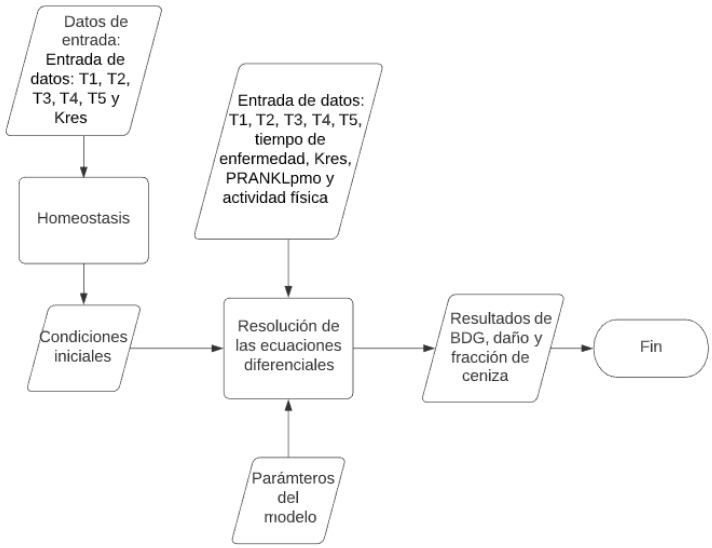


Figura 6.3 Diagrama general del modelo matemático.

- T1: es la dosis del tratamiento normal en mg (también llamado tratamiento 1).
- T2: es el tiempo en días entre el tratamiento normal.
- T3: es la dosis del tratamiento en la segunda fase en mg (también llamado tratamiento 2).

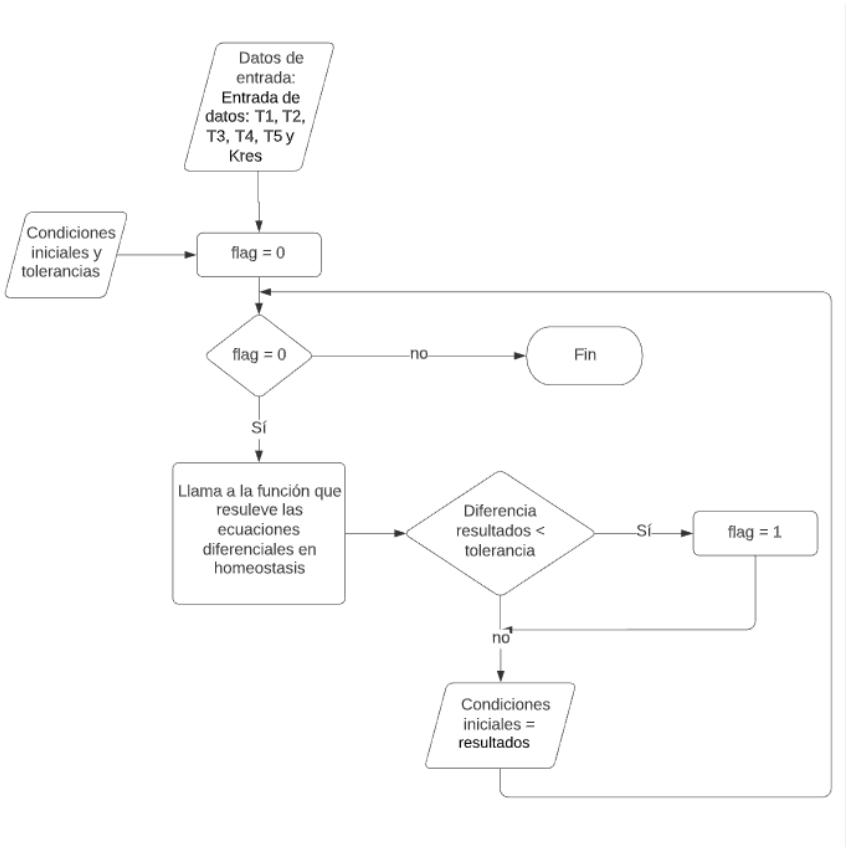


Figura 6.4 Diagrama de la obtención de las condiciones iniciales del modelo mediante la homeostasis.

- T4: es el tiempo en días entre el tratamiento de la segunda fase.
- T5: es el tiempo entre los dos tratamientos.

Para obtener los resultados, se hizo un programa llamado *funcion_principal* que reunía todo lo necesario para realizar los cálculos. Esto se hizo, para que de forma automática se realizaran todas las simulaciones necesarias, al ser un número elevado de ellas. Sin embargo, las distintas iteraciones se tuvieron que hacer por partes, ya que el coste computacional que requería crear una matriz de orden de magnitud de 1000 filas en una única ejecución del programa, era demasiado elevado. Por esto, las simulaciones tenían como máximo 200 iteraciones, que era el máximo número que permitía el ordenador en una ejecución del programa. Posteriormente, se tuvo que recurrir a un programa auxiliar (*Formacion_matrices.m*) que reuniera todas las matrices de 200 filas y formar así las deseadas.
La simulación del modelo matemático realizada en el programa *funcion_principal*, se particularizó para el caso en el que el tratamiento de la enfermedad se prolongara durante 10 años. Los resultados de BDG, fracción de ceniza y daño se fueron obteniendo cada 6 meses entre el primer año de tratamiento y el 10º año.
Para evaluar con que número de casos de entrenamiento se conseguía un mejor diseño se tomaron matrices de entrada y objetivo de tamaño 2000, 1000 y 500 datos.

6.2 Diseño de la red neuronal

Para llegar al objetivo final del trabajo, hay que conocer los resultados que da el modelo matemático para, que con el tratamiento que se suministre, se obtengan los mejores resultados posibles en

ganancia de densidad ósea, daño y fracción de ceniza. Para ello, hace falta conocer una función que de directamente esos datos partiendo del tratamiento y las características del paciente, ya que el coste computacional sería excesivo si se implementara el modelo matemático directamente en la optimización. Por esto, se vio la necesidad de crear una red neuronal que predijera dichos parámetros, y después utilizarla en un algoritmo de optimización, como se verá más adelante.

El diagrama 6.5 resume lo necesario para una comprensión de lo realizado en el trabajo. Esto consistió primeramente en la adición de las matrices de entrada y objetivo de la red. De estos datos, se tomaron el 75 % de forma aleatoria para el entrenamiento, mientras que un 15 % fue para la validación y el otro 15 % restante para el testeo. A continuación, se personalizan los parámetros de la red neuronal, es decir, el número de capas, el número de neuronas por capa y la función de entrenamiento. Estos parámetros se han ido cambiando cada vez que se entrenaba la red neuronal, para ver cuáles eran los que hacían que se ajustara mejor la red. Por esto, se probaron distintas combinaciones de ellos y los que se utilizaron están explicados en el capítulo 7. Posteriormente, se realiza el entrenamiento de la red, que consiste en un entrenamiento por lotes (epoch), de forma que va recorriendo todos los datos de entrenamiento varias veces, y a su vez, va validando la red con los datos destinados a ello, con esto, Matlab disminuye la probabilidad de sobreentrenamiento de la red, ya que si el número de epoch o vueltas es demasiado elevado, los parámetros de entrenamiento se ajustarán muy bien a la predicción, pero esto será debido a que la red lo ha aprendido de "memoria", por lo que el resto de parámetros los predecirá mal, de esta forma si la red es válida, es decir, que el error de validación está por debajo de lo establecido (es un valor que viene definido en Matlab), se obtiene la función correspondiente. Con esta función y los datos de testeo, se calculan los errores de la red, es decir, la diferencia que se produce entre los valores que deberían salir (que están incluidos en las matrices objetivo de la red) y los que da la función obtenida de la red para esos datos de testeo, y se comparan con los obtenidos de otras redes anteriores. La que tenga un error más pequeño es la que mejor predice los valores de la salida del modelo, y por lo tanto, los que más se ajustan a la realidad.

La primera red que se realizó predecía las tres variables de salida a la vez, es decir, el daño, BDG y la fracción de ceniza, pero como se verá posteriormente en el capítulo 7, los resultados obtenidos no eran muy buenos, por lo que se decidió entrenar las variables de salida por separado, y así independizando las redes, conseguir un mejor ajuste.

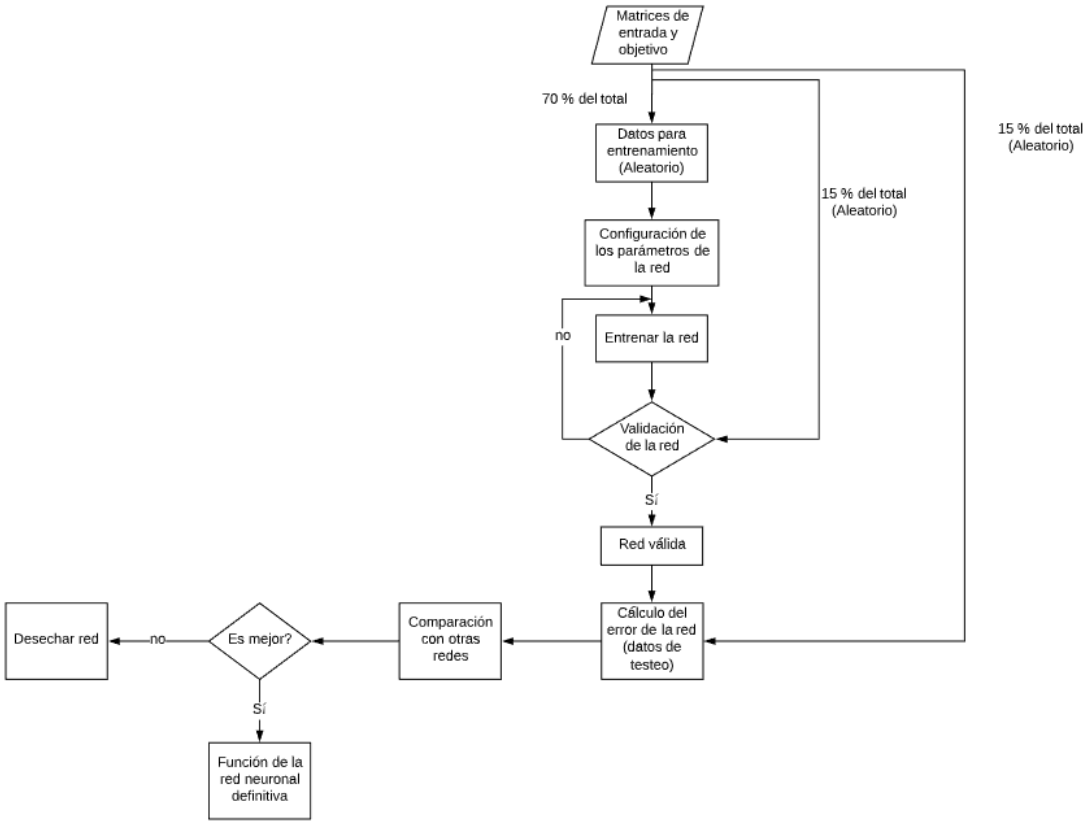


Figura 6.5 Diagrama de flujo de entrenamiento de la red neuronal.

6.2.1 Comparación de redes

Aunque la red se entrene con unas mismas características en cuanto al número de capas, neuronas, función de entrenamiento y números de casos, no siempre da la misma solución, ya que Matlab inicializa con índices de la matriz distintos cada vez, por lo que para unos mismos datos de entrenamiento, se ha ejecutado varias veces mostrando los mejores resultados de la curva de regresión mostradas en el capítulo 7. Posteriormente, de cada uno de ellos, se elige el que de una mejor curva de test, ya que esa indica cómo se aproximan los resultados que daría la función generada por la red a los resultados que deberían de dar realmente, que son los directamente obtenidos del modelo matemático aplicado anteriormente.

6.3 Optimización

El objetivo final de este trabajo es conseguir un tratamiento óptimo para una determinada paciente, por esto, hay que utilizar un algoritmo de optimización. Sin embargo, para ver cuál era el que iba a usarse había que realizar unos estudios previos.

6.3.1 Estudios previos

En primer lugar, se estudió cómo influyen cada una de las variables de entrada características del tratamiento en las salidas, para ver si se alcanza un solo mínimo absoluto o si por otra parte, hay más mínimos locales, ya que dependiendo de esto se utlizan unos algoritmos de optimización u otros. Para ello, se iban variando de dos en dos los parámetros del tratamiento, por ejemplo, se variaban T1 y T3 manteniendo constante el resto. Se tomaba un año cualquiera de la salida de BDG para ver cómo influye en ese año determinado la variación de los parámetros anteriores, y así se construye

una gráfica tridimensional, en la que se verán directamente los mínimos locales o el mínimo único. Con un programa llamado *estudio_met_opt* (ver en el anexo), se hizo una primera prueba en la que se mantenían constantes los valores de T1, T3 y T5 (tomados de forma aleatoria), variando T2 y T4 dentro del rango establecido. Entonces, mediante las funciones obtenidas en la red neuronal, se iban calculando para un año en concreto, la salida BDG en función de T2 y T4. Si las superficies obtenidas tenían más de un mínimo, se debía llevar a cabo mediante un algoritmo genético la optimización, ya que un método de gradiente no serviría al quedarse atascado en un mínimo local, que depende del mínimo que se estimó inicialmente, es decir, la solución depende de la estimación inicial, mientras que en un algoritmo genético, al abarcar toda la superficie que se quiere optimizar, no tiende a quedarse con el primer mínimo que encuentra, sino que intenta buscar el absoluto.

6.3.2 Aplicación del algoritmo genético

Tras haber deducido que el algoritmo genético es el mejor para este tipo de aplicación, se realizan una serie de pruebas para ponerlo en funcionamiento. Para ello, se tomaron inicialmente una población de tamaño 200 y generaciones de 250. Posteriormente, se tomaron mayores poblaciones y generaciones, para contrastar los resultados que se obtenían y, viendo que convergían en valores similares, y que a mayor población el tiempo computacional crecía, se escogieron los primeros valores. Por otro lado, tomando menores valores de poblaciones y de generaciones los valores discernían, por lo que se descartaron estas opciones.
En cuanto al programa realizado, en primer lugar, era necesaria una llamada a los parámetros característicos del paciente, que son introducidos por el usuario, donde en todo el proceso de optimización se mantienen constantes. A continuación se tienen en cuenta las restricciones a las que están sometidas las variables a optimizar, como se verá en el capítulo 7. Estas restricciones se han tomado de forma que limiten los valores que se obtienen dentro de unos rangos típicos, es decir, se han establecido esos valores para que el resultado esté controlado.
Posteriormente, se lleva a cabo el bucle de optimización, en el que se van cambiando las variables a optimizar de forma aleatoria (realmente es pseudoaleatoria) dentro de los rangos establecidos y se va llamando a la función a optimizar (para buscar el mínimo), a la que se le van pasando los parámetros constantes definidos por el usuario, y las variables del tratamiento que van cambiando en cada iteración del bucle de optimización. La función de optimización se realizó de forma que llamaba a las funciones obtenidas de las mejores redes neuronales (se verá en el capítulo 7), de forma que se obtenían las salidas en los distintos instantes de tiempo del BDG, daño y fracción de ceniza. Con estos valores, se calculó el promedio aritmético de BDG y en cuanto a los valores del daño y de la fracción de ceniza se procedió de otra manera. Se utilizó un bucle que iba recorriendo cada componente de los vectores de estas salidas, y si las componentes superaban un cierto valor, se penalizaba. Se estableció que los límites fueran 0,72 para la fracción de ceniza y 0,03 para el daño, ya que por encima de estos valores, el daño podría llegar a fracturar el hueso y en cuanto a la fracción de ceniza, si los valores son altos, la mineralización fragilizaría al hueso. La función de optimización se puede resumir en la siguiente expresión:

$$F_{optimizacin} = -\frac{\sum_{i=1}^{n} BDG_i}{n} + penalizaciones \tag{6.1}$$

Donde n es el tamaño del vector de BDG, o el número de instantes en que se evalúa el BDG.
El diagrama 6.6 representa cómo funciona un algoritmo genético.

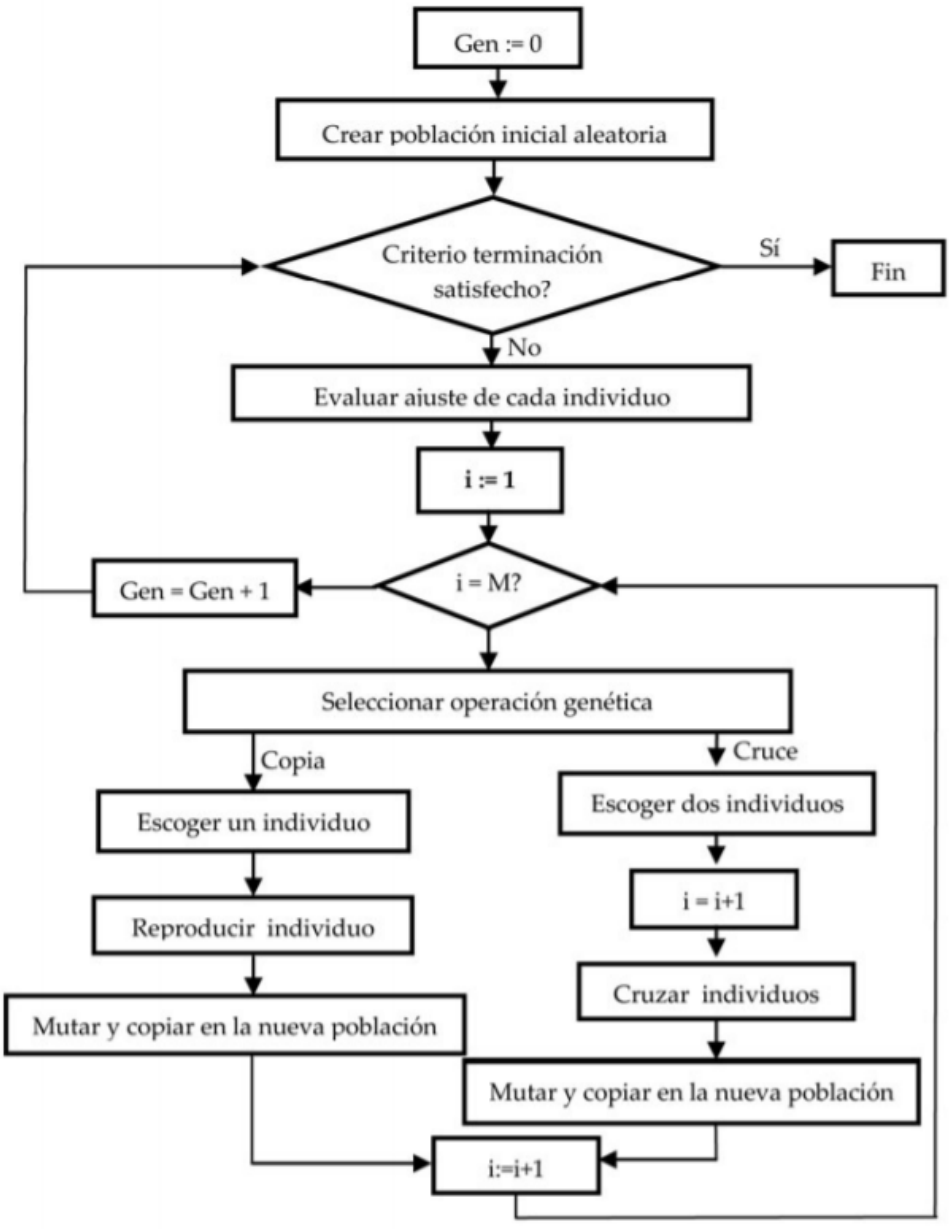


Figura 6.6 Diagrama de flujo de programación genética [35].

7 Resultados y discusión

El objetivo de este capítulo es exponer los resultados obtenidos en las distintas fases del trabajo. Para ello se usarán gráficas y tablas, que posteriormente se comentarán a modo de discusión.

7.1 Obtención del conjunto de datos

Los resultados obtenidos en primer lugar fueron las matrices de entrada y objetivo de la red neuronal. Unas matrices de tamaño 2000, 1000 y 500 datos.
Las columnas de las matrices de entrada correspondían al conjunto de parámetros de entrada, T1, T2, T3, T4, T5, k_{res}, P^{PMO}_{RANKL}, $\Delta\sigma$ y el tiempo de enfermedad, que se encontraban dentro de unos rango de valores establecidos. Se han dividido estos datos en dos grupos, según si en la herramienta final son las variables a optimizar (ver tabla 7.1) o, si en cambio, son los parámetros que introducirá el usuario final del programa y que son característicos del paciente (ver tabla 7.2). Por otro lado, las filas de las matrices eran las distintas variaciones de los parámetros escogidos de forma aleatoria mediante un comando de Matlab, como se verá en el capítulo 8.

Tabla 7.1 Rangos de variación de los parámetros que se van a optimizar en el programa final.

Parámetro	Variación	
	Mínimo	Máximo
T1 (mg)	10	120
T2 (días)	30	360
T3 (mg)	10	120
T4 (días)	30	360
T5 (días)	720	3600

Las matrices de salida se obtuvieron mediante las matrices de entrada anteriores y aplicando el modelo matemático para cada fila de datos de entrada. Así se construyen las matrices de salida fila a fila (ver figura 7.1).

Tabla 7.2 Rangos de variación de los parámetros que se van a introducir por el usuario final.

Parámetro	Variación	
	Mínimo	Máximo
k_{res} (%)	100	300
P_{PMO}^{RANKL} (pM/día)	1000	4000
Tiempo de enfermedad (años)	1	8
$\Delta\sigma$ (MPa)	-0,2	0,2

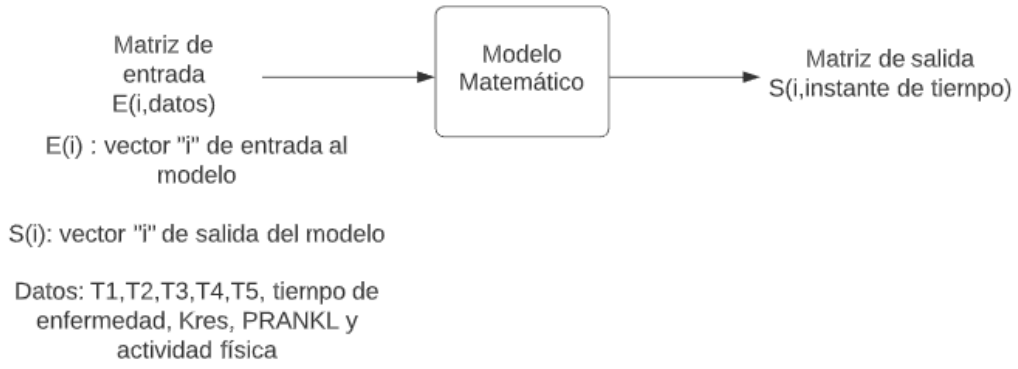


Figura 7.1 Diagrama de la creación de las matrices de salida partiendo de las de entrada.

7.2 Predicción del BDG, daño y fracción de ceniza

En esta sección se van a ver los resultados obtenidos de los distintos entrenamientos de las redes neuronales.

7.2.1 Primeros ensayos

Los resultados obtenidos de los primeros ensayos que, consistían en redes que incluían a todas las salidas en su predicción, no eran muy buenos, ya que los resultados que se obtenían en algunos valores no tenían sentido biológico, por ejemplo, había algunos instantes de tiempo en que el daño daba valores negativos. Por esto se realizó una red neuronal para cada una de las salidas, BDG, daño y fracción de ceniza, y para cada una de las matrices de 2000, 1000 y 500 filas.

7.2.2 Comparación de redes

Se realizaron varias pruebas que consistían en la combinación de distintas condiciones de entrenamiento como son, las funciones de ajuste, el número de capas en la red, así como las neuronas que había en cada capa. Los resultados obtenidos dependiendo de la red y de las distintas salidas analizadas fueron:
Para la fracción de ceniza, α:

- Matriz de 500 filas:

- Condiciones de entrenamiento: 1 capa con 20 neuronas y la función trainlm (ver figura 7.2).
- Condiciones de entrenamiento: 2 capas con 20 y 10 neuronas y la función trainlm (ver figura 7.3).
- Condiciones de entrenamiento: 2 capas con 20 y 10 neuronas y la función trainbr (ver figura 7.4).
- Condiciones de entrenamiento: 3 capas con 10, 10 y 5 neuronas y la función trainbr (ver figura 7.5).
- Condiciones de entrenamiento: 3 capas con 40, 10 y 10 neuronas y la función trainlm (ver figura 7.6).

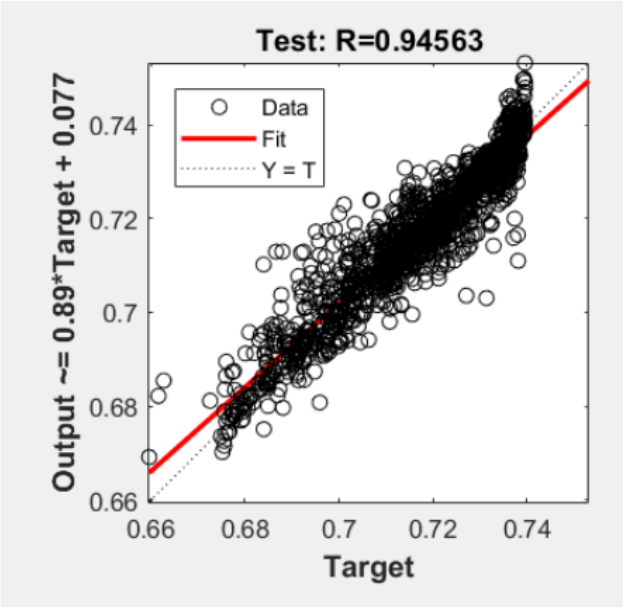


Figura 7.2 Curva de regresión que da la red para el caso de la salida de la fracción de ceniza, un entrenamiento con 500 casos y unas condiciones de: 1 capa con 20 neuronas y la función trainlm.

De entre las mejores conseguidas para la fracción de ceniza en el caso de 500 datos, se obtiene que la mejor red es el caso de 3 capas con 10, 10 y 5 neuronas y la función trainbr, correspondiente a la figura 7.5.

- Matriz de 1000 filas:
  - Condiciones de entrenamiento: 1 capa con 20 neuronas y la función trainlm (ver figura 7.7).
  - Condiciones de entrenamiento: 2 capas con 20 y 10 neuronas y la función trainlm (ver figura 7.8).
  - Condiciones de entrenamiento: 2 capas con 20 y 10 neuronas y la función trainbr (ver figura 7.9).
  - Condiciones de entrenamiento: 2 capas con 40 y 10 neuronas y la función trainbr (ver figura 7.10).

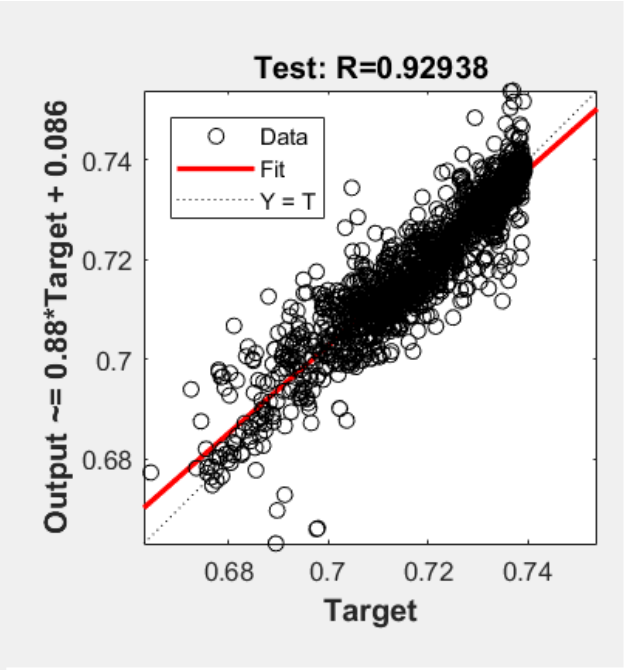


Figura 7.3 Curva de regresión que da la red para el caso de la salida de la fracción de ceniza, un entrenamiento con 500 casos y unas condiciones de: 2 capas con 20 y 10 neuronas y la función trainlm.

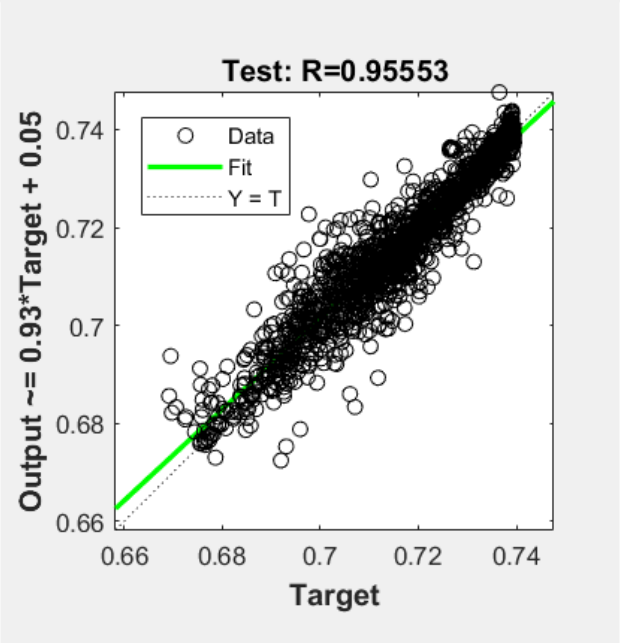


Figura 7.4 Curva de regresión que da la red para el caso de la salida de la fracción de ceniza, un entrenamiento con 500 casos y unas condiciones de: 2 capas con 20 y 10 neuronas y la función trainbr.

– Condiciones de entrenamiento: 3 capas con 40, 10 y 10 neuronas y la función trainlm (ver figura 7.11).

Viendo el coeficiente R en las curvas de regresión, se obtiene que la mejor red es el caso de 2 capas con 40 y 10 neuronas y la función trainbr, correspondiente a la figura 7.10.

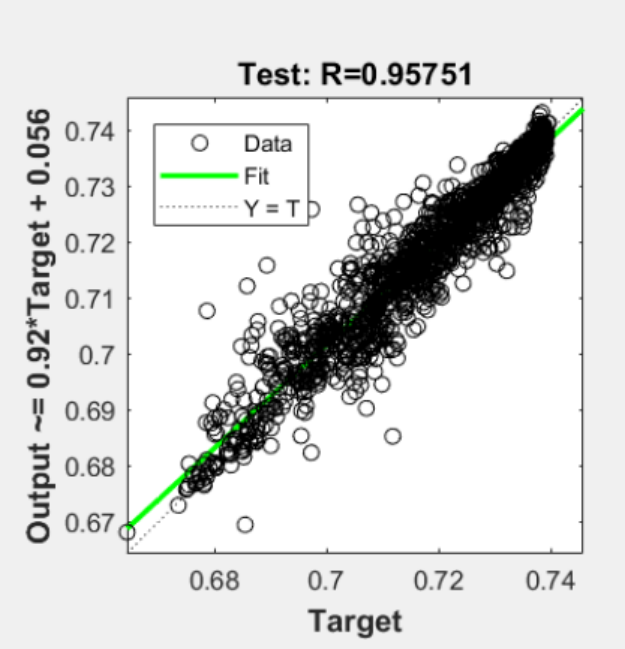


Figura 7.5 Curva de regresión que da la red para el caso de la salida de la fracción de ceniza, un entrenamiento con 500 casos y unas condiciones de: 3 capas con 10, 10 y 5 neuronas y la función trainbr.

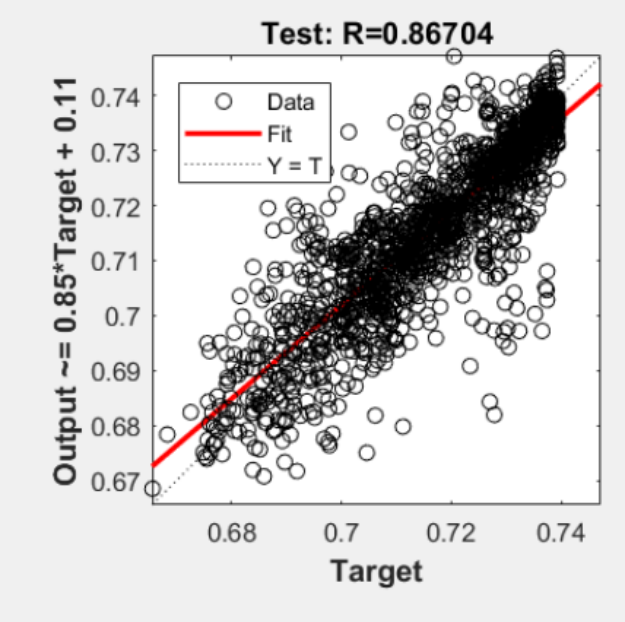


Figura 7.6 Curva de regresión que da la red para el caso de la salida de la fracción de ceniza, un entrenamiento con 500 casos y unas condiciones de: 3 capas con 40, 10 y 10 neuronas y la función trainlm.

- Matriz de 2000 filas:

  – Condiciones de entrenamiento: 1 capa con 20 neuronas y la función trainlm (ver figura 7.12).

Figura 7.7 Curva de regresión que da la red para el caso de la salida de la fracción de ceniza, un entrenamiento con 1000 casos y unas condiciones de: 1 capa con 20 neuronas y la función trainlm.

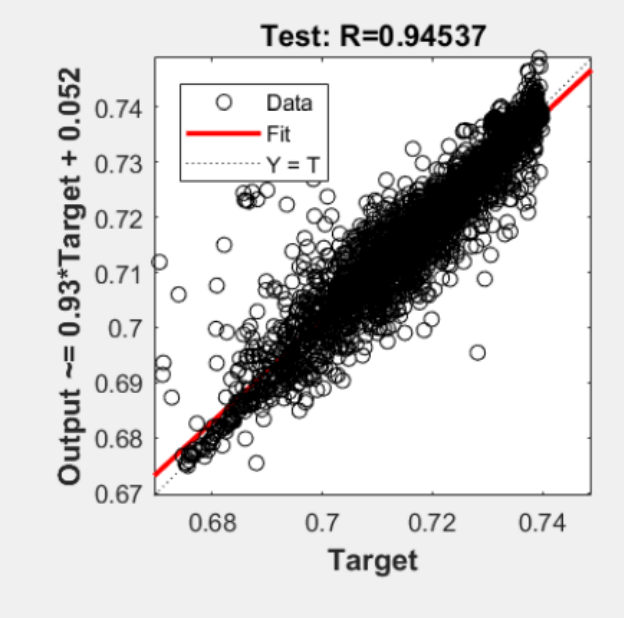


Figura 7.8 Curva de regresión que da la red para el caso de la salida de la fracción de ceniza, un entrenamiento con 1000 casos y unas condiciones de: 2 capas con 20 y 10 neuronas y la función trainlm.

- Condiciones de entrenamiento: 2 capas con 20 y 10 neuronas y la función trainlm (ver figura 7.13).
- Condiciones de entrenamiento: 2 capas con 20 y 10 neuronas y la función trainbr (ver figura 7.14).
- Condiciones de entrenamiento: 2 capas con 40 y 10 neuronas y la función trainbr (ver figura 7.15).

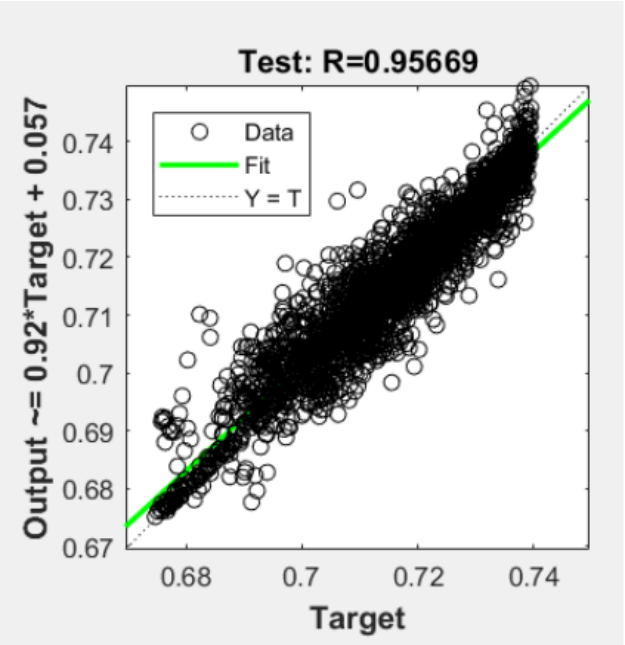


Figura 7.9 Curva de regresión que da la red para el caso de la salida de la fracción de ceniza, un entrenamiento con 1000 casos y unas condiciones de: 2 capas con 20 y 10 neuronas y la función trainbr.

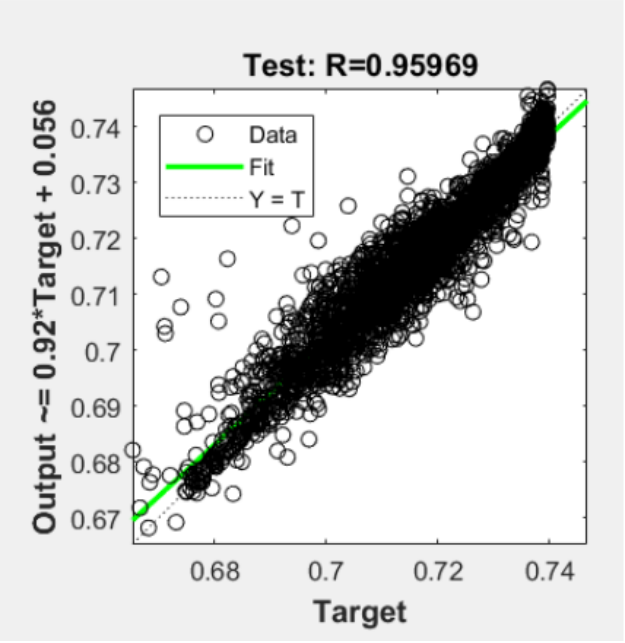


Figura 7.10 Curva de regresión que da la red para el caso de la salida de la fracción de ceniza, un entrenamiento con 1000 casos y unas condiciones de: 2 capas con 40 y 10 neuronas y la función trainbr.

– Condiciones de entrenamiento: 3 capas con 40, 10 y 10 neuronas y la función trainlm (ver figura 7.16).

Viendo el coeficiente R en las curvas de regresión, se obtiene que la mejor red es el caso de 2 capas con 40 y 10 neuronas y la función trainbr, correspondiente a la figura 7.15.

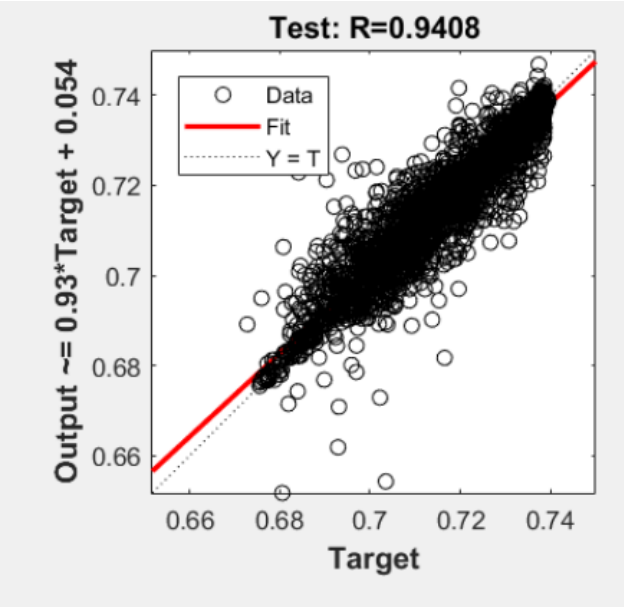


Figura 7.11 Curva de regresión que da la red para el caso de la salida de la fracción de ceniza, un entrenamiento con 1000 casos y unas condiciones de: 3 capas con 40, 10 y 10 neuronas y la función trainlm.

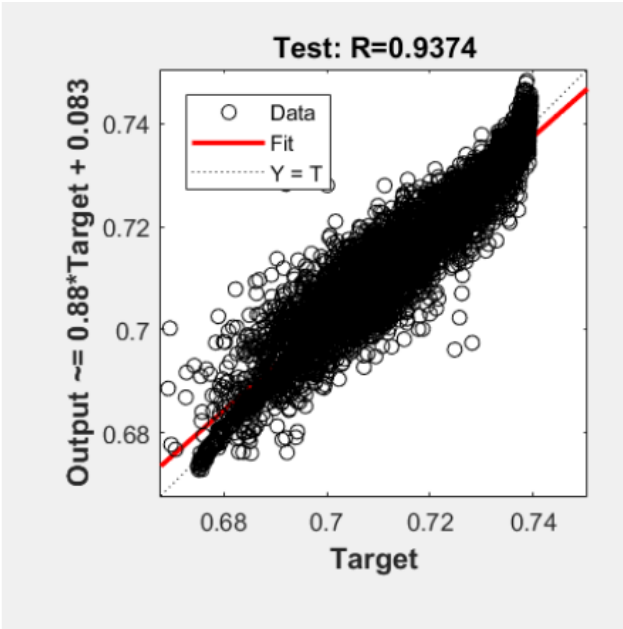


Figura 7.12 Curva de regresión que da la red para el caso de la salida de la fracción de ceniza, un entrenamiento con 2000 casos y unas condiciones de: 1 capa con 20 neuronas y la función trainlm.

Porcentaje de ganancia ósea BDG:

- Matriz de 500 filas:
  - Condiciones de entrenamiento: 1 capa con 20 neuronas y la función trainlm (ver figura 7.17).

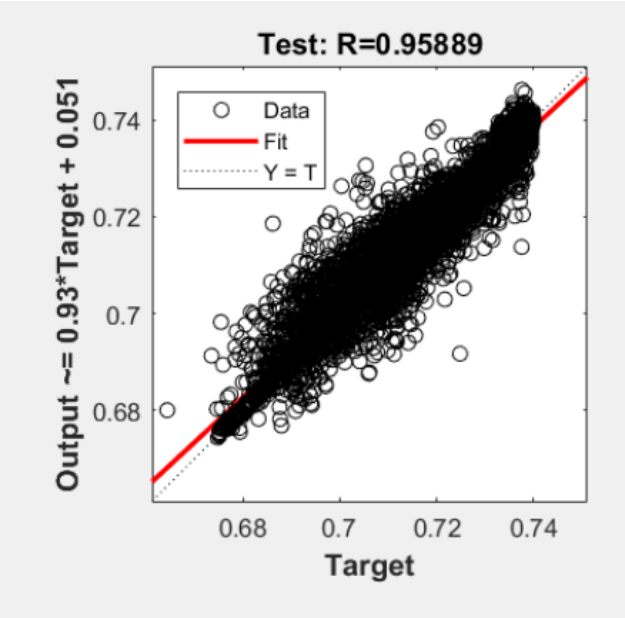


Figura 7.13 Curva de regresión que da la red para el caso de la salida de la fracción de ceniza, un entrenamiento con 2000 casos y unas condiciones de: 2 capas con 20 y 10 neuronas y la función trainlm.

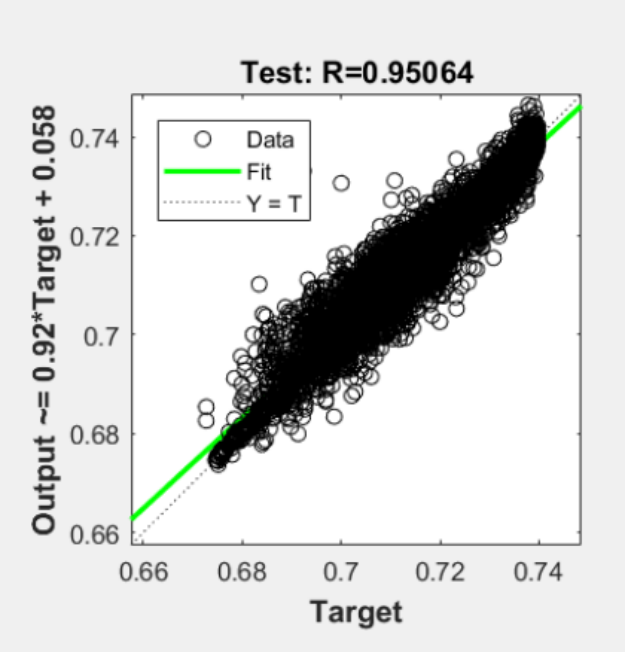


Figura 7.14 Curva de regresión que da la red para el caso de la salida de la fracción de ceniza, un entrenamiento con 2000 casos y unas condiciones de: 2 capas con 20 y 10 neuronas y la función trainbr.

- Condiciones de entrenamiento: 2 capas con 20 y 10 neuronas y la función trainlm (ver figura 7.18).
- Condiciones de entrenamiento: 2 capas con 20 y 10 neuronas y la función trainbr (ver figura 7.19).

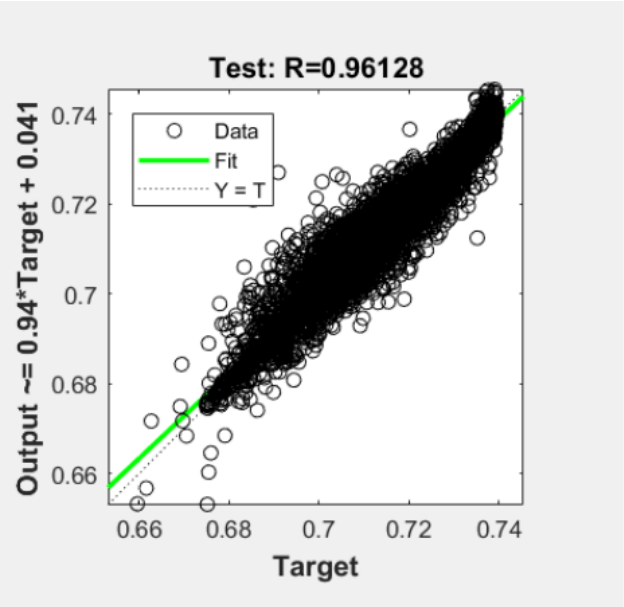


Figura 7.15 Curva de regresión que da la red para el caso de la salida de la fracción de ceniza, un entrenamiento con 2000 casos y unas condiciones de: 2 capas con 40 y 10 neuronas y la función trainbr.

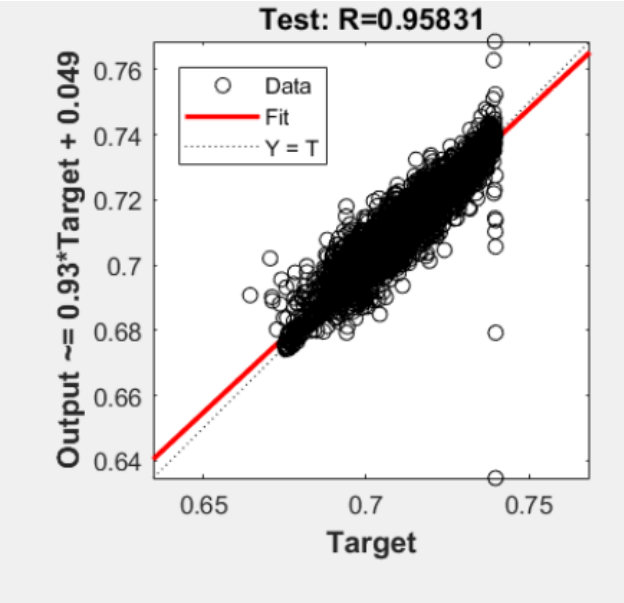


Figura 7.16 Curva de regresión que da la red para el caso de la salida de la fracción de ceniza, un entrenamiento con 2000 casos y unas condiciones de: 3 capas con 40, 10 y 10 neuronas y la función trainlm.

  - Condiciones de entrenamiento: 2 capas con 40 y 10 neuronas y la función trainbr (ver figura 7.20).

Viendo el coeficiente R en las curvas de regresión, se obtiene que la mejor red es el caso de 2 capas con 40 y 10 neuronas y la función trainbr, correspondiente a la figura 7.20.

- Matriz de 1000 filas:

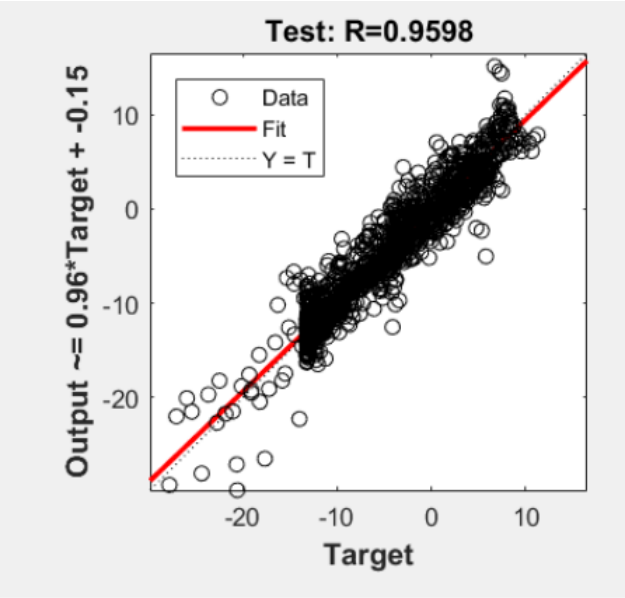


Figura 7.17 Curva de regresión que da la red para el caso de la salida del BDG, un entrenamiento con 500 casos y unas condiciones de: 1 capa con 20 neuronas y la función trainlm.

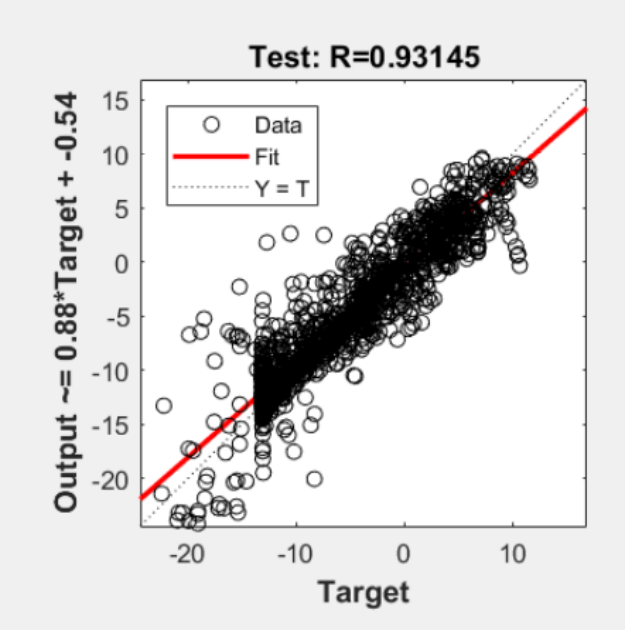


Figura 7.18 Curva de regresión que da la red para el caso de la salida del BDG, un entrenamiento con 500 casos y unas condiciones de: 2 capas con 20 y 10 neuronas y la función trainlm.

- Condiciones de entrenamiento: 2 capas con 20 y 10 neuronas y la función trainlm (ver figura 7.21).
- Condiciones de entrenamiento: 2 capas con 20 y 10 neuronas y la función trainbr (ver figura 7.22).
- Condiciones de entrenamiento: 2 capas con 40 y 10 neuronas y la función trainbr (ver figura 7.23).

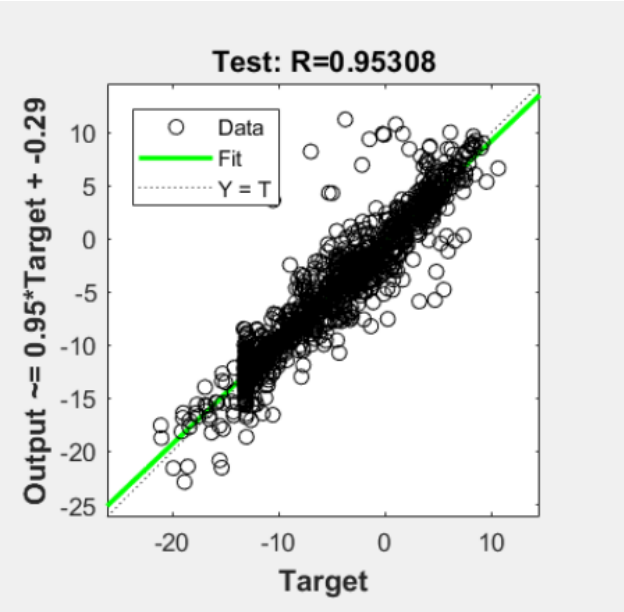


Figura 7.19 Curva de regresión que da la red para el caso de la salida del BDG, un entrenamiento con 500 casos y unas condiciones de: 2 capas con 20 y 10 neuronas y la función trainbr.

Figura 7.20 Curva de regresión que da la red para el caso de la salida del BDG, un entrenamiento con 500 casos y unas condiciones de: 2 capas con 40 y 10 neuronas y la función trainbr.

Viendo el coeficiente R en las curvas de regresión, se obtiene que la mejor red es el caso de 2 capas con 40 y 10 neuronas y la función trainbr, correspondiente a la figura 7.23.

- Matriz de 2000 filas:
  - Condiciones de entrenamiento: 1 capa con 20 neuronas y la función trainlm (ver figura 7.24).
  - Condiciones de entrenamiento: 2 capas con 20 y 10 neuronas y la función trainlm (ver figura 7.25).

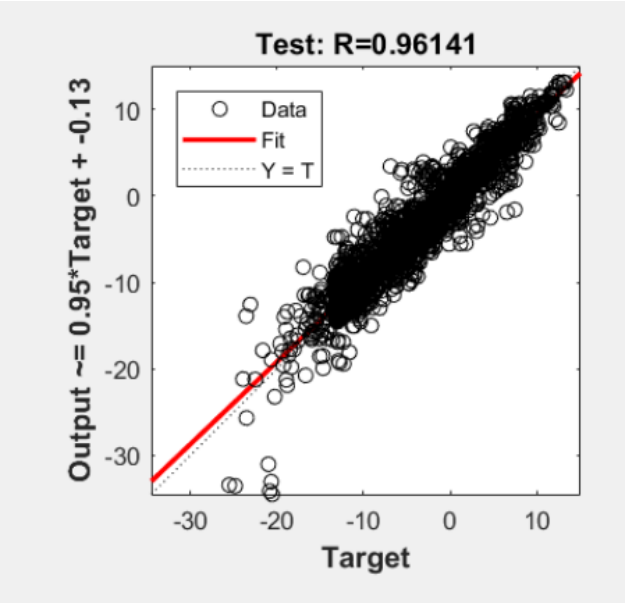


Figura 7.21 Curva de regresión que da la red para el caso de la salida del BDG, un entrenamiento con 1000 casos y unas condiciones de: 2 capas con 20 y 10 neuronas y la función trainlm.

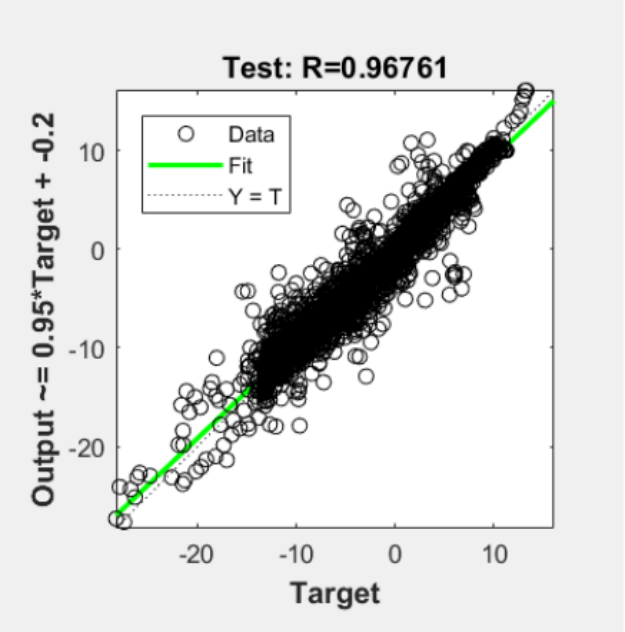


Figura 7.22 Curva de regresión que da la red para el caso de la salida del BDG, un entrenamiento con 1000 casos y unas condiciones de: 2 capas con 20 y 10 neuronas y la función trainbr.

- Condiciones de entrenamiento: 2 capas con 20 y 10 neuronas y la función trainbr (ver figura 7.26).
- Condiciones de entrenamiento: 2 capas con 40 y 10 neuronas y la función trainbr (ver figura 7.27).

Viendo el coeficiente R en las curvas de regresión, se obtiene que la mejor red es el caso de 2 capas con 40 y 10 neuronas y la función trainbr, correspondiente a la figura 7.27.

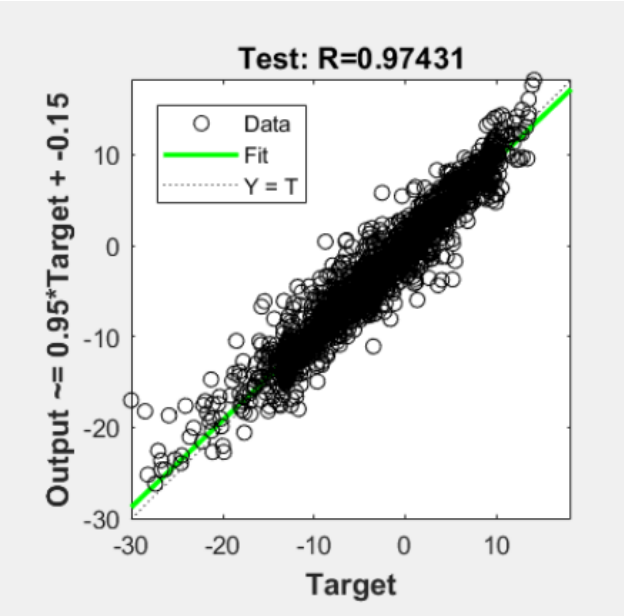


Figura 7.23 Curva de regresión que da la red para el caso de la salida del BDG, un entrenamiento con 1000 casos y unas condiciones de: 2 capas con 40 y 10 neuronas y la función trainbr.

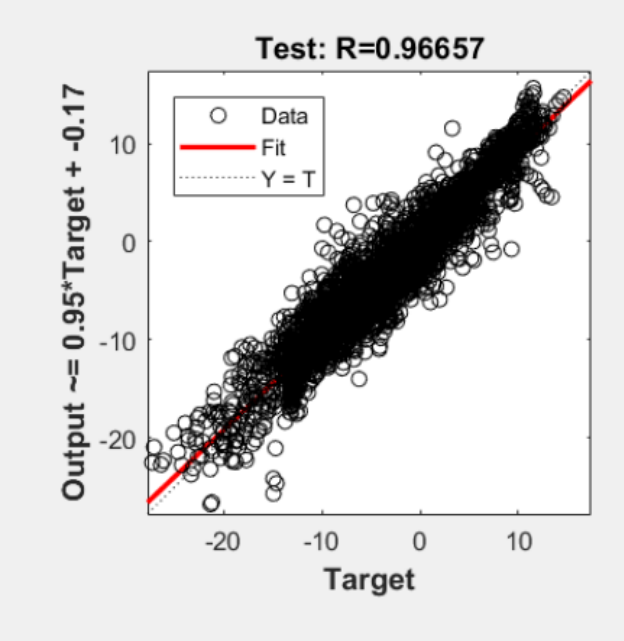


Figura 7.24 Curva de regresión que da la red para el caso de la salida del BDG, un entrenamiento con 2000 casos y unas condiciones de: 1 capa con 20 neuronas y la función trainlm.

Como comentario a las gráficas obtenidas para el daño, se puede ver que los valores se concentran o, cerca de 0 o en torno a 1, lo normal y mejor es que estén rondando los primeros, ya que en los segundos indicaría que el hueso ha sufrido una fractura, al estar los valores en tanto por uno.
Daño:

- Matriz de 500 filas:

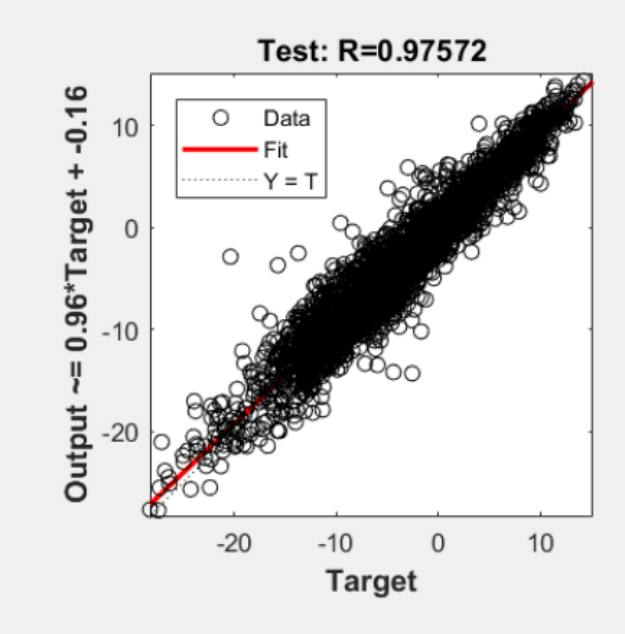


Figura 7.25 Curva de regresión que da la red para el caso de la salida del BDG, un entrenamiento con 2000 casos y unas condiciones de: 2 capas con 20 y 10 neuronas y la función trainlm.

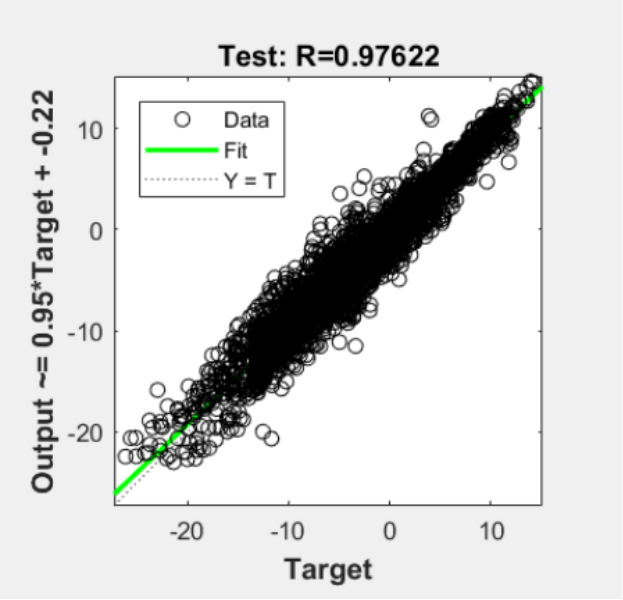


Figura 7.26 Curva de regresión que da la red para el caso de la salida del BDG, un entrenamiento con 2000 casos y unas condiciones de: 2 capa con 20 y 10 neuronas y la función trainbr.

- Condiciones de entrenamiento: 1 capa con 20 neuronas y la función trainlm (ver figura 7.28).
- Condiciones de entrenamiento: 2 capas con 20 y 10 neuronas y la función trainlm (ver figura 7.29).

Viendo el coeficiente R en las curvas de regresión, se obtiene que la mejor red es el caso de 1 capa con 20 neuronas y la función trainlm, correspondiente a la figura 7.28.

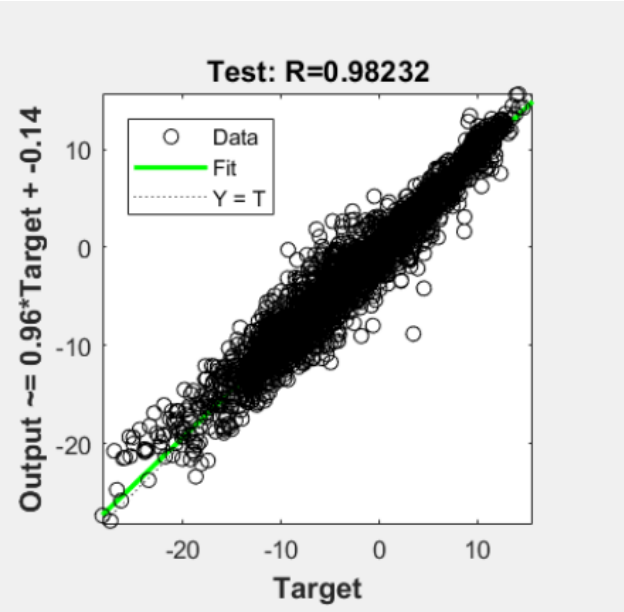


Figura 7.27 Curva de regresión que da la red para el caso de la salida del BDG, un entrenamiento con 2000 casos y unas condiciones de: 2 capa con 40 y 10 neuronas y la función trainbr.

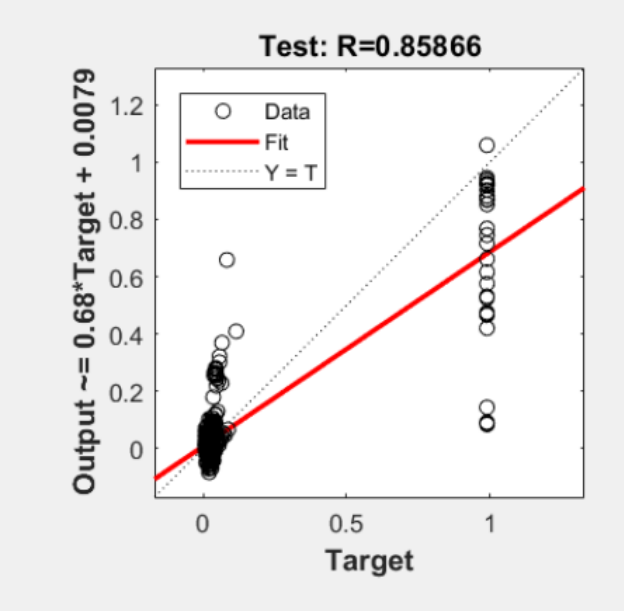


Figura 7.28 Curva de regresión que da la red para el caso de la salida del daño, un entrenamiento con 500 casos y unas condiciones de: 1 capa con 20 neuronas y la función trainlm.

- Matriz de 1000 filas:
  - Condiciones de entrenamiento: 2 capas con 20 y 10 neuronas y la función trainlm (ver figura 7.30).
  - Condiciones de entrenamiento: 2 capas con 20 y 10 neuronas y la función trainbr (ver figura 7.31).

Viendo el coeficiente R en las curvas de regresión, se obtiene que la mejor red es el caso de 2 capas con 20 y 10 neuronas y la función trainbr, correspondiente a la figura 7.31.

Figura 7.29 Curva de regresión que da la red para el caso de la salida del daño, un entrenamiento con 500 casos y unas condiciones de: 2 capas con 20 y 10 neuronas y la función trainlm.

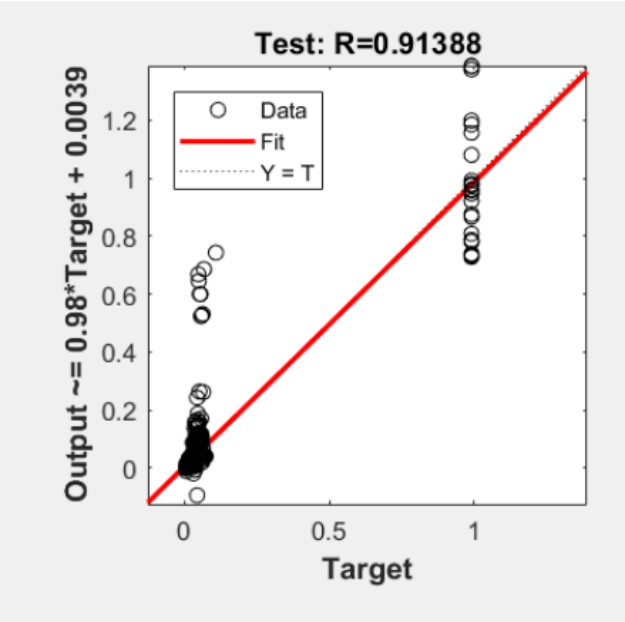


Figura 7.30 Curva de regresión que da la red para el caso de la salida del daño, un entrenamiento con 1000 casos y unas condiciones de: 2 capas con 20 y 10 neuronas y la función trainlm.

- Matriz de 2000 filas:
  - Condiciones de entrenamiento: 1 capa con 20 neuronas y la función trainlm (ver figura 7.32).
  - Condiciones de entrenamiento: 2 capas con 20 y 10 neuronas y la función trainlm (ver figura 7.33).
  - Condiciones de entrenamiento: 3 capas con 40, 20 y 10 neuronas y la función trainscg (ver figura 7.34).

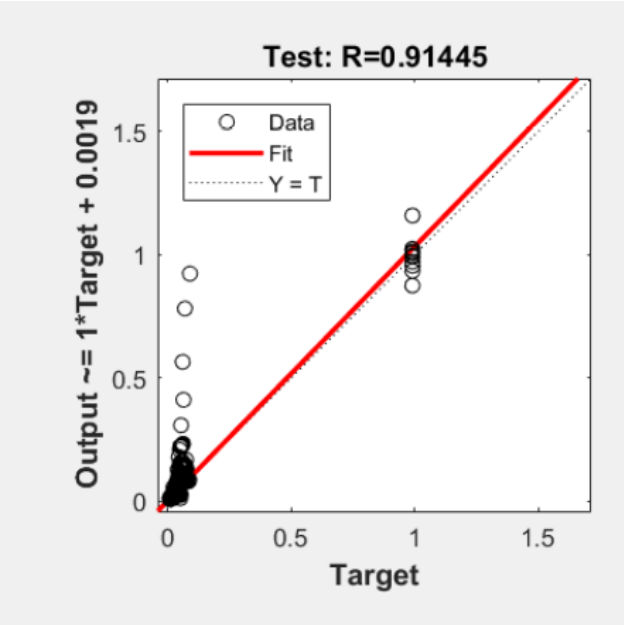


Figura 7.31 Curva de regresión que da la red para el caso de la salida del daño, un entrenamiento con 1000 casos y unas condiciones de: 2 capas con 20 y 10 neuronas y la función trainbr.

– Condiciones de entrenamiento: 3 capas con 40, 10 y 10 neuronas y la función trainlm (ver figura 7.35).

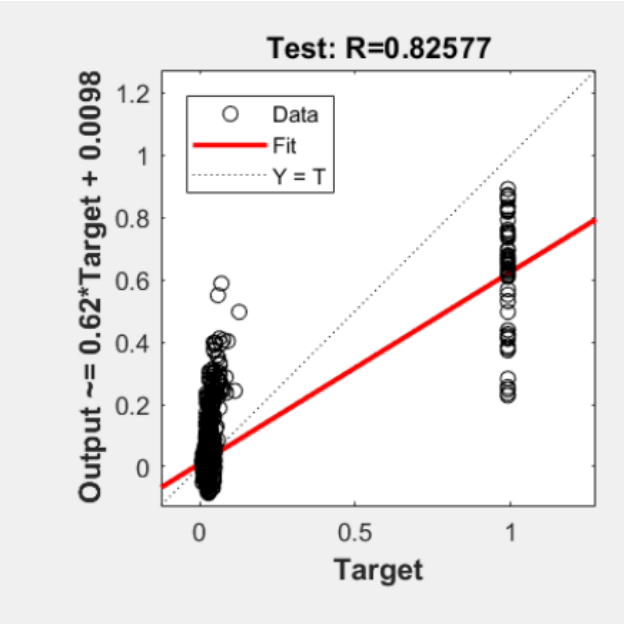


Figura 7.32 Curva de regresión que da la red para el caso de la salida del daño, un entrenamiento con 2000 casos y unas condiciones de: 1 capa con 20 neuronas y la función trainlm.

Viendo el coeficiente R en las curvas de regresión, se obtiene que la mejor red es el caso de 3 capas con 40, 20 y 10 neuronas y la función trainscg, correspondiente a la figura 7.34.

Además, se puede deducir que la red que predice mejor el daño es la de 1000, ya que cuando el resultado debe estar en torno a 1, no se va a valores pequeños, lo cual podría ser peligroso al estar

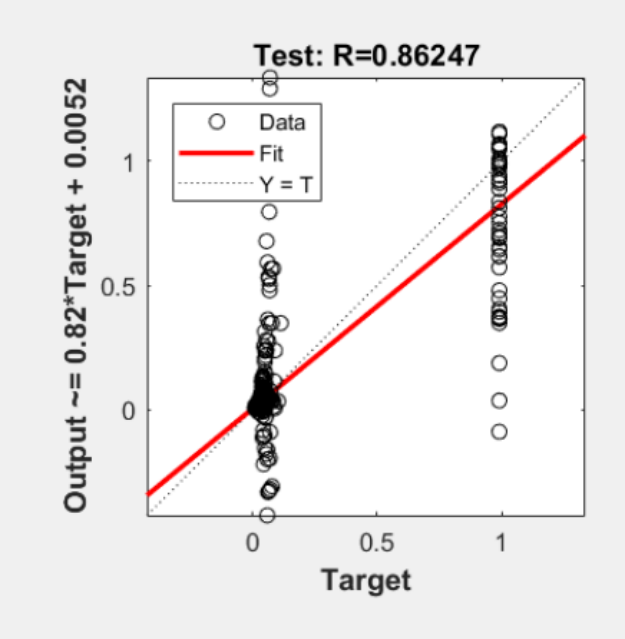


Figura 7.33 Curva de regresión que da la red para el caso de la salida del daño, un entrenamiento con 2000 casos y unas condiciones de: 2 capas con 20 y 10 neuronas y la función trainlm.

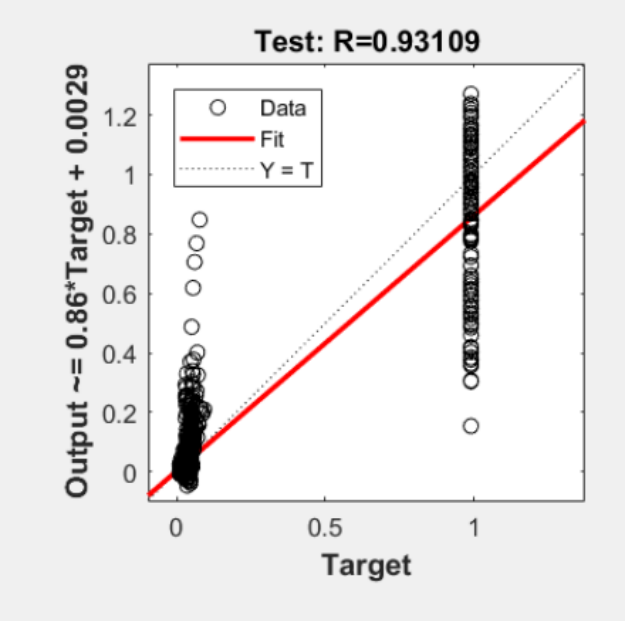


Figura 7.34 Curva de regresión que da la red para el caso de la salida del daño, un entrenamiento con 2000 casos y unas condiciones de: 3 capas con 40, 20 y 10 neuronas y la función trainscg.

diciendo que el hueso no se rompe cuando en la realidad sí pasaría.
Por último, se van a comparar las mejores funciones obtenidas para cada número de casos mediante unos programas en Matlab llamados *error_red_ash/BDG/damage.m* y *cuantif_error_ash/BDG/damage.m*, (se pueden ver en el anexo), que han dibujado en una sola gráfica las medias de los errores de cada salida en los distintos instantes de tiempo, es decir, para cada dato de testeo, se han tomado todos los que corresponden a un instante de tiempo y se ha realizado el promedio. Viendo la gráfica 7.36, es difícil ver cuál es la que predice mejor, por eso,

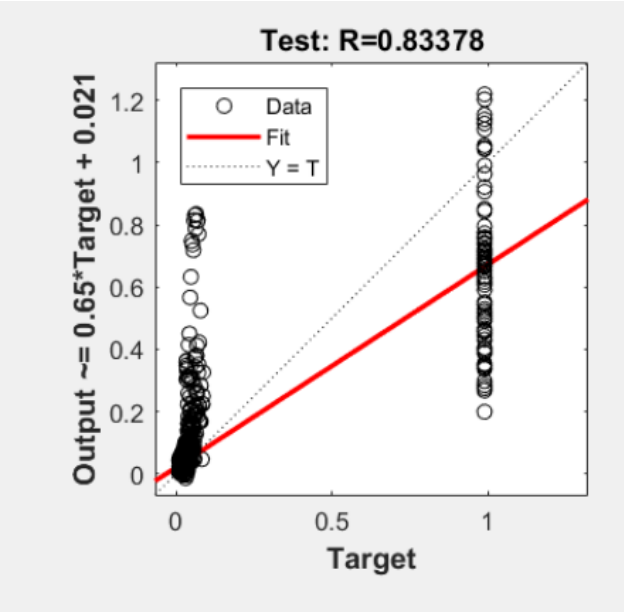


Figura 7.35 Curva de regresión que da la red para el caso de la salida del daño, un entrenamiento con 2000 casos y unas condiciones de: 3 capas con 40, 10 y 10 neuronas y la función trainlm.

se llevó a cabo un cálculo adicional en el que se hacía un promedio de las medias de los errores obtenidos, como se aprecia en el anexo, y se extrajo que la que menor error tenía era la red de 2000 casos, al ser de 0,0028, mientras que los otros fueron de 0,0030. Las siguientes expresiones resumen los cálculos realizados:

$$Error_{medio\ total} = -\frac{\sum_{i=1}^{k} Salida(r)_i}{k} \tag{7.1}$$

$$Error_{medio\ total} = -\frac{\sum_{i=1}^{n} Errormed_i}{n} \tag{7.2}$$

La ecuación 7.1 es la expresión que refleja el promedio de una de las salidas (BDG, fracción de ceniza o daño) en un instante determinado "r", y donde k es el número de datos de testeo. Mientras que la ecuación 7.2, refleja los cálculos adicionales que han sido necesarios para el caso de la fracción de ceniza, y n es el número de instantes en que se evalúa la fracción de ceniza.
En el caso del BDG, se ve claramente en la figura 7.37 que también es la de 2000, mientras que para el daño con 2000 casos está sobreentrenada y la que mejor predice es la de 1000 al estar los errores más cercanos a 0 a lo largo del tiempo, como se aprecia en la figura 7.38.

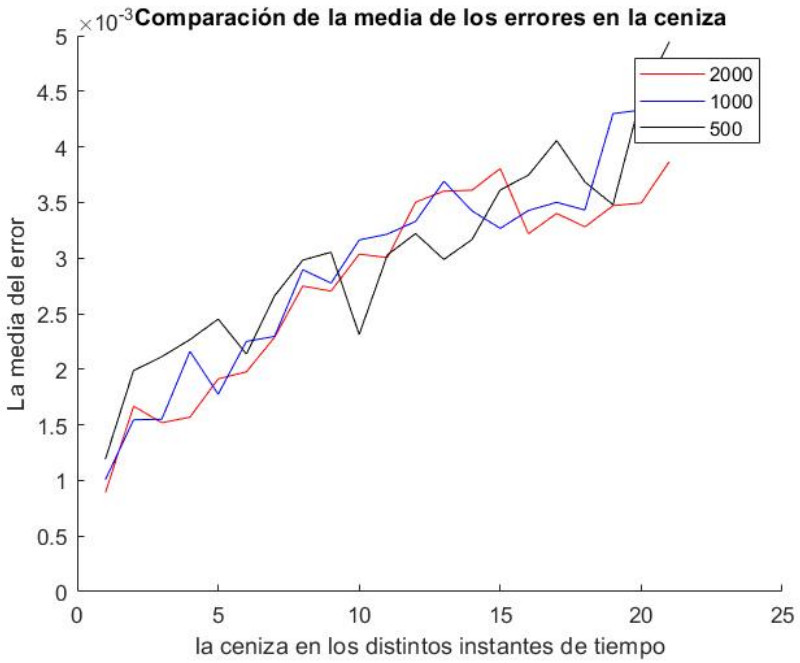


Figura 7.36 Curva que representa la media de los errores de la fracción de ceniza obtenido por las redes neuronales con respecto al tiempo.

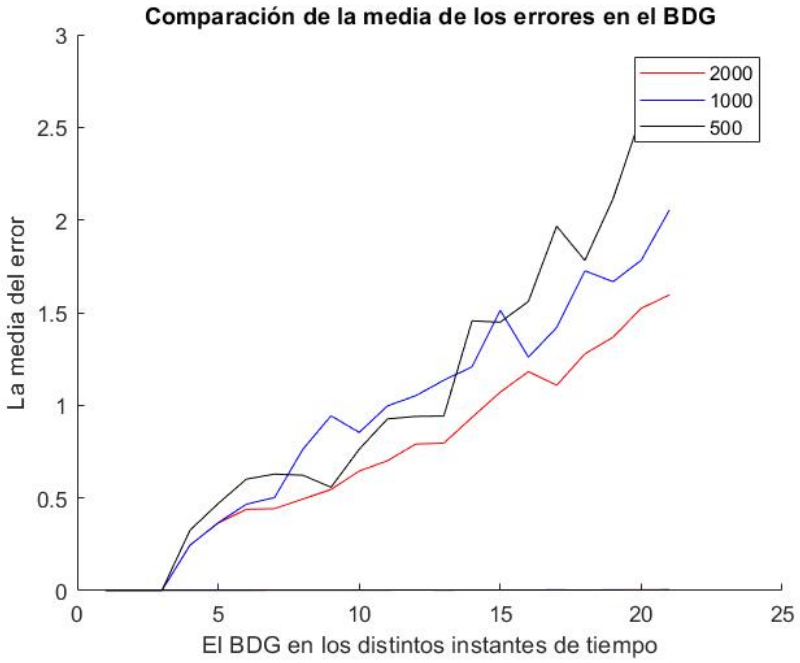


Figura 7.37 Curva que representa la media de los errores del BDG obtenido por las redes neuronales con respecto al tiempo.

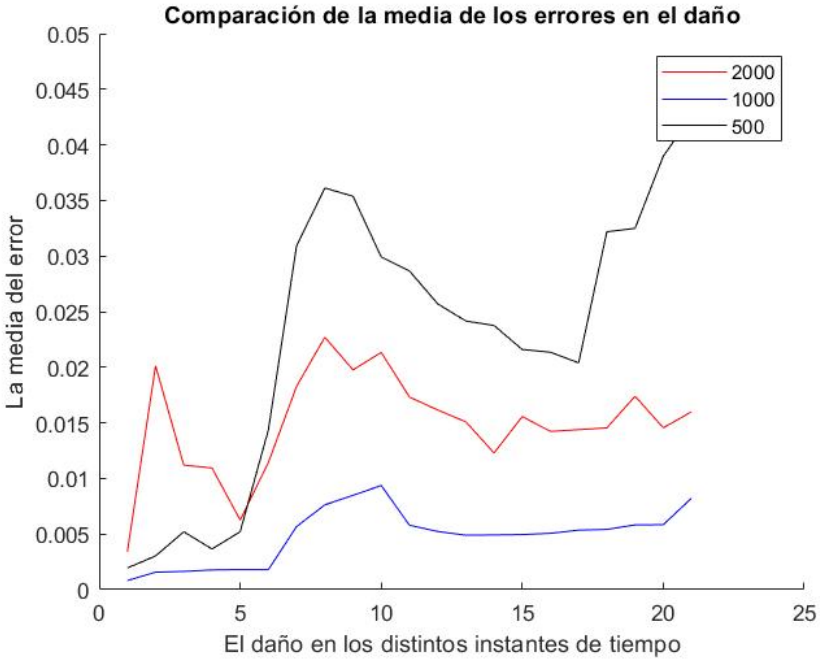


Figura 7.38 Curva que representa la media de los errores del daño obtenido por las redes neuronales con respecto al tiempo.

7.3 Algoritmo genético

Los resultados que se obtuvieron en el algoritmo genético y los estudios previos que se realizaron para decidir qué tipo de herramienta de optimización se usaría se ven en esta sección.

7.3.1 Estudios previos

En primer lugar, en los estudios previos a la realización de un algoritmo genético, se obtuvo que la superficie que se obtenía al representar T2 y T4 frente al BDG, tenía más de un mínimo local, por lo que la mejor herramienta de optimalidad en esta situación era un algoritmo genético (ver figura 7.39) y además, no fue necesario realizar más pruebas, ya que si con las primeras se obtenían más de un mínimo local, no sería adecuado un algoritmo de gradiente.

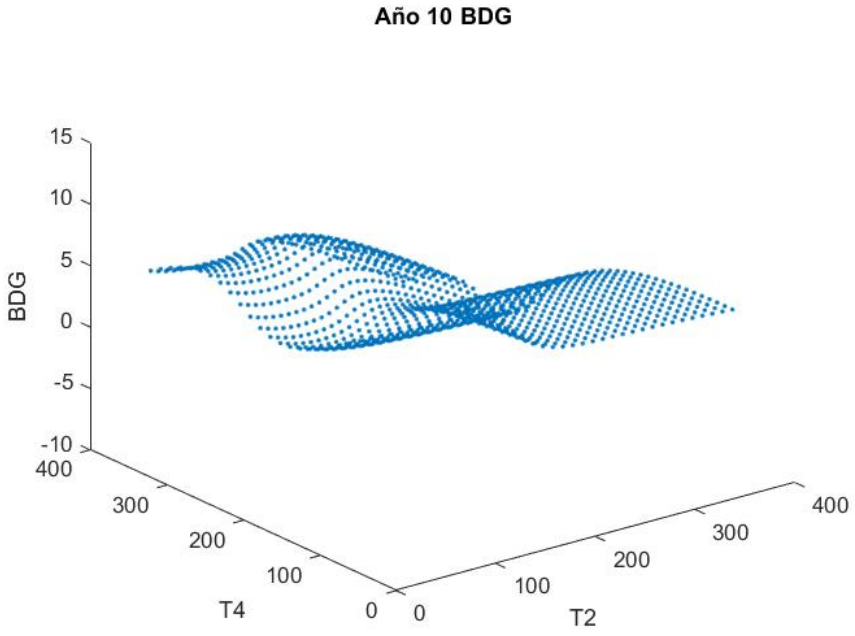


Figura 7.39 Superficie que representa el valor del BDG en el año 10 de enfermedad, variando T2 y T4, siendo T1, T3 y T5 constantes. Se aprecian varios mínimos.

7.3.2 Aplicación del algoritmo genético

Se ha querido estudiar cómo varían los parámetros del tratamiento, que son las variables optimizadas, y las salidas BDG, daño y fracción de ceniza que se obtienen para distintas combinaciones de los datos del paciente. Para ello, se han realizado una serie de figuras en las que se comparan estos parámetros.

Estudio de la variación de k_{res}

En primer lugar, se experimentó con el caso de variación de k_{res} manteniendo el resto de parámetros constantes, en los que la variación de la actividad física $\Delta\sigma = 0{,}1MPa$, el tiempo de enfermedad = 3 años y $P^{PMO}_{RANKL} = 2000pM/dia$ (ver figuras 7.40 y 7.41 para la evolución de los tratamientos y las figuras 7.42, 7.43 y 7.44 la evolución de las distintas salidas).

Como se puede observar, en el caso de la influencia en el tratamiento, en cuanto a la dosis de los mismos, a medida que k_{res} va creciendo, hasta el valor de 150 %, es necesaria la inyección de menor cantidad de Denosumab para alcanzar el tratamiento óptimo. Sin embargo, cuando supera ese valor, cada vez es necesario más tratamiento. Por otro lado, las frecuencias de suministro de los

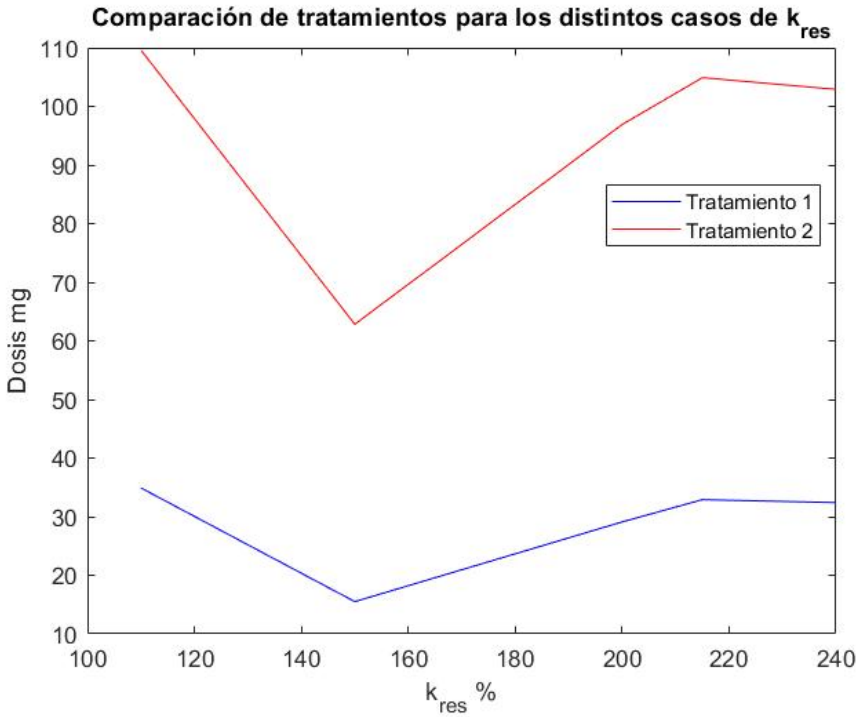


Figura 7.40 Curva de las dosis de los tratamientos (k_{res}).

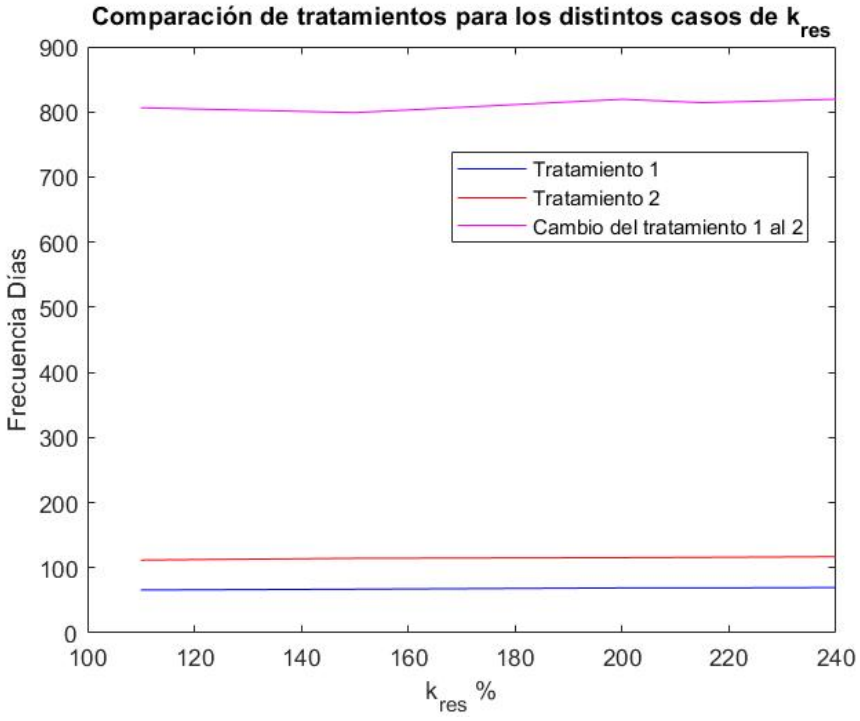


Figura 7.41 Curva de las frecuencias de los tratamientos (k_{res}).

tratamiento a pacientes no están apenas influenciadas por el k_{res}, ya que se mantienen relativamente constantes.

La fracción de ceniza, al igual que el BDG no están influenciados por la variación de k_{res}, sin embargo, se puede apreciar que a mayor valor de esta variable, hay más mineralización y ganancia ósea. El daño si está más afectado por esta variable. De forma llamativa se aprecia que cuando el

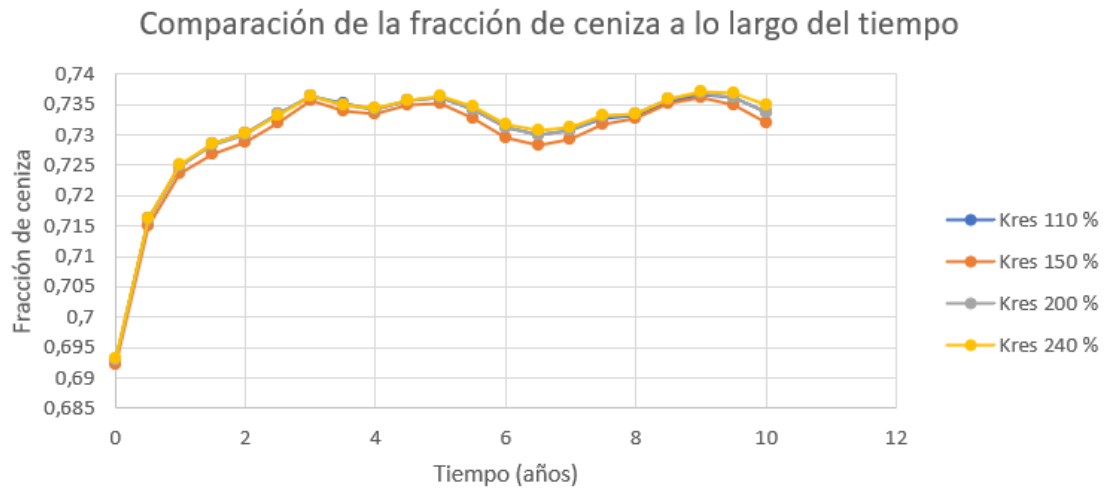


Figura 7.42 Curvas de la fracción de ceniza (k_{res}).

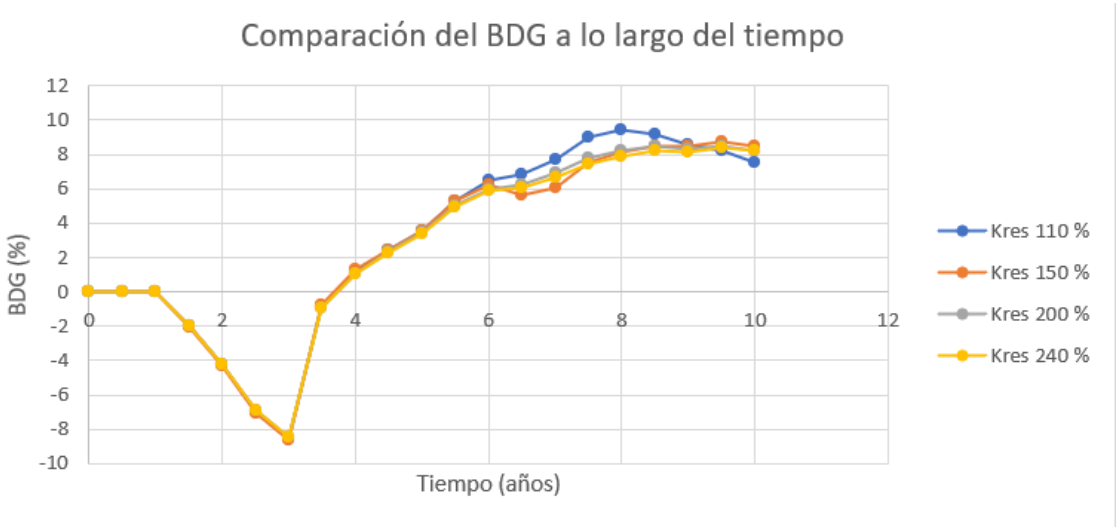


Figura 7.43 Curvas del BDG (k_{res}).

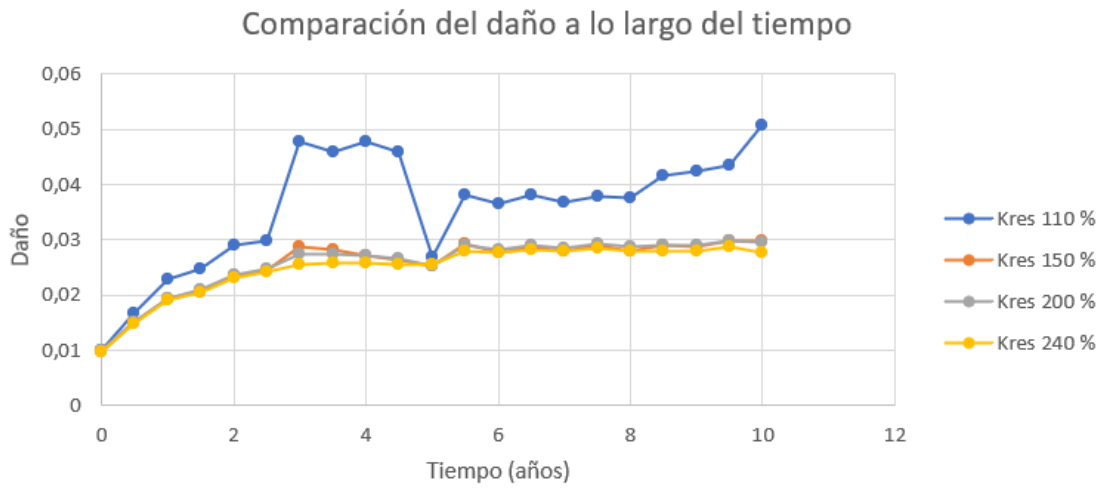


Figura 7.44 Curvas del daño (k_{res}).

k_{res} es pequeño, produce mayor daño, pudiendo ser tan elevado que existe el riesgo de rotura. Esto puede ser debido a que como el k_{res} es la tasa relativa de reabsorción ósea normalizada con respecto a la reabsorción ósea normal, a menor valor de ésta, se produce menos reabsorción, impidiendo la regeneración del hueso, lo que implica que éste se quede con mayor daño y mineralización.

Estudio de la variación de $\Delta\sigma$

A continuación, se experimentó variando $\Delta\sigma$ manteniendo el resto de parámetros constantes, en los que $k_{res} = 200\,\%$, el tiempo de enfermedad = 3 años y $P_{RANKL}^{PMO} = 2000 pM/dia$ (ver figuras 7.45 y 7.46 para la evolución de los tratamientos y las figuras 7.47, 7.48 y 7.49 la evolución de las distintas salidas).

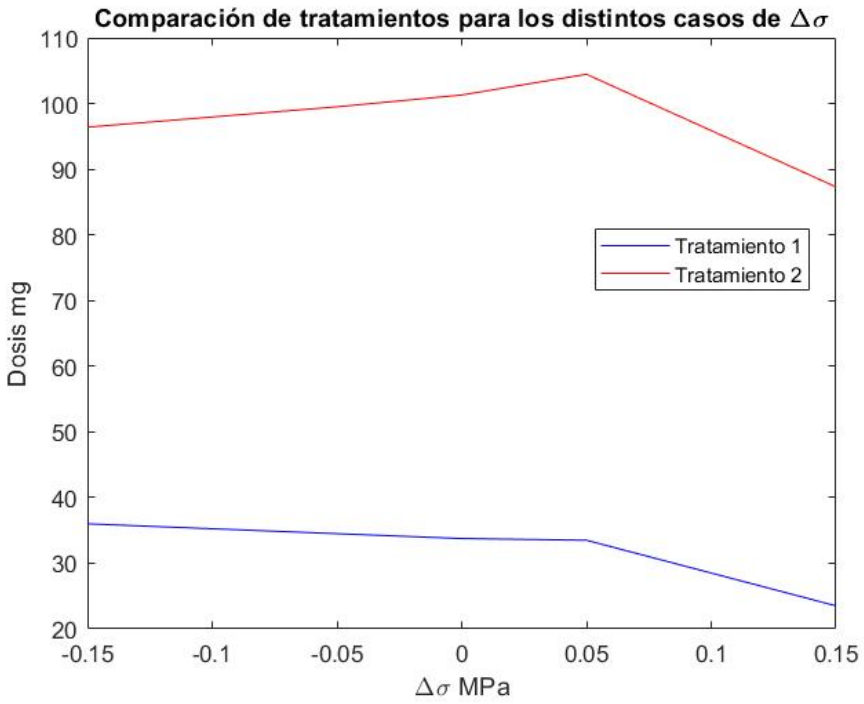


Figura 7.45 Curva de las dosis de los tratamientos ($\Delta\sigma$).

Como se aprecia en las figuras correspondientes a la variación de la actividad física, se observa que a mayor cantidad de actividad, en el tratamiento 1 influye en una menor necesidad de inyección de Denosumab, mientras que en el tratamiento 2, cuando la actividad es insuficiente (menor que la de equilibrio), a medida que aumenta ésta, hace falta mayor dosis, pero una vez que ha superado un cierto valor, de 0,05 MPa, la necesidad disminuye. En cuanto a las frecuencias de los tratamientos, esta variable apenas influye.

Por otro lado, cuanto mayor es la actividad física, menor es la fracción de ceniza resultante tras el tratamiento, además de que hay mayor incremento de densidad ósea. Esto está influenciado porque cuando se realiza ejercicio, la remodelación ósea se activa, tal y como se vio en el capítulo 2, al producir una variación en las tensiones que reciben los osteocitos. Por esto, como los osteoblastos en la fase de formación depositan osteoide, que no tiene fase orgánica, la mineralización es menor. Esto también explica por qué cuando la actividad física es casi nula, la mineralización es alta. Sin embargo, si la actividad física crece demasiado, esto puede dar lugar a un estado peligroso en el que el daño de valores muy cercanos a 1, provocando fracturas óseas.

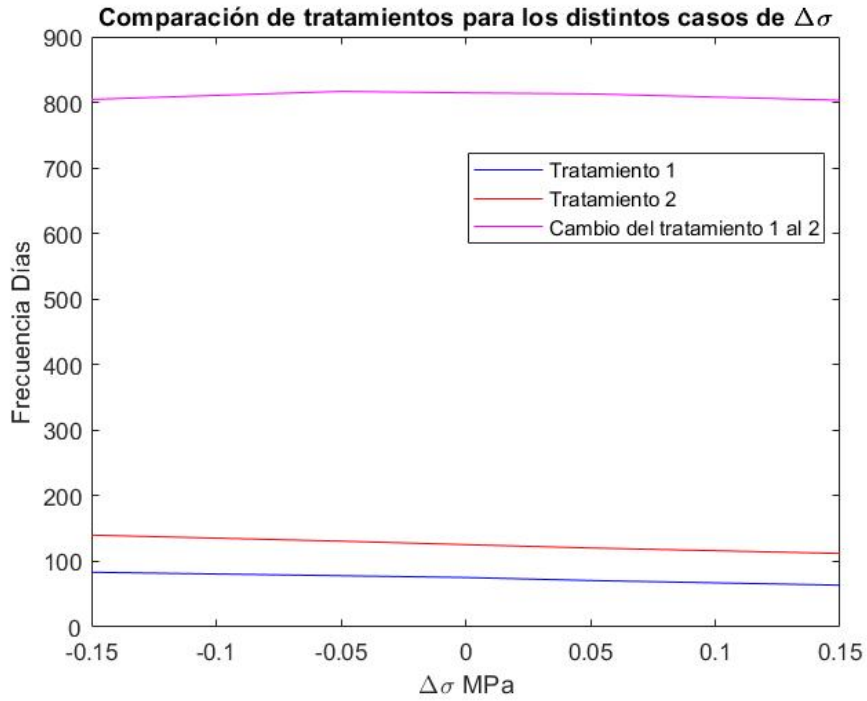


Figura 7.46 Curva de las frecuencias de los tratamientos ($\Delta\sigma$).

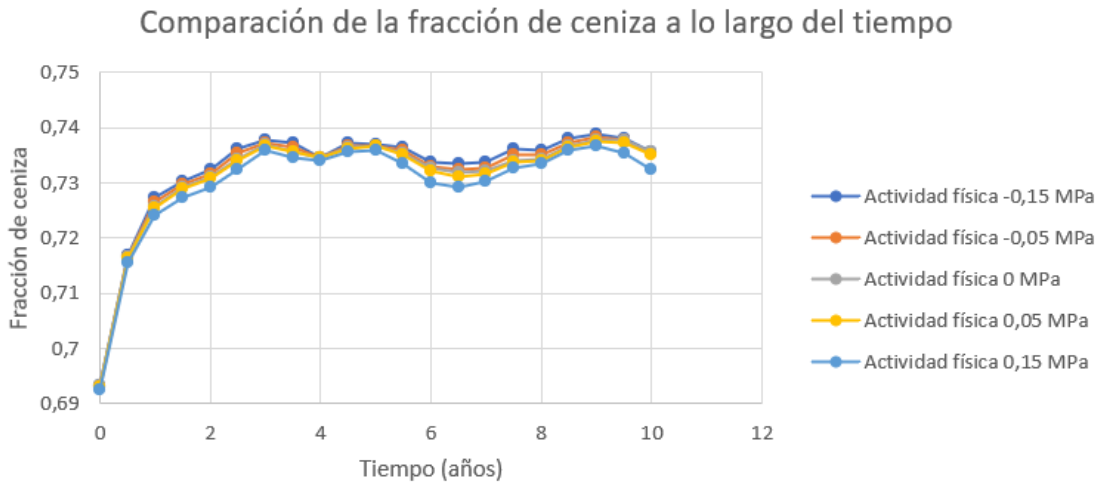


Figura 7.47 Curvas de la fracción de ceniza ($\Delta\sigma$).

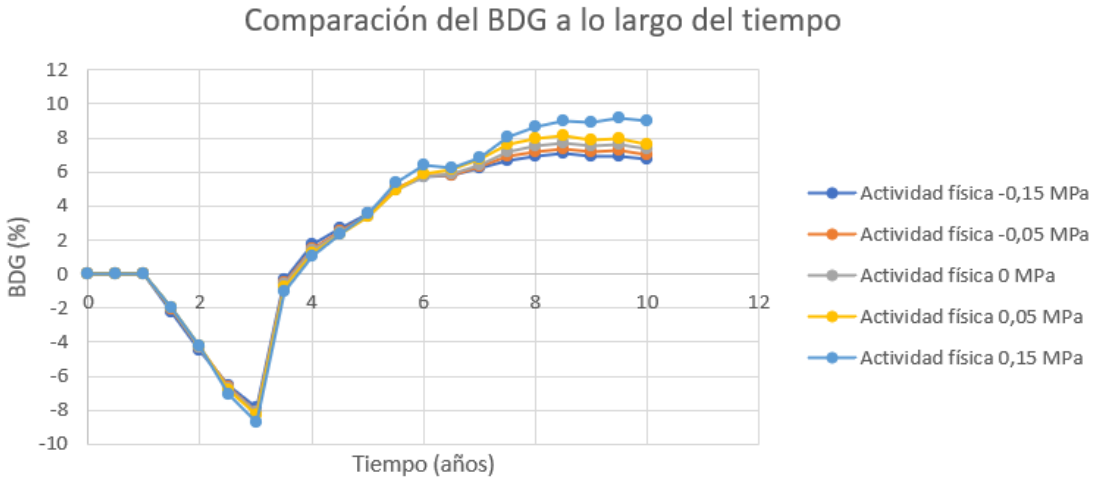


Figura 7.48 Curvas del BDG ($\Delta\sigma$).

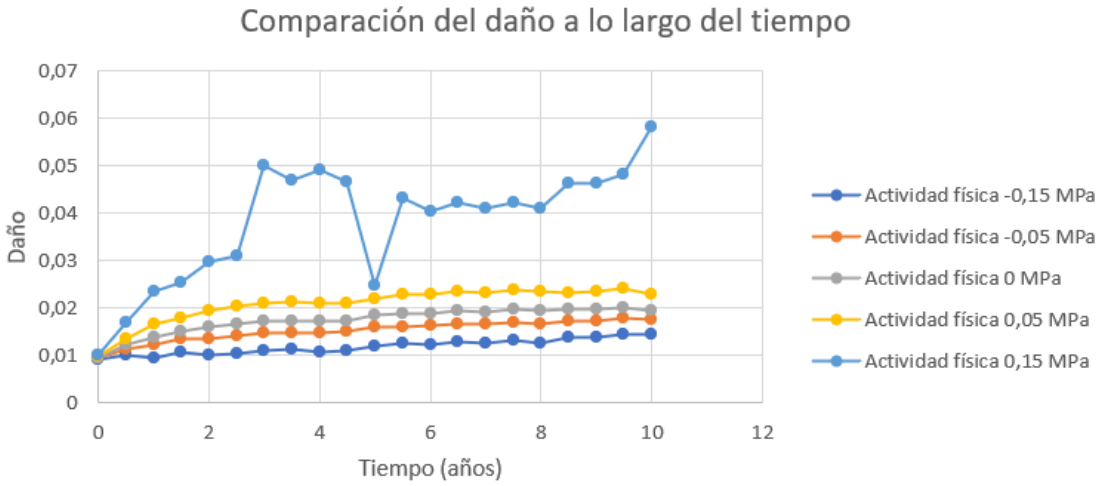


Figura 7.49 Curvas del daño ($\Delta\sigma$).

Estudio de la variación del tiempo de enfermedad

Posteriormente, se experimentó variando el tiempo de enfermedad manteniendo el resto de parámetros constantes, en los que $k_{res} = 200\,\%$, $\Delta\sigma = 0{,}1MPa$ y $P_{RANKL}^{PMO} = 2000pM/dia$ (ver figuras 7.50 y 7.51 para la evolución de los tratamientos y las figuras 7.52, 7.53 y 7.54 la evolución de las distintas salidas).

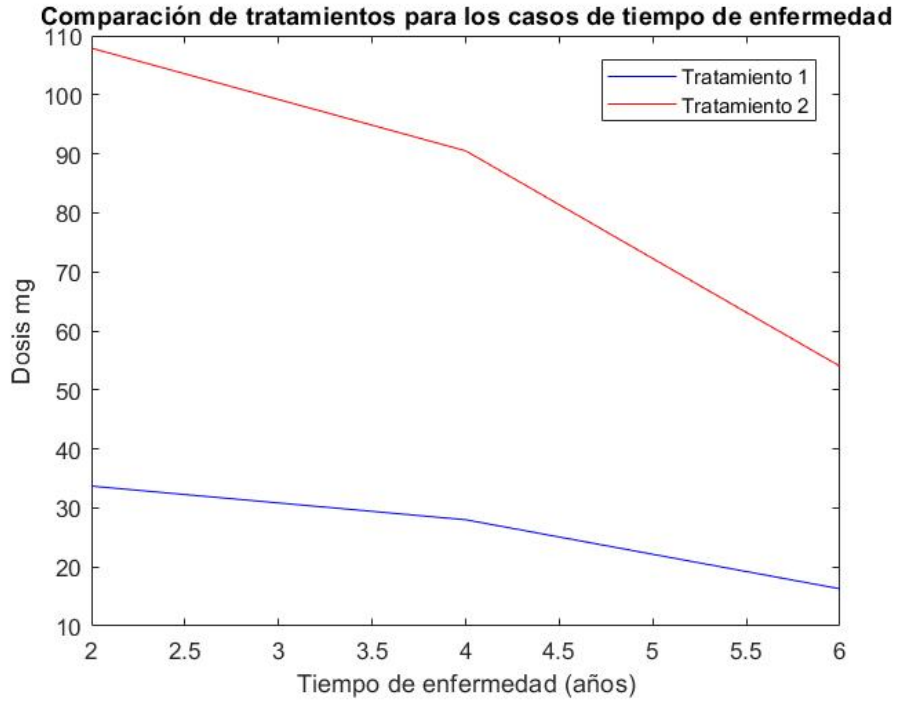


Figura 7.50 Curva de las dosis de los tratamientos (tiempo de enfermedad).

Viendo las figuras, se deduce que a medida que el tiempo de enfermedad previa al tratamiento crece, la cantidad de tratamiento necesaria disminuye. En cuanto a las frecuencias de los tratamientos, se aprecia que el tratamiento 1 es suministrado con menos frecuencia que el 2, hasta el año 5, sin embargo, a partir de ahí ocurre lo contrario.
La evolución de la curva de la fracción de ceniza informa de que a mayor tiempo de enfermedad, menor es la mineralización y el daño a largo plazo tras el tratamiento óptimo suministrado. Sin embargo, la ganancia ósea es significativamente mayor en el caso de un paciente tratado con menos años de enfermedad previa.

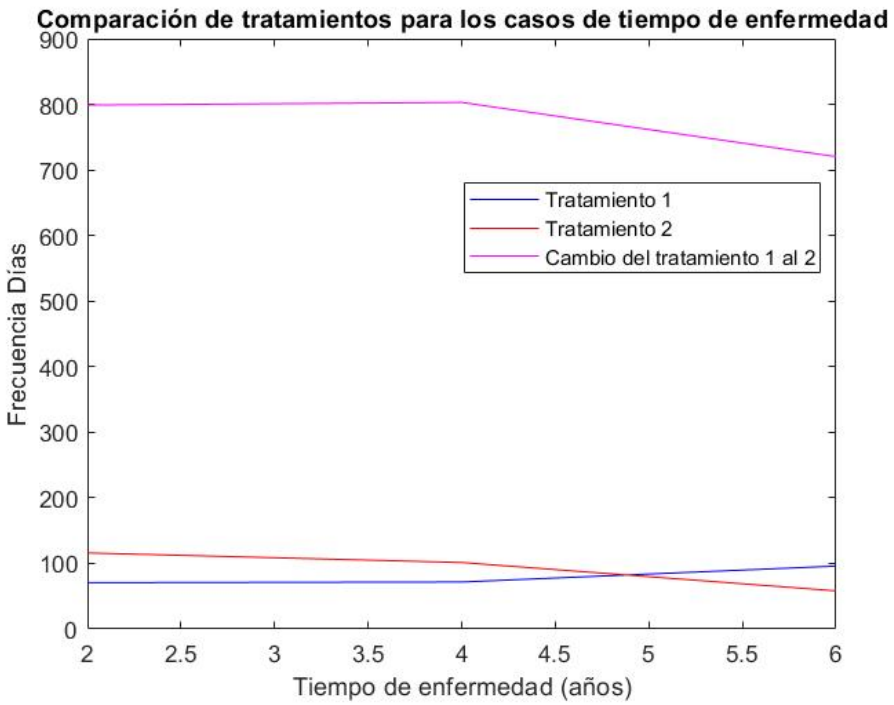


Figura 7.51 Curva de las frecuencias de los tratamientos (tiempo de enfermedad).

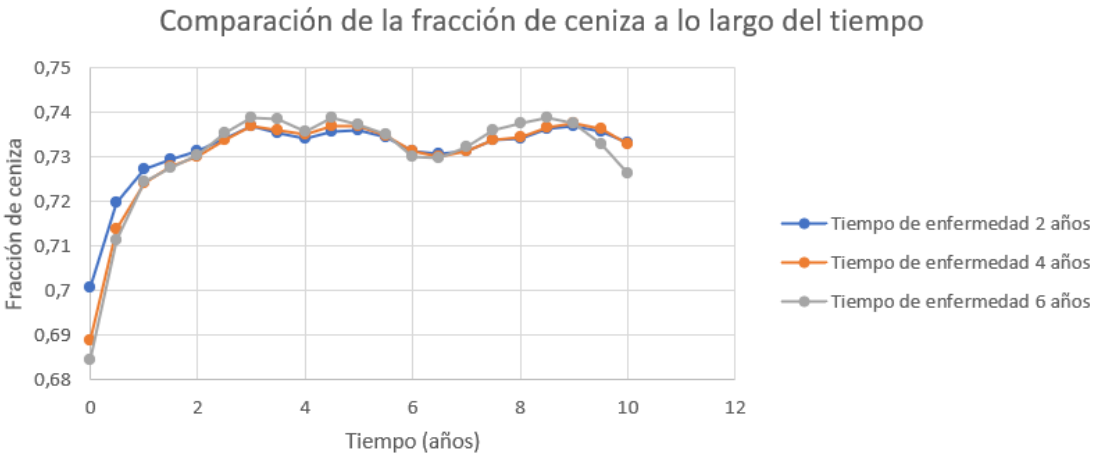


Figura 7.52 Curvas de la fracción de ceniza (tiempo de enfermedad).

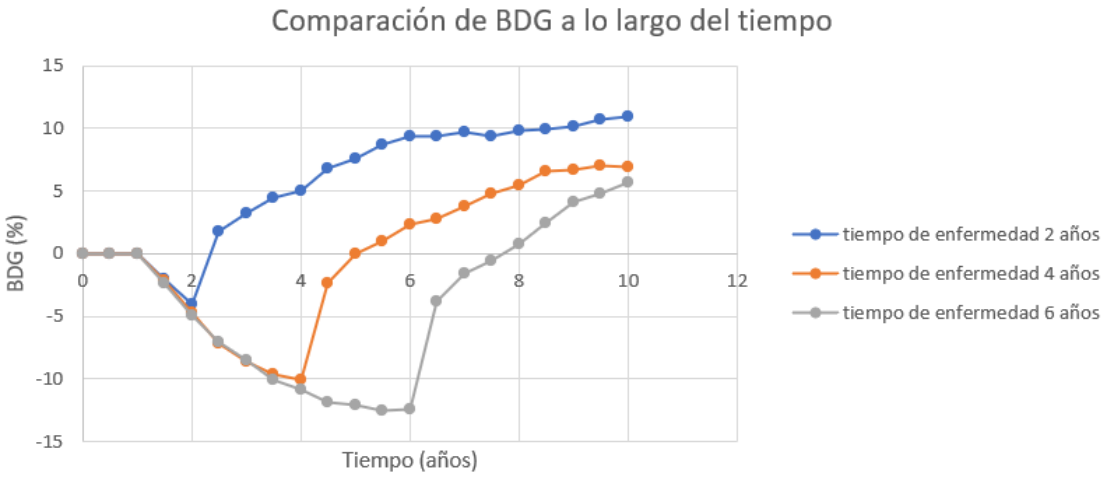


Figura 7.53 Curvas del BDG (tiempo de enfermedad).

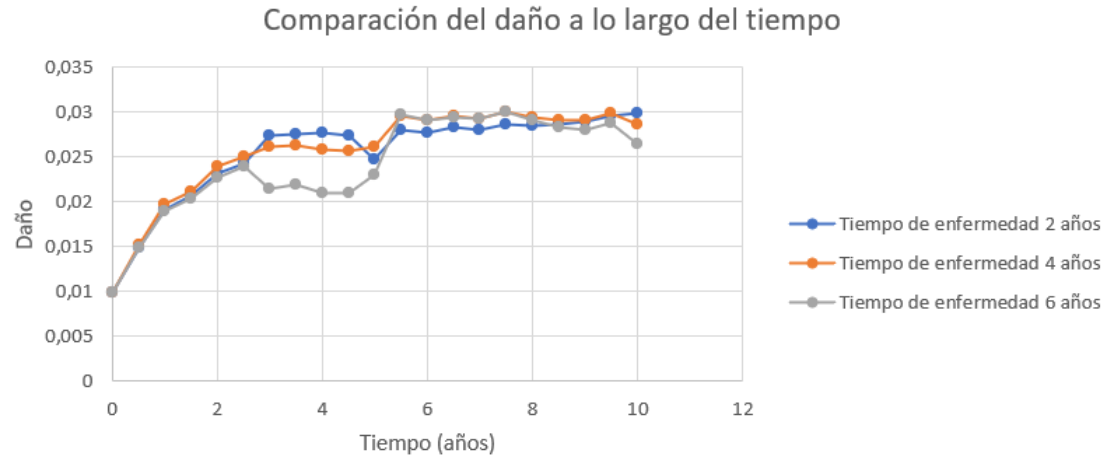


Figura 7.54 Curvas del daño (tiempo de enfermedad).

Estudio de la variación de P_{RANKL}^{PMO}

Por último, se experimentó variando P_{RANKL}^{PMO} manteniendo el resto de parámetros constantes, en los que $k_{res} = 200\,\%$, $\Delta\sigma = 0{,}1MPa$ y el tiempo de enfermedad = 3 años (ver figuras 7.55 y 7.56 para la evolución de los tratamientos y las figuras 7.57, 7.58 y 7.59 la evolución de las distintas salidas).

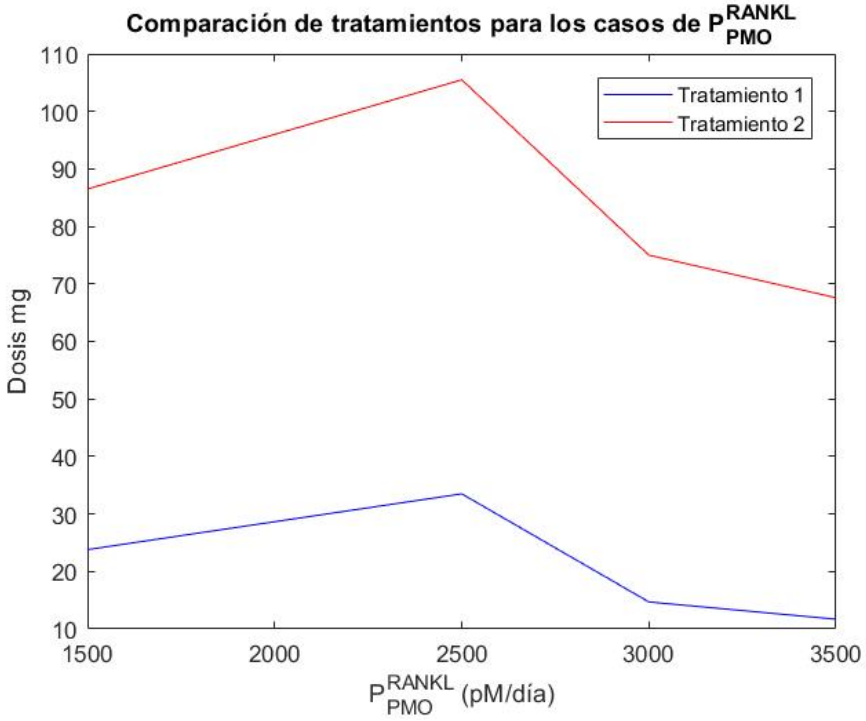


Figura 7.55 Curva de las dosis de los tratamientos (P_{RANKL}^{PMO}).

En este caso, con el aumento de P_{RANKL}^{PMO} hasta el valor de 2500 pM/día aproximadamente, se produce un aumento en la necesidad de dosis del tratamiento óptimo. Sin embargo, cuando los valores de esta variable supera este valor, la necesidad de mayor tratamiento disminuye.
Por otro lado, cuando el valor de P_{RANKL}^{PMO} es mayor, los resultados que se obtienen son mejores, es decir, la mineralización y el daño son pequeños, mientras que el porcentaje de ganancia ósea aumenta. Esto es debido a que esta variable se corresponde con la tasa de producción de RANKL correspondiente con la enfermedad de osteoporosis postmenopáusica, que implica que a mayor valor de esto, más cantidad de RANKL se produce, aumentando la diferenciación de los osteoclastos.

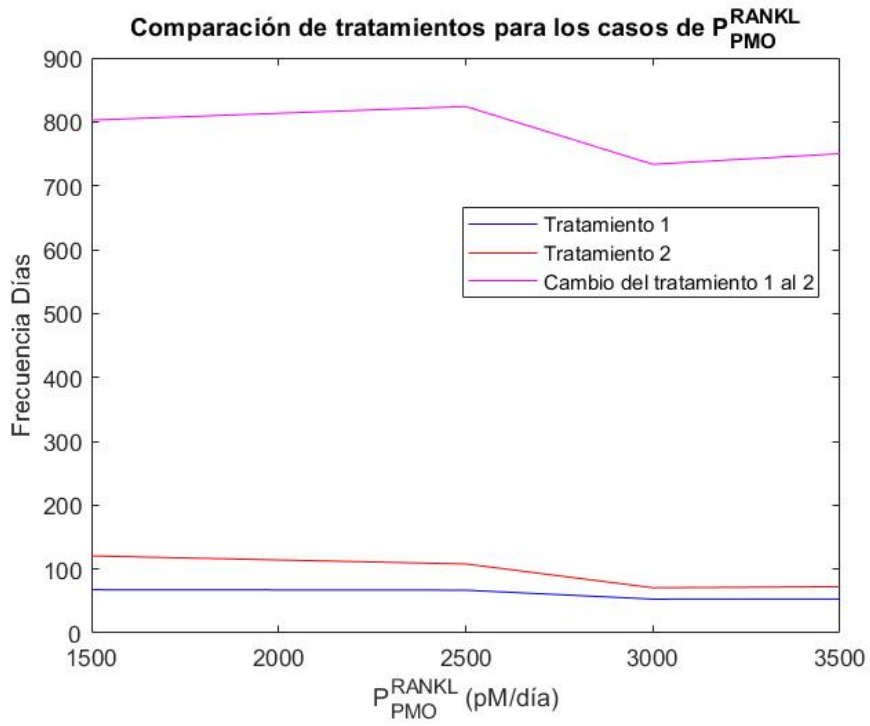


Figura 7.56 Curva de las frecuencias de los tratamientos (P^{PMO}_{RANKL}).

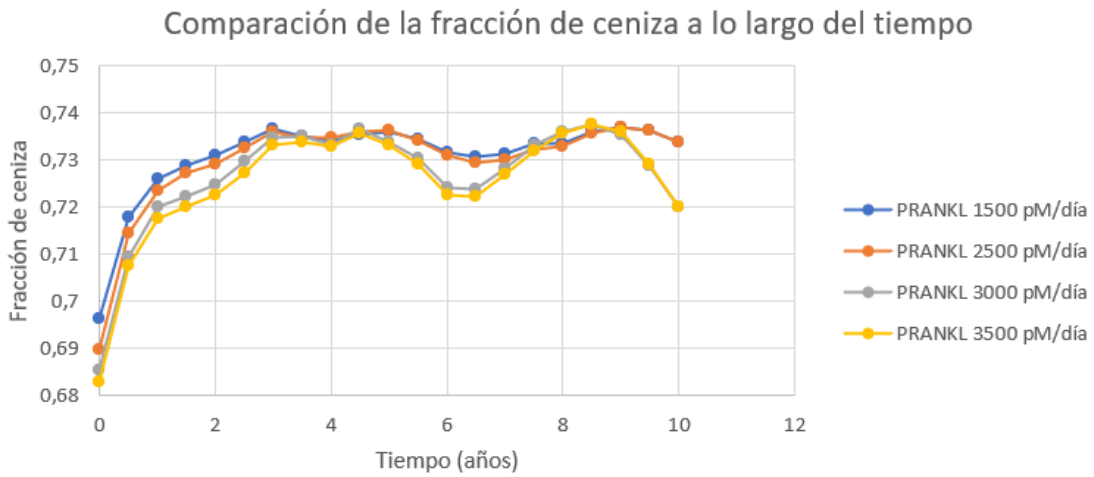


Figura 7.57 Curvas de la fracción de ceniza (P^{PMO}_{RANKL}).

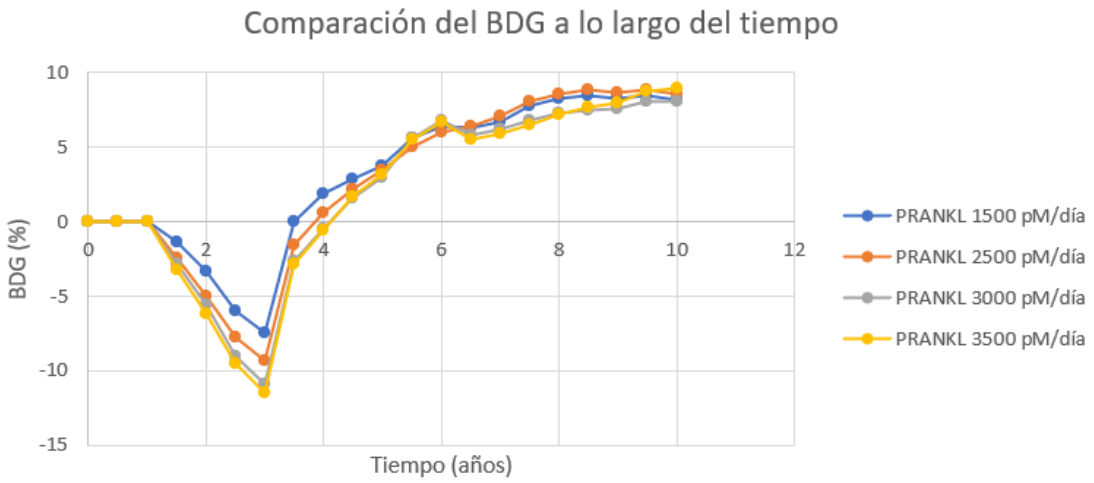


Figura 7.58 Curvas del BDG (P^{PMO}_{RANKL}).

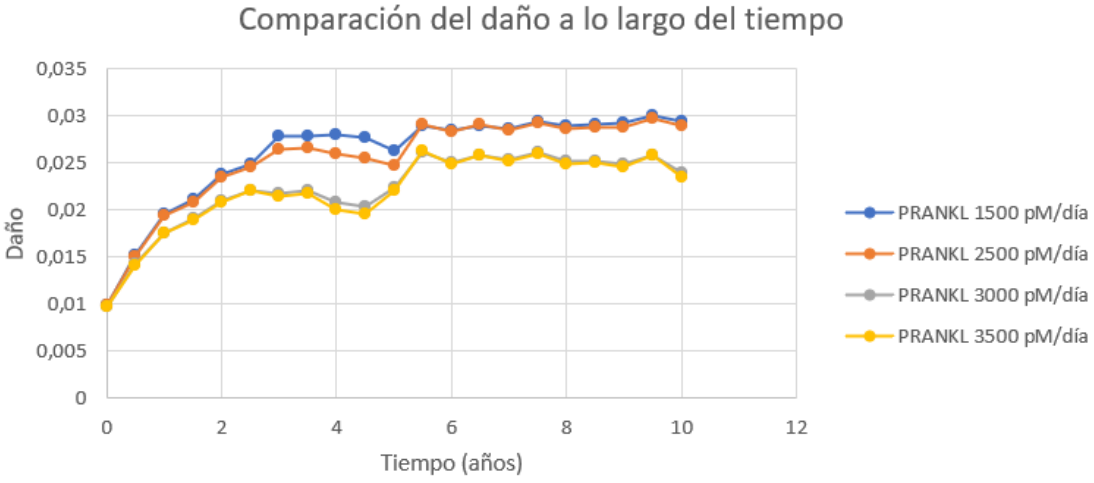


Figura 7.59 Curvas del daño (P^{PMO}_{RANKL}).

8 Manual del programador

El objetivo de este capítulo es entender cómo se ha programado la rutina y qué significan las líneas del código de cada uno de los archivos utilizados para llevar a cabo el programa, como un breve manual de programador, además de cuál es el sentido global del programa y de las funciones. La rutina está dividida en 3 grandes partes: la aplicación del modelo matemático, las redes neuronales y el algoritmo de optimización genético.

8.1 Aplicación del modelo matemático

El objetivo de los siguientes archivos es obtener las matrices que se usarán más adelante para entrenar la red neuronal y obtener la función correspondiente mediante la utilización del modelo matemático explicado anteriormente.

8.1.1 Archivo funcion_principal

El archivo de la función principal es un blucle "for" que va construyendo las matrices de fila en fila. En primer lugar, se pone un comando que es *rng('shuffle')* que ayuda a que el programa tome los valores más aleatorios, ya que Matlab toma valores seudoaleatorios y cuando se empieza de nuevo la sesión, siempre empieza por el mismo. A continuación, mediante la función *randi* se toman valores de los tratamientos (T1-T5) y las constantes que serían propias del paciente en los intervalos adecuados (k_{res}, $\Delta\sigma$, el tiempo que lleva de enfermedad y el P^{PMO}_{RANKL} del paciente).
Posteriormente se llama a la función *run_bonemodel12* que es la que aplica el modelo matemático y devuelve las tres salidas que se quieren analizar, BDG, fracción de ceniza y el daño, durante los 10 años de tratamiento cada 6 meses. Para montar las matrices, se cargan los resultados y se hace un bucle "for" que va cogiendo los valores correspondientes a los intervalos de 6 meses, por eso se multiplica por 720 (Se consideran años de 360 días para facilitar la tarea).
Por último, se guardan las matrices formadas.

Función run_bonemodel12

Esta función tiene de entrada los parámetros característicos del tratamiento y las constantes propias del paciente.
En primer lugar, se declaran una serie de parámetros constantes y se carga el tratamiento 02 (viene en el código de Matlab como predeterminado), a elegir por el usuario. Posteriormente llama a la función *fu_modelparameters12*, que tiene los parámetros del modelo, y éstos se actualizan mediante los datos de entrada de la función *run_bonemodel12*. A continuación, se determinan los parámetros que definen el intervalo de dosis del tratamiento y enfermedad.
Los valores iniciales de las variables del modelo se calculan mediante otra función llamada *Main_homeost_v2*, que parte de unos valores iniciales predeterminados y corre la simulación hasta que

alcanza la homeostasis, es decir, hasta que se consigue que los valores se mantengan estables y se actualizan los parámetros. De aquí en adelante, se lleva a cabo el modelo matemático explicado en el capítulo 3. Para ello se han necesitado una serie de funciones extra como son:

- *fu_updateVFPREV12*: esta función actualiza la cola VFPREV con la cantidad de tejido formado (DVF) y reabsorbido (DVR) cada día.
- *fu_calmineral12*: esta función proporciona la fracción de volumen mineral de un pedazo de hueso formado hace t días.
- *fu_loadcase12*: esta función modifica los parámetros del modelo en función del estado de carga/caso simulado.
- *fu_bonecelldynamics12*: esta función resuelve las ecuaciones y está compuesta de otras funciones:
  - *fu_PTH12*: esta función calcula los valores de la función activadora / represora PTH.
  - *fu_TGFbe_d12*: esta función calcula la concentración de $TGF\beta$ y luego los valores de la función activadora / represora.
  - *fu_mechanics12*: esta función calcula las cantidades de la regulación mecánica.
  - *fu_RANKL12*: esta función calcula los valores de la función activadora / represora RANKL
  - *specific_surface12*: esta función calcula la superficie específica S_V.
  - *fu_dano12*: esta función evalúa el daño.
- *denosumab12*: esta función sirve para calcular la concentración de Denosumab.

Para terminar, se calcula el BDG. Para ello, se definen los instantes de tiempo en que se van a guardar los resultados mediante un bucle "for". Después se va calculando la ganancia ósea en porcentaje con respecto al primer año de tratamiento. Finalmente, se guardan todos los resultados.

Función Main_homeost_v2

Esta función tiene de entrada los valores del tratamiento y el k_{res} del paciente. El objetivo de esta función es calcular los valores iniciales de las variables dinámicas del modelo tras alcanzar la homeostasis, que es una propiedad de los organismos que consiste en su capacidad de mantener una condición interna estable compensando los cambios en su entorno, con una tansión constante en este caso de 0,5 MPa.
Para conseguir esos valores, en primer lugar, hay que establecer unos que vienen predeterminados y, mediante unas funciones auxiliares parecidas a la que se usaron en la función principal, *fu_modelparameters12homeost*, que contiene los parámetros de la homeostasis y *run_bonemodel12_homeost*, al que solo se le pasan los valores del tratamiento, los parámetros de la homeostasis y el volumen de referencia inicial, se van consiguiendo unos valores de las variables dinámicas que podrían ser los nuevos iniciales. Estas funciones son utilizadas dentro de un algoritmo de convergencia mediante un bucle "while", que para de iterar cuando los valores iniciales predeterminados o los de la última simulación realizada, y los obtenidos mediante la simulación actual convergen en una tolerancia menor de la indicada. Por último, se guardan los resultados.

Función injection12

Esta función tiene como entradas las características del tratamiento (T1-T5), el peso del paciente y el tiempo de inyección. Se pretende crear un vector que contenga las inyecciones que correspondan durante los tiempos determinados, y se hace de forma automática mediante un bucle "while", que tiene un contador y va colocando de forma ordenada los parámetros.

8.2 Redes neuronales

El objetivo de este programa es obtener las funciones que vayan a utilizarse en la optimización. Para ello, se ha hecho lo más general posible, de forma que el usuario vaya introduciendo las características y matrices tanto de entrada como de salida que vayan a usarse para entrenar la red o validarla.

8.2.1 Archivo red_neuronal_general.m

En primer lugar, el programa reclama al usuario que ponga el número de capas y la cantidad de neuronas por capa, en forma de vector, por ejemplo, [10,20], esta red tendría 2 capas, en la primera, 10 neuronas y en la segunda 20. A continuación, se le pide al usuario que ponga la función de entrenamiento que se quiera usar (trainlm, trainbr o trainscg). Posteriormente se cargan las matrices de entrada y salida que se obtuvieron en el anterior programa mediante el comando *load*, y se crea un código para que el usuario escoja las que quiere entrenar (se han cargado todas las matrices).
En las siguientes líneas se crea la red neuronal en sí. En primer lugar, se le pone un nombre, después, mediante la función *feedforwardnet*, se ponen las capas y la función de entrenamiento que ya escogió el usuario. Con la función *configure* se le pasa a la red las entradas y las salidas. Con la función *view* aparece un cuadro con la red neuronal que se va a entrenar, viéndose las capas, neuronas, las entradas y las salidas.
A continuación, con la función *train*, se lleva a cabo el entrenamiento de la red, obteniendo como salida la red y los datos que se han usado para el entrenamiento, validación o testeo.
Por último, se guardaría la red con el comando *save* y se genera la función correspondiente mediante *genFunction*.

8.2.2 Archivo error_red_salidasgen.m

En este archivo, se trata de cuantificar el error de las redes neuronales obtenidas en el apartado 8.2.1. Para ello, se empieza cargando las redes, mediante el comando *load* y se toman de cada, una las salidas que se vayan a analizar con los índices de testeo guardados en la red neuronal.
Para calcular el error relativo entre lo que debería dar la red y lo que da, se realiza un bucle "for", que va calculando para cada una de las entradas correspondientes al testeo, la salida que se quiera comprobar, mediante la función generada por la red neuronal. Despues, se restan ambos vectores y se guardan.

8.2.3 Archivos cuantif_error_ash/BDG/damage.m

Son varios archivos pero todos tienen la misma estructura, el objetivo es comparar en una misma gráfica los errores de los resultados de distintas redes neuronales que se han calculado anteriormente. Para compararlos, se cargan los errores que se han guardado en el archivo *error_red_salidasgen.m*, y se hace una media aritmética con los vectores de las distintas redes (con diferentes casos de entrenamiento). Y, finalmente se pinta la gráfica comparando las 3 redes neuronales, de 500, 1000 y 2000 casos de entrenamiento.

8.3 Optimización

El objetivo de este programa es obtener el tratamiento óptimo de Denosumab para un paciente determinado, para ello, hay que interactuar con el programa una vez ejecutado, de esa forma, es lo más general posible y el código no se cambia para cada paciente.

8.3.1 Archivo algoritmo_genetico.m

Este archivo se ha realizado de forma que cuando se ejecute, el programa interactúe con el usuario. Las primeras líneas consisten en la entrada de datos del programa por parte del usuario, donde se tendrá que poner el k_{res}, $\Delta\sigma$, el tiempo que lleva de enfermedad y el P_{RANKL}^{PMO} del paciente. A continuación hay una línea con los valores iniciales de las constante incógnita del problema de optimización. Son unos valores comprendidos en el rango en el que se tienen que encontrar dichas variables.
Las siguientes líneas del código muestran los límites entre los que se pueden encontrar las variables incógnita, además de unas matrices de restricción por si hubiese que relacionar las incógnitas, pero en nuestro caso está en blanco.
Una vez que ya se tienen las condiciones iniciales y restricciones, se procede al algoritmo en sí. En este caso se han hecho 15 iteraciones mediante un bucle "for". Dentro de ese bucle, en primer lugar, se toman las variables que se van a optimizar de forma aleatoria, por eso se usa el comando "randi", que las va tomando en el intervalo que se indique. Después, mediante la función *gaoptimiset*, una función de Matlab donde se ponen las opciones del algoritmo (tamaño de la población, número de generaciones, tolerancia de la función etc.). En la siguiente línea está la llamada a la función a optimizar *fecuacionesga12*, pasando los parámetros dados por el usuario. Para terminar el bucle, se llama al algoritmo genético.
Se han puesto también 3 líneas de código en la que se analiza cúanto valen para ese tratamiento óptimo, la fracción de ceniza, el daño y BDG.
Por último, están las líneas que muestran el resultado final, mediante la función "disp" y los resultados de BDG, daño y fracción de ceniza que se optienen para ese tratamiento.

Función fecuacionesga12

Esta función es la que se va a optimizar y tiene de entrada en primer lugar, las incónitas y en segundo lugar, los parámetros constantes. A continuación, se llama a las funciones obtenidas de la red neuronal mejor ajustadas, para que den los valores de salida, BDG, fracción de ceniza y daño, mediante las variables *salida_BDG*, *salida_ash*, *salida_damage* respectivamente.
Después se lleva a cabo un problema de penalización, de forma que si el daño y la fracción de ceniza superan un cierto valor, se incrementa este en 100, para que así, a la hora de la optimización, se deseche esa opción de forma rápida. Esto se ha llevado a cabo a través de dos bucles "for", que iban recorriendo las componentes de los vectores que daban las funciones anteriormente mencionadas.
Por último, se hace un promedio con las componentes del vector *salida_BDG*, y la función objetivo a minimizar estará compuesta por el menos promedio del BDG, y la suma de las penalizaciones del daño y la fracción de ceniza.

9 Manual del usuario

El objetivo de este capítulo es facilitar la tarea al usuario final de este programa, para ello, se va a ilustrar mediante imágenes y breves comentarios sobre cómo debe ir contestando para obtener el resultado final. El programa con el que tiene que interactuar es Matlab.
En primer lugar, teniendo el archivo de *algoritmo_genetico.m* abierto (ver en la figura 9.1), se le da a F5 o al triángulo superior verde. Entonces aparecerá un diálogo con el usuario en el que se le solicitará el k_{res} del paciente en el *Command window*, ver en la figura 9.2a. A continuación, de igual forma, el programa pide al usuario que inserte el incremento (+) o decremento (-) de la actividad física del paciente, mediante la escritura de $eps_usuario$, ver figura 9.2b. Después, pedirá el tiempo que lleva con la enfermedad sin tratamiento (tiempo_enfermedad), ver en la figura 9.2c. Por último, se solicita el P^{PMO}_{RANKL} del paciente (Prankl_usuario), ver en la figura 9.2d.

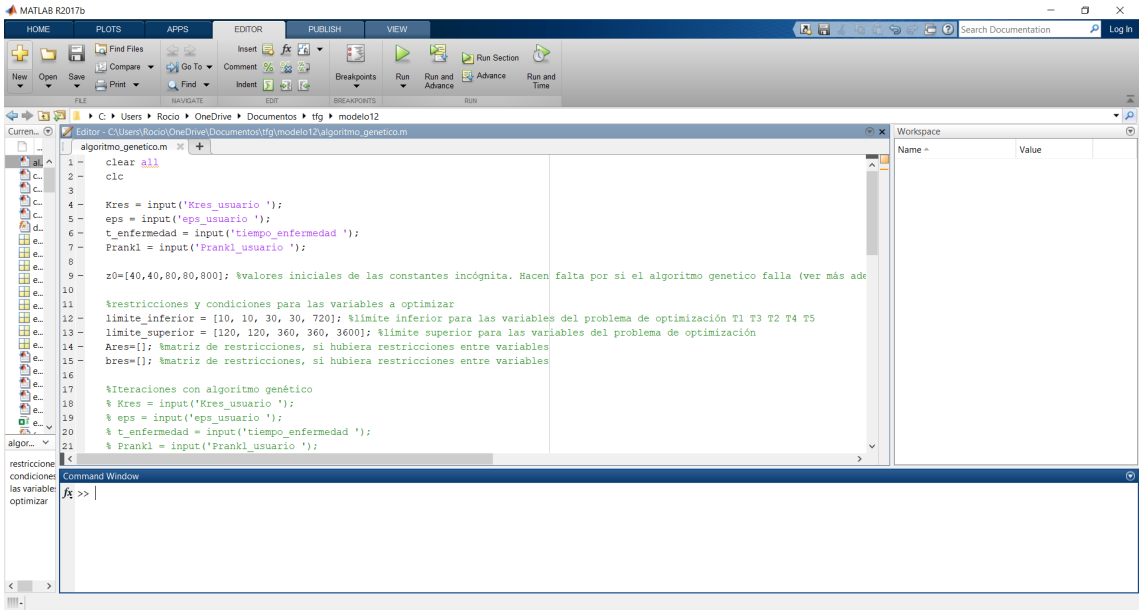
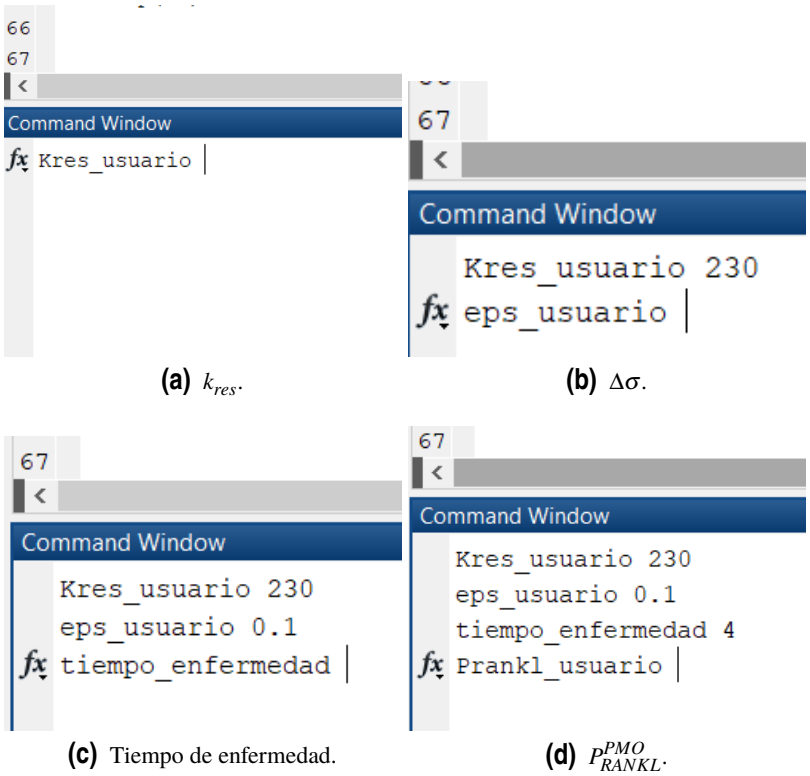


Figura 9.1 Impresión de la pantalla con el programa abierto.

Posteriormente, cuando el programa ha terminado de ejecutarse aparecerá de nuevo en el *Command window* un comentario que explica cuál es el tratamiento óptimo (ver en la figura 9.3)



(a) k_{res}. **(b)** $\Delta\sigma$.

(c) Tiempo de enfermedad. **(d)** P^{PMO}_{RANKL}.

Figura 9.2 Comunicación del usuario con el *Command window* para escribir los datos del paciente.

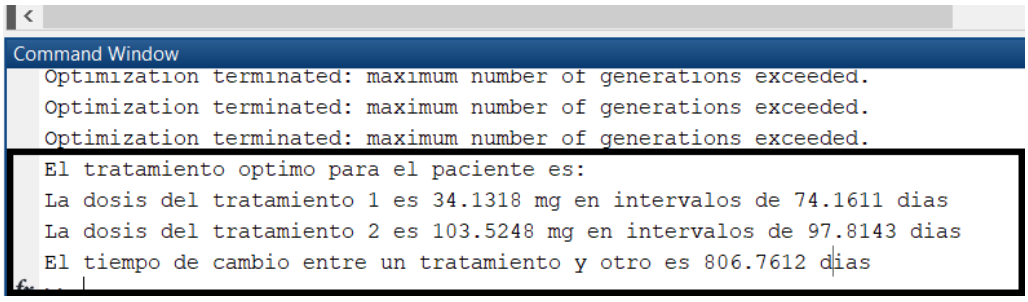


Figura 9.3 Comunicación final con el usuario en la que le expone el tratamiento óptimo para el paciente concreto.

10 Conclusiones y trabajos futuros

Tras haber realizado este trabajo, se han obtenido una serie de conclusiones en cuanto a los parámetros que caracterizan a las redes neuronales, para que ajusten de forma correcta los datos. En cuanto a la función de entrenamiento, se vió que la función trainbr, aunque tarda más la ejecución, es la que dio mejores resultados para la red de la fracción de ceniza y para el BDG. Sin embargo, para el caso del daño, la que mejor resultado dio fue la función trainscg. Esto puede ser debido a la peculiaridad de la distribución de los datos, que estaban o en torno a 0 o cerca de 1, pero no había datos entre ambos valores.
Por otro lado, en cuanto a la cantidad de capas y neuronas en la red, se apreció que donde mejores resultados había era en el caso de 2 capas, con 40 y 10 neuronas respectivamente.
El número de casos afecta al entrenamiento de la red. Así, se esperaba que a mayor número de datos, la red ajustara mejor, pero en el daño no ocurrió así, sino que en el caso de 2000, se sobreentrenó la red, dando peores resultados que el de 1000 casos. Como era de esperar, con 500 casos se obtuvieron los peores resultados, ya que era un número insuficiente de entradas.
Otra conclusión que se ha extraído en cuanto a la red neuronal es que de forma general, ésta ha ido prediciendo peor las salidas a medida de que el tiempo iba creciendo. Esto podría mejorarse en un futuro cambiando parámetros de la red.
En cuanto a los resultados en sí, se puede apreciar que en general, la frecuencia de los tratamientos no cambia con la variación de los parámetros del paciente, siendo el tiempo entre una inyección y otra del tratamiento 2 mayor que en el tratamiento 1, salvo en el caso de la variación del tiempo de enfermedad, ya que sobre el año 5, se intercambian las frecuencias de los 2 tratamientos, además la dosis necesaria para el tratamiento, es mayor en la fase 2 o tratamiento 2 que en la 1. Por otro lado, de forma más particular, se pueden ir discutiendo los resultados obtenidos para la variación de cada uno de los parámetros característicos del paciente:

- En el caso de variación de k_{res}, se obtuvo que la cantidad de dosis necesaria para lograr el tratamiento óptimo iba disminuyendo a medida que aumentaba k_{res}. Sin embargo, cuando ésta valía 150 % se producía un mínimo, a partir del cual, era necesario más dosis de tratamiento.

  Evaluando cómo influye esta variable en cada una de las salidas, se aprecia que en la fracción de ceniza y en el BDG apenas afecta su variación, mientras que en el caso del daño, si k_{res} está en torno a 100 %, alcanza valores altos, pudiendo llegar a ser fractura, aunque esto es un caso más límite.

- Para el caso de variación de $\Delta\sigma$, se aprecia que para el tratamiento 1, a medida que la actividad física aumenta, hace falta inyectar menos dosis para lograr el tratamiento óptimo, mientras que en el tratamiento 2, cuando la variación de actividad física es negativa y ésta va aumentando positivamente, hay que suministrar mayor dosis, pero cuando se alcanza un cierto valor de actividad física la cantidad necesaria de tratamiento disminuye.

Viendo cómo afecta la variación de $\Delta\sigma$ en cada una de las salidas, se aprecia que no influye mucho en la fracción de ceniza, pero se puede observar que a mayor actividad física, la mineralización es menor. En cuanto a la influencia en el BDG, se ve que a mayor actividad física, el incremento de ganacia ósea es mayor, mientras que en el caso del daño, si la actividad física es demasiado elevada, puede provocar rotura del hueso. Por el contrario, cuando la actividad es demasiado pequeña, o casi nula, el daño es casi inexistente.

- En el caso del estudio de la variación del tiempo de enfermedad, se obtuvo que si el tiempo que ha transcurrido de enfermedad sin tratamiento es pequeño, hace falta más dosis para conseguir el tratamiento óptimo.

  Evaluando cómo influye esta variable en cada una de las salidas, se aprecia que en la fracción de ceniza, cuanto mayor sea el tiempo de enfermedad, menor mineralización habrá a largo plazo con el tratamiento óptimo. Sin embargo, en cuanto al porcentaje de ganancia ósea, se puede ver que cuanto menos tiempo de enfermedad tenga, se consigue de forma significativa un BDG mayor. Por otro lado, en el caso del daño, a largo plazo no influye apenas, lográndose un ligero menor daño cuando el tiempo de enfermedad es mayor.

- Por último, estudiando el caso de variación de P_{RANKL}^{PMO}, se obtuvo que hasta valores de 2500 pM/día, había que suministrar mayor dosis de tratamiento para que fuera el óptimo, sin embargo, a partir de ese valor, hacía falta menos dosis.

  Viendo la influencia de esta variable en cada una de las salidas, se aprecia que cuanto mayor es la cantidad de P_{RANKL}^{PMO}, la mineralización a largo plazo es menor. En cuanto al BDG, apenas influye la variación de esta variable, consiguiéndose valores muy similares a largo plazo, mientras que en el daño, cuando el P_{RANKL}^{PMO} es mayor, se logran mejores resultados.

Como posible trabajo futuro a partir del realizado en este TFG, se podrían añadir más parámetros característicos del paciente como por ejemplo el peso, automatizándolo de la misma forma que el k_{res}, tiempo de enfermedad etc. También sería interesante analizar qué pasaría en el caso de que el incremento de carga fuera progresivo, y no un valor constante desde el comienzo.

Índice de Figuras

Índice de Tablas

Índice de Códigos

Bibliografía

[1] "Hueso-ecured," https://www.ecured.cu/Hueso, 2019, accedido 10-06-2020.

[2] "Tipos de huesos del esqueleto humano - esqueleto humano," http://esqueletohumano.net/tipos-huesos-del-esqueleto-humano, 2020, accedido 10-06-2020.

[3] "Tema 2. fisiopatología ósea," https://www.ucm.es/data/cont/docs/420-2014-02-18-01%20fisiopatologia%20osea.pdf, 2019, accedido 10-06-2020.

[4] P. Mandalunis, "Remodelación ósea," *Actualizaciones en Osteología*, vol. 2, no. 1, pp. 16–18, 2006.

[5] P. Pivonka, J. Zimak, D. W. Smith, B. S. Gardiner, C. R. Dunstan, N. A. Sims, T. J. Martin, and G. R. Mundy, "Model structure and control of bone remodeling: a theoretical study," *Bone*, vol. 43, no. 2, pp. 249–263, 2008.

[6] J. A. Riancho and J. Delgado-Calle, "Mecanismos de interacción osteoblasto-osteoclasto," *Reumatología Clínica*, vol. 7, pp. 1–4, 2011.

[7] Cellpath, "Traposteoclastculture," https://es.wikipedia.org/wiki/Osteoclasto/media /Archivo:TRAPosteoclastculture.jpg, 2007, accedido 10-06-2020.

[8] G. Caponetti, "Osteoblast," https://es.wikipedia.org/wiki/Osteoblasto/media /Archivo:Osteoblast.jpg, 2012, accedido 10-06-2020.

[9] I. F.-T. Hernández-Gil, M. A. A. Gracia, M. Pingarrón, and L. B. Jerez, "Bases fisiológicas de la regeneración ósea i. histología y fisiología del tejido óseo," *Med Oral Patol Oral Cir Bucal*, vol. 11, pp. 47–51, 2006.

[10] J. González Macías and J. Olmos Martínez, "Fisiopatología de la osteoporosis y mecanismo de acción de la pth," *Rev Osteoporos Metab Miner*, vol. 2, no. Supl 2, pp. S5–S17, 2010.

[11] Posible2006, "Osteocitos," https://es.wikipedia.org/wiki/Archivo:Osteocitos.jpg, 2018, accedido 10-06-2020.

[12] J. N. Bilbao, A. C. Sánchez, and S. P. Gil-Antuñano, "Regulación del metabolismo óseo a través del sistema rank-rankl-opg," *Revista de Osteoporosis y Metabolismo Mineral*, vol. 3, no. 2, pp. 105–112, 2011.

[13] S. E. de Geriatría y Gerontología, https://www.segg.es/ciudadania/2017/02/23/en-españa-la-osteoporosis-afecta-a-3-millones-y-medio-de-personas, 2018, accedido 11-06-2020.

[14] T. D. Rachner, S. Khosla, and L. C. Hofbauer, "Osteoporosis: now and the future," *The Lancet*, vol. 377, no. 9773, pp. 1276–1287, 2011.

[15] M. Hermoso de Mendoza, "Clasificación de la osteoporosis: Factores de riesgo. clínica y diagnóstico diferencial," in *Anales del sistema sanitario de Navarra*, vol. 26. SciELO Espana, 2003, pp. 29–52.

[16] M. P. Alcázar, "Osteoporosis: Huesos frágiles," *Farmacia profesional*, vol. 16, no. 2, pp. 46–52, 2002.

[17] S. Scheiner, P. Pivonka, D. Smith, C. Dunstan, and C. Hellmich, "Mathematical modeling of postmenopausal osteoporosis and its treatment by the anti-catabolic drug denosumab," *International journal for numerical methods in biomedical engineering*, vol. 30, no. 1, pp. 1–27, 2014.

[18] J. Martínez-Reina and P. Pivonka, "Effects of long-term treatment of denosumab on bone mineral density: insights from an in-silico model of bone mineralization," *Bone*, vol. 125, pp. 87–95, 2019.

[19] P. Pivonka, P. R. Buenzli, S. Scheiner, C. Hellmich, and C. R. Dunstan, "The influence of bone surface availability in bone remodelling—a mathematical model including coupled geometrical and biomechanical regulations of bone cells," *Engineering Structures*, vol. 47, pp. 134–147, 2013.

[20] H. Frost, "Bone's mechanostat: A 2003 update," *The Anatomical Record Part A*, vol. 275, no. A, pp. 1081–1101, 2003.

[21] J. Lemaitre and J. Chaboche, *Mechanics of Solid Materials*. Cambridge, UK: Cambridge University Press, 1990.

[22] C. Pattin, W. Caler, and D. Carter, "Cyclic mechanical property degradation during fatigue loading of cortical bone," *J Biomech*, vol. 29, no. 1, pp. 69–79, 1996.

[23] J. Martínez-Reina, J. Garía-Aznar, J. Domínguez, and M. Doblaré, "A bone remodelling model including the directional activity of bmus," *Biomech Model Mechanobiol*, vol. 8, no. 2, pp. 111–127, 2009.

[24] J. Currey, "Tensile yield in compact bone is determined by strain, post-yield behaviour by mineral content," *J Biomech*, vol. 37, no. 4, pp. 549–556, 2004.

[25] "Significado de neurona," https://www.significados.com/neurona/, 2019, accedido 01-07-2020.

[26] "Conceptos básicos sobre redes neuronales," http://grupo.us.es/gtocoma/pid/pid10/ Redes-Neuronales.htm, accedido 01-07-2020.

[27] F. Mateo Jiménez, "Redes neuronales y preprocesado de variables para modelos y sensores en bioingeniería," Ph.D. dissertation, 2012.

[28] A. Ballesteros, "Clasificación de las redes neuronales respecto al aprendizaje," http://www.redes-neuronales.com.es/tutorial-redes-neuronales/ clasificacion-de-redes-neuronales-respecto-al-aprendizaje.htm, accedido 01-07-2020.

[29] E. Allibhai, "Building a deep learning model using keras," https://towardsdatascience.com/building-a-deep-learning-model-using-keras-1548ca149d37, 2018, accedido 17-06-2020.

[30] H. Demuth, M. Beale, and M. Hagan, "Neural network toolbox," *For Use with MATLAB. The MathWorks Inc*, vol. 2000, 1992.

[31] J. López, L. Lanzarini, and G. Leguizamón, "Optimización multiobjetivo: aplicaciones a problemas del mundo real," *Buenos Aires, Argentina, Universidad Nacional de la Plata*, pp. 66–90, 2013.

[32] F. W. Glover and G. A. Kochenberger, *Handbook of metaheuristics*. Springer Science & Business Media, 2006, vol. 57.

[33] F. Caparrini, "Metaheurísticas para búsqueda y optimización (parte 1) - fernando sancho caparrini," https://www.citethisforme.com/cite/sources/websiteautociteeval, 2018, accedido 02-07-2020.

[34] J. Arranz de la Peña and A. Parra Truyol, "Algoritmos genéticos," *Universidad Carlos III*, 2007.

[35] J. P. A. Marcos; Rivero Gestal (Daniel; Rabuñal, Juan Ramón; Dorado and M. Gestal, *Introducción a los algoritmos genéticos y la programación genética*. Universidade da Coruña, 2010.

Anexo I

1 Programa principal del modelo matemático

```
% clc
% clear all
% close all
% delete *.dat% delete initial data files

function run_bonemodel12(trat,eps,K_res,alfa,PRANKLpmo,tiempo_enfermedad
    )

T1 = trat(1);
T3 = trat(2);
T2 = trat(3);
T4 = trat(4);
T5 = trat(5);

% run_BONEREMODELING
%
% bone remodeling model based on the BCPM of Pivonka et al. 2008
% extended towards mechanical feedback and consideration of PMO
% and treatment with Denosumab
%
% JL 2019 model debbuged and improved for clarity
%
% Javier Martínez-Reina, 2019. Inclusion of damage
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Difference with run_bonemodel_v10b:
% What is new in this routine?
% Proliferation term is redefined as a piecewise linear function
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% run file for mechanostat theory
%
% as input parameters a proper loading case LC needs to be chosen
% currently a uniaxial compressive stress state is set as the
```

```
% load case at sig = - 30 MPa; this could be representative for bending in
% a long bone;
%
% mechanical load cases:
% LC=14 ... underloading sig_new = - 25 MPa
% LC=15 ... overloading sig_new = - 35 MPa
%
%
% the mechanostat needs to work under the following load cases
%
% (1) PTH continuous administration:
%    split up of DOBu*OBu*pi_TGFBe - into a diff & prolif term is very
%    sensitive - check that the continuous PTH case works
%
% (2) Mechanical loading - gain in bone mass after increase of mech load
%
% (3) Mechanical unloading - loss of bone (low turnover)
%
% (4) case of enforcing osteocyste apoptosis
%
%

%%%%%%
%%%%%% PARAMETERS DECIDED BY THE USER
%%%%%%

LoadCase = 37; %see numbers in fu_loadcase
tinterval = tiempo_enfermedad;
years = tinterval+10;
tdisease = years; %time in years for TimeInterval_3. End of drug administration interval/disease simulation
% Import the temporal evolution of Denosumab with the corresponding dose
load('treatment_02REV.mat') % The array Denosumab is imported


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% read model input parameters
ModelParameter = fu_modelparameters12();

ModelParameter.Alpha = alfa;
ModelParameter.P_RANKL_PMO = PRANKLpmo;
ModelParameter.Kres = K_res;
K_res  = ModelParameter.K_res;
tresid = ModelParameter.tresid;
DT     = ModelParameter.DT;
```

```
79  MINP   = ModelParameter.MINP;
80  XKAPPA = ModelParameter.XKAPPA;
81  VMPRIM = ModelParameter.VMPRIM;
82  VMMAX  = ModelParameter.VMMAX;
83  DeltaT = ModelParameter.DeltaT;
84  dam_t0 = ModelParameter.dam_t0;
85  OCY_t0 = ModelParameter.OCY_t0;
86  fbm_t0 = ModelParameter.fbm_t0;
87  OBp_t0 = ModelParameter.OBp_t0;
88  OBa_t0 = ModelParameter.OBa_t0;
89  OCp_t0 = ModelParameter.OCp_t0;
90  OCa_t0 = ModelParameter.OCa_t0;
91
92  % define drug administration interval/disease
93  TimeInitial = 0;
94  TimeEnd = years*360/DeltaT;
95  TimeInterval_1 = 0; %start to simulate drug administration/disease
96  TimeInterval_2 = tinterval*360; % Days elapsed from the onset of
        disease to the beginning of treatment
97  TimeInterval_3 = tdisease*360; %end of drug administration interval/
        disease simulation
98  BW = ModelParameter.BW;
99
100 %Initialization of cells numbers
101 cells_0 = [OBp_t0,...
102           OBa_t0,...
103           OCp_t0,...
104           OCa_t0];
105
106 %Initialization of CellsVector (solution vector)
107 CellsVector = zeros(TimeEnd, 10);
108 ParameterVector = zeros(TimeEnd, 30);
109 % Initialization of VFPREV
110 % Comment of the following
111  VFPREV = zeros(tresid/DeltaT,1);
112 %load('VFPREV.mat')
113
114 %%%%%%%%
115 %%%%%%%% STEADY STATE CALCULATION
116 %%%%%%%%
117 %
118 % options=optimset('TolFun',1e-14,'TolX',1e-14); % Option tolerances
119 % ecsteadystate = @(z)fu_steadystate8(z,ModelParameter); %JL
120 % [cells_t0] = fsolve(ecsteadystate,cells_0,options);
121 % OBp_t0 = cells_t0(1);
122 % OBa_t0 = cells_t0(2);
123 % OCp_t0 = cells_t0(3);
124 % OCa_t0 = cells_t0(4);
125 %
126 % %update structure parameters
```

```
% ModelParameter.OBp_t0 = OBp_t0;
% ModelParameter.OBa_t0 = OBa_t0;
% ModelParameter.OCp_t0 = OCp_t0;
% ModelParameter.OCa_t0 = OCa_t0;

% some of the model parameters are not independent:
% (1)K_form ... determined by K_res and dBV_t0. resorption/formation rate from original parameter set

% K_form = K_res*OCa_t0/OBa_t0;

Cell = Main_homeost_v2(trat,K_res,alfa);
OBp_t0 = Cell(end,1);
OBa_t0 = Cell(end,2);
OCY_t0 = Cell(end,3);
OCp_t0 = Cell(end,4);
OCa_t0 = Cell(end,5);
fbm_t0 = Cell(end,6);
dam_t0 = Cell(end,7);
ash_t0 = Cell(end,8);

ModelParameter.OBp_t0 = OBp_t0;
ModelParameter.OBa_t0 = OBa_t0;
ModelParameter.OCY_t0 = OCY_t0;
ModelParameter.OCp_t0 = OCp_t0;
ModelParameter.OCa_t0 = OCa_t0;
ModelParameter.fbm_t0 = fbm_t0;
ModelParameter.dam_t0 = dam_t0;
ModelParameter.ash_t0 = ash_t0;

K_form = ModelParameter.K_form;

% Calculate the initial ash fraction (ash_fraction0) as a function of the initial turnover rate

% First, the queue VFPREV is initialized as a function of DVF and DVR,
% the volumes of tissue formed and resorbed at the equilibrium

DVR = OCa_t0*K_res*DeltaT/100;
DVF = DVR;            % Remodelling equilibrium is assumed
p   = 1-fbm_t0/100;   % Initial porosity in [0,1]

i=0;
while i<length(VFPREV)
    VFPREV = fu_updateVFPREV12(p,DVR,DVF,VFPREV);
    i=i+1;
end

% Then, the initial ash fraction is calculated

```

```
175 VM_t0 = 0;
176
177 for i=1:length(VFPREV)-1
178 VM_t0 = VM_t0 + VFPREV(i)*fu_calmineral12(i*DeltaT,DT,MINP,XKAPPA,VMPRIM
        ,VMMAX);
179 end
180
181 VM_t0 = VM_t0 + VFPREV(length(VFPREV)) * VMMAX; % Interstitial tissue
        at maximum mineral content
182
183 VM_t0=VM_t0/(1-p);
184 % %
185 % % ash_t0 = 3.2*VM_t0/(3.2*VM_t0+1.1*3.0/7.0);
186 % %
187 % % ModelParameter.ash_t0 = ash_t0; %save the new parameter
188
189 % Update Modelparameter as a function of the load case. Call individual
        loadcase fu, function of the loadcase and loading times.
190 ModelParameter = fu_loadcase12(LoadCase,TimeInterval_1,TimeInterval_2,
        TimeInterval_3,TimeInitial,ModelParameter,eps);
191
192 D_OBa  = ModelParameter.D_OBa;
193
194 % some of the model parameters are not independent:
195 % (2)A_OCY ... determined by D_OBa and dOCY_t0
196 % compute OCY apoptosis rate
197 A_OCY = D_OBa*OBa_t0/OCa_t0;
198
199 ModelParameter.A_OCY = A_OCY; %save the new parameter
200
201 % define the initial conditions for the ODE solver
202
203 CellsVector(1,:)=[OBp_t0,OBa_t0,OCY_t0,OCp_t0,OCa_t0,fbm_t0,dam_t0,VM_t0
        ,0,0];
204
205 %%%%%%%%
206 %%%%%%%% ODE CALCULATION
207 %%%%%%%%
208
209 label_form = 0;
210 for i=2:TimeEnd
211     % Integration of ODEs
212     [xdot,AuxParameterVector] = fu_bonecelldynamics12(i,CellsVector(i
            -1,:),ModelParameter,LoadCase,TimeInterval_1,TimeInterval_2,
            TimeInterval_3,eps);
213     ParameterVector(i-1,:)=AuxParameterVector';
214     for j=1:7
215         CellsVector(i,j) = CellsVector(i-1,j) + xdot(j) * DeltaT;
216         if CellsVector(i,j) < 0
217             CellsVector(i,j) = 0;
```

```
218         end
219     end
220     % Checking that fbm lies within [1,100]
221     if CellsVector(i,6) > 100
222         CellsVector(i,6) = 100;
223         label_form = 1;
224     elseif CellsVector(i,6) < 1
225         CellsVector(i,6) = 1;
226         label_form = 1;
227     end
228 
229     % Checking if damage lies within [0,0.99]
230     if CellsVector(i,7) > 0.99
231         CellsVector(i,7) = 0.99;
232     elseif CellsVector(i,7) < 0
233         CellsVector(i,7) = 0.0;
234     end
235 
236     % Denosumab concentrations
237     [den_plasma,den_SC] = denosumab12(CellsVector(i-1,9),CellsVector(i
            -1,10),DeltaT);
238     CellsVector(i,9) = den_plasma;
239     time_injection = (i-1) * DeltaT - TimeInterval_2;
240     CellsVector(i,10) = den_SC + injection12(BW,time_injection,T1,T3,T2,
            T4,T5);
241 
242     OBa  = CellsVector(i,2);
243     OCa  = CellsVector(i,5);
244     fvas = 100 - CellsVector(i,6);
245 
246     %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
247     % Mineralization
248     % Update the queue VFPREV
249     if label_form == 0
250         aux1 = OCa*K_res*DeltaT/100;
251         aux2 = OBa*K_form*DeltaT/100;
252     else
253         aux1 = min(OCa*K_res*DeltaT/100,OBa*K_form*DeltaT/100);
254         aux2 = aux1;
255     end
256     VFPREV = fu_updateVFPREV12(fvas/100,aux1,aux2,VFPREV);
257 
258     % Calculate the mineral content
259     VM = 0;
260 
261     for j=1:length(VFPREV)-1
262         VM = VM+VFPREV(j)*fu_calmineral12(j*DeltaT,DT,MINP,XKAPPA,VMPRIM,
                VMMAX);
263     end
```

```
264     VM = VM + VFPREV(length(VFPREV)) * VMMAX; % Interstitial tissue at
            maximum mineral content
265
266     VM=VM/(1-fvas/100);
267
268     ash = 3.2*VM/(3.2*VM+1.1*3.0/7.0);
269
270     % Uncomment the following lines to switch off mineralization
271     % ash = ash_t0;
272     % VM = VM_t0;
273
274     % End of mineralization
275     %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
276     CellsVector(i,8)  = VM;
277 end
278 TimeVector = 1:TimeEnd;
279
280 %last component for ParameterVector
281 [~,AuxParameterVector] = fu_bonecelldynamics12(TimeEnd,CellsVector(
        TimeEnd,:),ModelParameter,LoadCase,TimeInterval_1,TimeInterval_2,
        TimeInterval_3,eps);
282 ParameterVector(TimeEnd,:) = AuxParameterVector';
283 ParameterVector(TimeEnd,23)=TimeEnd;
284
285 write_steady_state = 1;
286
287 if write_steady_state == 1
288
289     OBp_t0 = CellsVector(length(CellsVector(:,1)),1)
290     OBa_t0 = CellsVector(length(CellsVector(:,1)),2)
291     OCY_t0 = CellsVector(length(CellsVector(:,1)),3)
292     OCp_t0 = CellsVector(length(CellsVector(:,1)),4)
293     OCa_t0 = CellsVector(length(CellsVector(:,1)),5)
294     BV_t0  = CellsVector(length(CellsVector(:,1)),6)
295     dam_t0 = CellsVector(length(CellsVector(:,1)),7)
296     ash_t0 = CellsVector(length(CellsVector(:,1)),8)
297
298
299     PTH_tot       = ParameterVector(length(CellsVector(:,1)),1)
300     OPG_eff_t0    = ParameterVector(length(CellsVector(:,1)),2)
301     OPG_t0        = ParameterVector(length(CellsVector(:,1)),3)
302     RANK_t0       = ParameterVector(length(CellsVector(:,1)),4)
303     RANKL_eff_t0  = ParameterVector(length(CellsVector(:,1)),5)
304     RANKL__tot_t0 = ParameterVector(length(CellsVector(:,1)),6)
305     RANKL_t0      = ParameterVector(length(CellsVector(:,1)),7)
306     RANKL_RANK_t0 = ParameterVector(length(CellsVector(:,1)),8)
307     Pi_PTH_act    = ParameterVector(length(CellsVector(:,1)),9)
308     Pi_PTH_rep    = ParameterVector(length(CellsVector(:,1)),10)
309 Pi_TGFbeta_OBu_act = ParameterVector(length(CellsVector(:,1)),11)
310 Pi_TGFbeta_OCa_act = ParameterVector(length(CellsVector(:,1)),12)
```

```
Pi_TGFbeta_OBp_rep = ParameterVector(length(CellsVector(:,1)),13)
    Pi_RANK_act   = ParameterVector(length(CellsVector(:,1)),14)
    Pi_MCSF_act   = ParameterVector(length(CellsVector(:,1)),15)
      sig_macro   = ParameterVector(length(CellsVector(:,1)),16)
        SED_bm0   = ParameterVector(length(CellsVector(:,1)),17)
        SED_bm    = ParameterVector(length(CellsVector(:,1)),18);
        Pi_eps    = ParameterVector(length(CellsVector(:,1)),19)
end

out1              = TimeVector(1:length2);
out2              = ParameterVector(1:length2,25);
out3              = CellsVector(1:length2,8);
fbm               = CellsVector(1:length2,6);
ash               = 3.2*out3 ./ (3.2*out3 + 1.1*3.0/7.0);
strain            = ParameterVector(1:length2,27);

%save('treatment_24REV_new_denosumab.mat',...
%    'out1','out2','ash','fbm','CellsVector','TimeVector')

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%Inclusión del BDG

timep = out1'*0.25;               % El vector de pasos de integración se
    convierte a tiempo, multiplicando por Delta_t
dens60 = out2;                    % Se le da otro nombre a la densidad
    aparente

% Se definen los instantes de tiempo donde se van a guardar los
    resultados.

t(1) = 1;

for x = 1:((14400/720))
    t(x+1) = 720*x;
end



years      = 360*4; %Step en el que comienza el tratamiento (después de
    1 año de PMO)
total      = length(timep); %30660



BDG(1:years) = 0;

for x2 = years:(total)
    BDG(x2) = (dens60(x2)-dens60(years))/dens60(years)*100;
end
```

```
357
358
359
360 for n2 = 1:21
361     BDG_vector(n2) = BDG(t(n2));
362 end
363
364 save('results52_sigma0.5T_RANKLPMO=2e3_sigmoidal=2years_ONSET=3
        years_zeta=0.2.mat','out1','out2','ash','fbm','CellsVector','strain'
        ,...
365     'ParameterVector','BDG_vector','BDG')
366
367 % save('VFPREV.mat','VFPREV')
```

2 Formación del vector de concentraciones del tratamiento

```
1  function denos_SC = injection12(BW,time,T1,T3,T2,T4,T5)
2
3  T1 = T1*(1e+06)/BW;
4  T3 = T3*(1e+06)/BW;
5
6  %T1 es la dosis del tratamiento normal en mg
7  %T3 es la dosis del tratamiento reducido en mg
8  %T2 es el tiempo en días entre el tratamiento normal
9  %T4 es el tiempo en días entre el tratamiento reducido
10 %T5 es el tiempo entre los dos tratamientos
11
12 %Calculo el máximo común divisor entre los tiempos de tratamiento
13 minbwT2T4 = gcd(T2,T4);
14 minbwT5T4 = gcd(T4,T5);
15 minbwT2T5 = gcd(T2,T5);
16
17 n_minintervalo = min([minbwT2T4,minbwT5T4,minbwT2T5]);
18
19 inj_time1 = [0:n_minintervalo:7200];
20
21   cantidad_actual = T1;
22   cantidad_cambio = T3;
23   periodicidad_actual = T2;
24   periodicidad_cambio = T4;
25
26   dias_pasados = 0;
27   pos_vector = 1;
28   vector = [];
29   vector(pos_vector) = cantidad_actual;
30   contador_dosis = periodicidad_actual;
31   contador_cambio = T5;
32
```

```
  while (dias_pasados <= (7200-n_minintervalo))
   %Pasan los días y se actualizan los contadores
     dias_pasados = dias_pasados+n_minintervalo;
     contador_dosis = contador_dosis-n_minintervalo;
     contador_cambio = contador_cambio-n_minintervalo;
     pos_vector = pos_vector+1;

    if contador_cambio == 0
      %Se cambian las variables. Esto en matlab se puede hacer con [b, a
         ] = deal(a,b)
      [cantidad_actual, cantidad_cambio] = deal(cantidad_cambio,
         cantidad_actual);
      [periodicidad_actual, periodicidad_cambio] = deal(
         periodicidad_cambio, periodicidad_actual);
      %Se reinicia el contador
      contador_cambio = T5;
      vector(pos_vector) = cantidad_actual;
      contador_dosis = periodicidad_actual;

    else

    %Si toca dosis, se pone la del tratamiento actual. Si no, se pone un
        0
    if(contador_dosis == 0)
      vector(pos_vector) = cantidad_actual;
      contador_dosis = periodicidad_actual;
    else
      vector(pos_vector) = 0;
    end
    end
  end

  multi_dose = vector;
  for ij = 1:length(inj_time1)
    if(time == inj_time1(ij))
        denos_SC = multi_dose(ij);
        break;
    else
        denos_SC = 0.0;
    end
end
```

3 Homeostasis. Programa principal

```
% clc
% clear all
% close all
function CellsVector = Main_homeost_v2(trat,K_res,alfa)
```

```
5
6  % alfa = 1;
7
8  %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
9  %valores iniciales
10 OBp0 = 1.235696223378287e-03; %en ppio fijo
11 OBa0 = 5.828051929437614e-04; %en ppio fijo
12 OCY0 = 1.000000000000000e-02; %en ppio fijo
13 OCp0 = 1.000000000000000e-03; %en ppio fijo
14 OCa0 = 1.165608258940796e-04; %en ppio fijo
15 factor_cond_inic = 1.1; % Incremento de fbm_t0 de un step al siguiente
16 fbm0 = 35;
17 d0   = 1e-05;
18 tresid=3000/0.25;
19 VFPREV = zeros(tresid,1);
20 VFPREV0=VFPREV;
21 iterfin=49; %iteraciones consecutivas para que corte el cálculo
22 %vector de cargas sobre las que se va a calcular el estado de
      homesotasis
23 eps = 0.5;
24 % Comment for compression
25 % eps = - eps;
26 %vector de tolerancias para cuando se considera que ha convergido
27 %[OBp0,OBa0,OCY0,OCp0,OCa0,fbm0,d0,ash0] % Para el daño, originalmente
      0.03
28  %tol=[0.01,0.01,0.01,0.01,0.01,0.01,0.01,0.01];
29  tol=[0.001,0.001,0.001,0.001,0.001,0.001,0.001,0.001];
30
31
32 aux0 = [OBp0,OBa0,OCY0,OCp0,OCa0,fbm0,d0];
33 ModelParameter=fu_modelparameters12homeost([aux0,K_res,alfa]);
34
35 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
36
37 %%%%%%%%calculo de ash0 para tener la referencia%%%%
38
39 DeltaT=0.25;
40 DT=12;
41 MINP=10;
42 XKAPPA=0.0045;
43 VMPRIM=1.205357142857143e-01;
44 VMMAX=0.42; % 0.442;
45 DVR = OCa0*K_res*DeltaT/100;
46 DVF = DVR;            % Remodelling equilibrium is assumed
47 p   = 1-fbm0/100;     % Initial porosity in [0,1]
48
49 i=0;
50 while i<length(VFPREV)
51     VFPREV = fu_updateVFPREV12(p,DVR,DVF,VFPREV);
52     i=i+1;
```

```
53 end
54
55 % Then, the initial ash fraction is calculated
56
57 VM0 = 0;
58
59 for i=1:length(VFPREV)-1
60 VM0= VM0+VFPREV(i)*fu_calmineral12(i*DeltaT,DT,MINP,XKAPPA,VMPRIM,VMMAX)
       ;
61 end
62 VM0= VM0 + VFPREV(length(VFPREV))*VMMAX;
63
64 VM0=VM0/(1-p);
65
66 ash0 = 3.2*VM0/(3.2*VM0+1.1*3.0/7.0);
67 %%%%%%%%% fin calculo ash0%%%%%
68
69 %tablas que muestran la evolución de la convergencia
70 T1=zeros(0,12);
71
72 T2=zeros(0,10);
73 Cell_0 = [OBp0,OBa0,OCY0,OCp0,OCa0,fbm0,d0,ash0];
74
75 %bucle donde va variando la carga
76 for i=1:length(eps)
77 flag=0;
78     if Cell_0(6)>45
79         Cell_0(6)=51;
80     end
81 i = 1;
82 iter(i)=0;
83     while flag==0 %llama a runbonemodel para cada carga hasta que
          converge
84         ModelParameter = fu_modelparameters12homeost([Cell_0(1:(length(
              Cell_0)-1)),K_res,alfa]);
85         [CellsVector,E_mod] =run_bonemodel12_homeost(VFPREV0,
              ModelParameter,trat);
86         [n,m]=size(CellsVector);
87         for j=1:(m-2)
88             dif(i,j)=abs((CellsVector(n,j)-Cell_0(j))/Cell_0(j));
89             if (j==7) && (CellsVector(n,j)<1e-7) %daño % Original 1e-4
90                 if abs((CellsVector(n,j)-Cell_0(j)))<1e-8 % Original 1e-4
91                     conver(i,j)=1;
92                 else
93                     conver(i,j)=0;
94                 end
95             else
96                 if dif(i,j)<tol(j)
97                     conver(i,j)=1;
98                 else
```

```
99                      conver(i,j)=0;
100                 end
101             end
102             if isnan(CellsVector(n,j))
103                 Cell_0(1,j) = Cell_0(1,j);
104             else
105                 Cell_0(1,j) = CellsVector(n,j);
106             end
107         end
108         if sum(conver(i,:))==(m-2)
109             flag=1;
110         elseif iter(i)>iterfin
111             break
112         end
113         def(i)=eps/E_mod*1e-3;
114         iter(i)=iter(i)+1 ;
115         Result(i,:,:)=CellsVector;
116         %actualiza la tabla de evolución
117         format short
118        T1=[T1; num2cell([eps,iter(i),dif(i,:),CellsVector(n,7),def(i)])]
119        T2=[T2; num2cell([eps,iter(i),conver(i,:)])]
120 % %       [eps(i),iter(i),dif(i,:),CellsVector(n,7),def(i)]
121 % %       [eps(i),iter(i),conver(i,:)]
122         format long
123         %guarda los resultados parciales
124         Result2(i,1)=eps;
125         Result2(i,2)=def(i);
126         Result2(i,3)=Result(i,n,7);
127         Result2(i,4)=Result(i,n,8);
128         Result2(i,5)=Result(i,n,6);
129         save('Result.mat','Result')
130         save('Result2.mat','Result2')
131     end
132     if flag==0
133         break
134     end
135 %     hold off %impongo condición para que no superponga los gráficos
136 %     if i==1
137 %         h1 = plot(eps(1:i),Result(1:i,n,6),'ro','XDataSource','eps(1:i)
    ','YDataSource','Result(1:i,n,6)');%pinto los puntos
138 %         xlabel('Sigma (MPa)')
139 %         ylabel('fbm')
140 %         hold on
141 %         h2 = plot(eps(1:i),Result(1:i,n,6),'r','XDataSource','eps(1:i)
    ','YDataSource','Result(1:i,n,6)'); %pinto las líneas que conecta
    los pares
142 %         drawnow;
143 % %         axis([-3,10,-3,7]) %defino los ejes para que no maree al
    cambiar de ejes
144 %     else
```

```
% %       axis([-3,10,-3,7]) %defino los ejes para que no maree al
    cambiar de ejes
%       refreshdata(h1,'base')
%       refreshdata(h2,'base')
%       drawnow;
%    end
    Cell_0(1,6) = factor_cond_inic * CellsVector(n,6);
end

% save('result_52.mat','Result','eps','def')
```

4 Creación de las matrices de entrada y objetivo de la red neuronal

```
clc
clear all
close all

n_iterations = 200;

%Creación de las matrices
for iterative = 1:n_iterations
rng('shuffle');
T1 = randi([10 120]);
T3 = randi([10 120]);
T2 = randi([30 360]);
T4 = randi([30 360]);
T5 = randi([720 3600]);
trat = [T1,T3,T2,T4,T5];

T(iterative,:) = [T1,T3,T2,T4,T5];


deps = -0.2 + (0.2 + 0.2)*rand;
% deps = 0.1;
K_res = randi([100 300]);
% alfa = 0.5 + (-0.5 + 1)*rand;
% alfa = round(alfa,1);
alfa = 1;
% alfa = 0.95;
PRANKLpmo = randi([1000 4000]);
tiempo_enfermedad = randi([1 8]);
t_enf(iterative,:) = tiempo_enfermedad;

epsiterative(iterative,:) = deps;
Kresiter(iterative,:) = K_res;
Prankliter(iterative,:) = PRANKLpmo;

%Llama al modelo que devuelve las salidas para cada una de las entradas
```

```
run_bonemodel12(trat,deps,K_res,alfa,PRANKLpmo,tiempo_enfermedad);
%Carga los resultados del modelo
load('results52_sigma0.5T_RANKLPMO=2e3_sigmoidal=2years_ONSET=3
    years_zeta=0.2.mat')


ash1(iterative,:) = ash(tiempo_enfermedad*1440:end);
damage1(1,:) = CellsVector(:,7)';
damage2(iterative,:) = damage1(1,(tiempo_enfermedad*1440):end);
BDG_vector1(iterative,:) = BDG_vector;



ash2(iterative,1) = ash1(iterative,1);
damage3(iterative,1) = damage2(iterative,1);
damage1 = [];
for n = 1:(((length(ash1)-1)/720))
   ash2(iterative,n+1) = ash1(iterative,720*n);
   damage3(iterative,n+1) = damage2(iterative,720*n);
end



% save('matrices_salida.mat','ash2','damage2','T','BDG_vector1', ...
%     'epsiterative','Kresiter','alfaiter','Prankliter')


save('matrices_salida6.mat','T','ash2','damage3', ...
    'epsiterative','Kresiter','Prankliter','t_enf','BDG_vector1')
end
```

5 Unión de las matrices de entrada y objetivo de la red neuronal

```
%%% Extracción de matrices de la principal (2000, 1000, 500)
clear all
clc

load('matrices2243.mat')

%%% Matrices 2000
Matriz_ash2000 = Matriz_ash(1:2000,:);
Matriz_BDG2000 = Matriz_BDG(1:2000,:);
Matriz_damage2000 = Matriz_damage(1:2000,:);
Matriz_eps2000 = Matriz_eps(1:2000,:);
Matriz_Kres2000 = Matriz_Kres(1:2000,:);
Matriz_Prankl2000 = Matriz_Prankl(1:2000,:);
Matriz_T2000 = Matriz_T(1:2000,:);
Matriz_tenf2000 = Matriz_tenf(1:2000,:);
```

```
16
17 %%% Matrices 1000
18 Matriz_ash1000 = Matriz_ash(501:1500,:);
19 Matriz_BDG1000 = Matriz_BDG(501:1500,:);
20 Matriz_damage1000 = Matriz_damage(501:1500,:);
21 Matriz_eps1000 = Matriz_eps(501:1500,:);
22 Matriz_Kres1000 = Matriz_Kres(501:1500,:);
23 Matriz_Prankl1000 = Matriz_Prankl(501:1500,:);
24 Matriz_T1000 = Matriz_T(501:1500,:);
25 Matriz_tenf1000 = Matriz_tenf(501:1500,:);
26
27 %%% Matrices 500
28 Matriz_ash500 = Matriz_ash(1:500,:);
29 Matriz_BDG500 = Matriz_BDG(1:500,:);
30 Matriz_damage500 = Matriz_damage(1:500,:);
31 Matriz_eps500 = Matriz_eps(1:500,:);
32 Matriz_Kres500 = Matriz_Kres(1:500,:);
33 Matriz_Prankl500 = Matriz_Prankl(1:500,:);
34 Matriz_T500 = Matriz_T(1:500,:);
35 Matriz_tenf500 = Matriz_tenf(1:500,:);
36
37 save('matrices2000.mat','Matriz_ash2000','Matriz_BDG2000','
       Matriz_damage2000'...
38     ,'Matriz_eps2000','Matriz_Kres2000','Matriz_Prankl2000','
          Matriz_T2000',...
39     'Matriz_tenf2000')
40
41 save('matrices1000.mat','Matriz_ash1000','Matriz_BDG1000','
       Matriz_damage1000'...
42     ,'Matriz_eps1000','Matriz_Kres1000','Matriz_Prankl1000','
          Matriz_T1000',...
43     'Matriz_tenf1000')
44
45 save('matrices500.mat','Matriz_ash500','Matriz_BDG500','Matriz_damage500
       '...
46     ,'Matriz_eps500','Matriz_Kres500','Matriz_Prankl500','Matriz_T500'
          ,...
47     'Matriz_tenf500')
```

6 Configuración de la red neuronal

```
1 clc
2 clear all
3
4
5 capas_red = 'introduzca el numero de capas que quiere y neuronas por
       capa ';
6 hiddenSizes = input(capas_red);
```

```
% hiddenSizes= [20]; %empezar primero con 2 layers(una solo hidden) y lueg ir incrementando si la performance no es adecuada. Aumentar el numero de neuronas puede mejorar la precisión pero cuidado con tiempo de computación y con overfitting


% Elegir la función de entrenamiento
% 'trainlm' es generalmente más rápida.
% 'trainbr' tarda más tiempo pero es mejor para problemas complicados.
% 'trainscg' usa menos memoria.

funcion_entrenamiento = 'Escriba la funcion que quiere para el entrenamiento: trainlm, trainbr o trainscg ';
trainFcn = input(funcion_entrenamiento,'s');
% trainFcn='trainbr';


load('matrices2000.mat') % Matriz de entrada con las T(), Kres, Prankl_PMO, tiempo de enfermedad, eps y matrices de salida con BDG, ceniza y daño
load('matrices1000.mat') % Matriz de entrada con las T(), Kres, Prankl_PMO, tiempo de enfermedad, eps y matrices de salida con BDG, ceniza y daño
load('matrices500.mat') % Matriz de entrada con las T(), Kres, Prankl_PMO, tiempo de enfermedad, eps y matrices de salida con BDG, ceniza y daño
load('matrices_entrada.mat')

matrices_entrada = 'Escriba la matriz de entrada que quiera estudiar ';
Matriz_entrada = input(matrices_entrada);
matrices_salida = 'Escriba la matriz de salida que quiera estudiar ';
Matriz_salida = input(matrices_salida);

% net = feedforwardnet(); %por defecto la crea con una capa de 10
%por defecto se crean 2 capas, la hidden con 10, con funcion de activacion tansig en la hidden y purelin en la de salida y entrenamiento trainlm

netw = 'escriba la red que desea entrenar p.ej net2000ash ';
net = input(netw,'s');
% net = netr;
net = feedforwardnet(hiddenSizes,trainFcn);
net = configure(net, Matriz_entrada', Matriz_salida')
view(net)

tr = input('Escriba el tr que va a utilizar p.ej: tr2000ash ','s');
[net,tr] = train(net, Matriz_entrada', Matriz_salida')


save net
```

```
45 funcion = input('Escribe como quieres que se guarde la funcion de la red
       ','s')
46 genFunction(net,'funcion')
47
48 %performance funcion por defecto para feedforward es mean square error.
       Individual square errors can also be weighted
49 %Hay dos tipos de entrenamientos: incremental y batch. En el primero se
       adaptan los pesos despues de meter cada input. En el batch todos
      los inputs se aplican antes de que los pesos se actualicen. EN
      general, batch es más rápido y produce menos error
50
51
52 %por defecto divide el set de entrenamiento usando la función
      divederand.
53 %Training 70%, validation 15% y testing 15% (selección aleatoria)
54 %ejemplo para cambiar porcentajes
55 %  net.divideParam.trainRatio = 0.75;
56 %  net.divideParam.valRatio = 0.15;
57 %  net.divideParam.testRatio = 0.1;
```

7 Cálculo de errores de la red neuronal

7.1 BDG

```
1  clear all
2  clc
3
4  load('net2000BDG41.mat')
5  load('net1000BDG41.mat')
6  load('net500BDG41.mat')
7
8  entrada_test2000BDG = Matriz_entrada2000(tr2000BDG41.testInd,:);
9  entrada_test1000BDG = Matriz_entrada1000(tr1000BDG41.testInd,:);
10 entrada_test500BDG = Matriz_entrada500(tr500BDG41.testInd,:);
11
12 for ind = 1:length(tr2000BDG41.testInd)
13     Y2000BDG(:,ind) = funciontrat2000BDG41(entrada_test2000BDG(ind,:)');
14 end
15
16 for ind = 1:length(tr1000BDG41.testInd)
17     Y1000BDG(:,ind) = funciontrat1000BDG41(entrada_test1000BDG(ind,:)');
18 end
19
20 for ind = 1:length(tr500BDG41.testInd)
21     Y500BDG(:,ind) = funciontrat500BDG41(entrada_test500BDG(ind,:)');
22 end
23
24 Y2000_BDG = (Y2000BDG);
```

```
Y1000_BDG = (Y1000BDG);
Y500_BDG = (Y500BDG);

M2000_salida_BDG = Matriz_BDG2000(tr2000BDG41.testInd,:)';
M1000_salida_BDG = Matriz_BDG1000(tr1000BDG41.testInd,:)';
M500_salida_BDG = Matriz_BDG500(tr500BDG41.testInd,:)';

error2000_BDG = Y2000_BDG-M2000_salida_BDG;
error1000_BDG = Y1000_BDG-M1000_salida_BDG;
error500_BDG = Y500_BDG-M500_salida_BDG;

save('error2000BDG','error2000_BDG','error1000_BDG','error500_BDG')
```

7.2 Fracción de ceniza

```
clear all
clc

load('net2000ash.mat')
load('net1000ash.mat')
load('net500ash.mat')

load('net2000damage.mat')
load('net1000damage.mat')
load('net500damage.mat')

load('net2000BDG.mat')
load('net1000BDG.mat')
load('net500BDG.mat')

entrada_test2000ash = Matriz_entrada2000(tr2000ash.testInd,:);
entrada_test1000ash = Matriz_entrada1000(tr1000ash.testInd,:);
entrada_test500ash = Matriz_entrada500(tr500ash.testInd,:);

entrada_test2000damage = Matriz_entrada2000(tr2000damage.testInd,:);
entrada_test1000damage = Matriz_entrada1000(tr1000damage.testInd,:);
entrada_test500damage = Matriz_entrada500(tr500damage.testInd,:);

entrada_test2000BDG = Matriz_entrada2000(tr2000BDG.testInd,:);
entrada_test1000BDG = Matriz_entrada1000(tr1000BDG.testInd,:);
entrada_test500BDG = Matriz_entrada500(tr500BDG.testInd,:);


for ind = 1:length(tr2000ash.testInd)
    Y2000ash(:,ind) = funciontrat2000ash(entrada_test2000ash(ind,:)');
end

for ind = 1:length(tr1000ash.testInd)
    Y1000ash(:,ind) = funciontrat1000ash(entrada_test1000ash(ind,:)');
```

```
35 end
36
37 for ind = 1:length(tr500ash.testInd)
38     Y500ash(:,ind) = funciontrat500ash(entrada_test500ash(ind,:)');
39 end
40
41 for ind = 1:length(tr2000damage.testInd)
42     Y2000damage(:,ind) = funciontrat2000damage(entrada_test2000damage(
           ind,:)');
43 end
44
45 for ind = 1:length(tr1000damage.testInd)
46     Y1000damage(:,ind) = funciontrat1000damage(entrada_test1000damage(
           ind,:)');
47 end
48
49 for ind = 1:length(tr500damage.testInd)
50     Y500damage(:,ind) = funciontrat500damage(entrada_test500damage(ind
           ,:)');
51 end
52
53 for ind = 1:length(tr2000BDG.testInd)
54     Y2000BDG(:,ind) = funciontrat2000BDG(entrada_test2000BDG(ind,:)');
55 end
56
57 for ind = 1:length(tr1000BDG.testInd)
58     Y1000BDG(:,ind) = funciontrat1000BDG(entrada_test1000BDG(ind,:)');
59 end
60
61 for ind = 1:length(tr500BDG.testInd)
62     Y500BDG(:,ind) = funciontrat500BDG(entrada_test500BDG(ind,:)');
63 end
64
65 Y2000_ash = (Y2000ash-0.55)./0.2;
66 Y1000_ash = (Y1000ash-0.55)./0.2;
67 Y500_ash = (Y500ash-0.55)./0.2;
68
69 Y2000_damage = Y2000damage./0.1;
70 Y1000_damage = Y1000damage./0.1;
71 Y500_damage = Y500damage./0.1;
72
73 % Y2000_damage = Y2000damage;
74 % Y1000_damage = Y1000damage
75 % Y500_damage = Y500damage;
76
77 Y2000_BDG = abs(Y2000BDG)./20;
78 Y1000_BDG = abs(Y1000BDG)./20;
79 Y500_BDG = abs(Y500BDG)./20;
80
81 M2000_salida_ash = (Matriz_ash2000(tr2000ash.testInd,:)'-0.55)./0.2;
```

```
M1000_salida_ash = (Matriz_ash1000(tr1000ash.testInd,:)'-0.55)./0.2;
M500_salida_ash = (Matriz_ash500(tr500ash.testInd,:)'-0.55)./0.2;

M2000_salida_damage = Matriz_damage2000(tr2000damage.testInd,:)'./0.1;
M1000_salida_damage = Matriz_damage1000(tr1000damage.testInd,:)'./0.1;
M500_salida_damage = Matriz_damage500(tr500damage.testInd,:)'./0.1;

% M2000_salida_damage = Matriz_damage2000(tr2000damage.testInd,:)';
% M1000_salida_damage = Matriz_damage1000(tr1000damage.testInd,:)';
% M500_salida_damage = Matriz_damage500(tr500damage.testInd,:)';

M2000_salida_BDG = abs(Matriz_BDG2000(tr2000BDG.testInd,:)')./20;
M1000_salida_BDG = abs(Matriz_BDG1000(tr1000BDG.testInd,:)')./20;
M500_salida_BDG = abs(Matriz_BDG500(tr500BDG.testInd,:)')./20;

error2000_ahs = Y2000_ash-M2000_salida_ash;
error1000_ahs = Y1000_ash-M1000_salida_ash;
error500_ahs = Y500_ash-M500_salida_ash;

error2000_damage = Y2000_damage-M2000_salida_damage;
error1000_damage = Y1000_damage-M1000_salida_damage;
error500_damage = Y500_damage-M500_salida_damage;

error2000_BDG = Y2000_BDG-M2000_salida_BDG;
error1000_BDG = Y1000_BDG-M1000_salida_BDG;
error500_BDG = Y500_BDG-M500_salida_BDG;

error2000sal = [error2000_ahs; error2000_damage; error2000_BDG];
error1000sal = [error1000_ahs; error1000_damage; error1000_BDG];
error500sal = [error500_ahs; error500_damage; error500_BDG];

% error2000 = Y2000-Matriz_salida2000(tr2000.testInd,:)';
% error1000 = Y1000-Matriz_salida1000(tr1000.testInd,:)';
% error500 = Y500-Matriz_salida500(tr500.testInd,:)';
%
% error2000_ash = (abs(error2000(1:21,:))-0.55)./0.2;
% error2000_damage = abs(error2000(22:42,:))./0.1;
% error2000_BDG = abs(error2000(43:63,:))./20;
%
% error1000_ash = (error1000(1:21,:)-0.55)./0.2;
% error1000_damage = error1000(22:42,:)./0.1;
% error1000_BDG = abs(error1000(43:63,:))./20;
%
% error500_ash = (error500(1:21,:))./0.2;
% error500_damage = error500(22:42,:)./0.1;
% error500_BDG = abs(error500(43:63,:))./20;
%
% error2000 = [error2000_ash; error2000_damage; error2000_BDG];
% error1000 = [error1000_ash; error1000_damage; error1000_BDG];
% error500 = [error500_ash; error500_damage; error500_BDG];
```

```
132
133 save('error2000sal')
134 save('error1000sal')
135 save('error500sal')
```

7.3 Daño

```
1  clear all
2  clc
3
4  load('net2000damage31.mat')
5  load('net1000damage31.mat')
6  load('net500damage11.mat')
7
8  entrada_test2000damage = Matriz_entrada2000(tr2000damage31.testInd,:);
9  entrada_test1000damage = Matriz_entrada1000(tr1000damage31.testInd,:);
10 entrada_test500damage = Matriz_entrada500(tr500damage11.testInd,:);
11
12 for ind = 1:length(tr2000damage31.testInd)
13     Y2000damage(:,ind) = funciontrat2000damage31(entrada_test2000damage(
           ind,:)');
14 end
15
16 for ind = 1:length(tr1000damage31.testInd)
17     Y1000damage(:,ind) = funciontrat1000damage31(entrada_test1000damage(
           ind,:)');
18 end
19
20 for ind = 1:length(tr500damage11.testInd)
21     Y500damage(:,ind) = funciontrat500damage11(entrada_test500damage(ind
           ,:)');
22 end
23
24
25 Y2000_damage = Y2000damage;
26 Y1000_damage = Y1000damage;
27 Y500_damage = Y500damage;
28
29 M2000_salida_damage = Matriz_damage2000(tr2000damage31.testInd,:)';
30 M1000_salida_damage = Matriz_damage1000(tr1000damage31.testInd,:)';
31 M500_salida_damage = Matriz_damage500(tr500damage11.testInd,:)';
32
33 error2000_damage = Y2000_damage-M2000_salida_damage;
34 error1000_damage = Y1000_damage-M1000_salida_damage;
35 error500_damage = Y500_damage-M500_salida_damage;
36
37
38
```

```
save('error2000damage','error2000_damage','error1000_damage','
    error500_damage')
```

8 Dibujo de la media de errores de la red neuronal

8.1 BDG

```
clear all
clc


load('error2000BDG.mat')

error2000 = error2000_BDG';
error1000 = error1000_BDG';
error500 = error500_BDG';

cuantferror2000 = [];
cuantferror1000 = [];
cuantferror500 = [];

for col = 1:21
    cuantferror2000(col) = sum(abs(error2000(:,col)))/300;
    cuantferror1000(col) = sum(abs(error1000(:,col)))/150;
    cuantferror500(col) = sum(abs(error500(:,col)))/75;
end


hold on
title('Comparación de la media de los errores en el BDG')
plot((cuantferror2000),'r')
plot((cuantferror1000),'b')
plot((cuantferror500),'k')
xlabel('El BDG en los distintos instantes de tiempo')
ylabel('La media del error')
legend('2000','1000','500')
axis([0 25 0 3])

err2000 = sum(cuantferror2000)/21
err500 = sum(cuantferror500)/21
err1000 = sum(cuantferror1000)/21
```

8.2 Fracción de ceniza

```
clear all
clc

```

```

load('error2000ash.mat')

error2000 = error2000_ash';
error1000 = error1000_ash';
error500 = error500_ash';

cuantferror2000 = [];
cuantferror1000 = [];
cuantferror500 = [];

for col = 1:21
    cuantferror2000(col) = sum(abs(error2000(:,col)))/300;
    cuantferror1000(col) = sum(abs(error1000(:,col)))/150;
    cuantferror500(col) = sum(abs(error500(:,col)))/75;
end


hold on
title('Comparación de la media de los errores en la ceniza')
plot((cuantferror2000),'r')
plot((cuantferror1000),'b')
plot((cuantferror500),'k')
xlabel('la ceniza en los distintos instantes de tiempo')
ylabel('La media del error')
legend('2000','1000','500')
axis([0 25 0 0.005])

err2000 = sum(cuantferror2000)/21
err500 = sum(cuantferror500)/21
err1000 = sum(cuantferror1000)/21
```

8.3 Daño

```
clear all
clc


load('error2000damage.mat')

error2000 = error2000_damage';
error1000 = error1000_damage';
error500 = error500_damage';

cuantferror2000 = [];
cuantferror1000 = [];
cuantferror500 = [];

for col = 1:21
```

```
    cuantferror2000(col) = sum(abs(error2000(:,col)))/300;
    cuantferror1000(col) = sum(abs(error1000(:,col)))/150;
    cuantferror500(col) = sum(abs(error500(:,col)))/75;
end


hold on
title('Comparación de la media de los errores en el daño')
plot((cuantferror2000),'r')
plot((cuantferror1000),'b')
plot((cuantferror500),'k')
xlabel('El daño en los distintos instantes de tiempo')
ylabel('La media del error')
legend('2000','1000','500')
axis([0 25 0 0.05])

err2000 = sum(cuantferror2000)/21
err500 = sum(cuantferror500)/21
err1000 = sum(cuantferror1000)/21
```

9 Algoritmo genético

```
clear all
clc

Kres = input('Kres_usuario '); %Para que el usuario escriba el Kres del
    paciente
eps = input('eps_usuario '); %Para que el usuario escriba el incremento
    de ejercicio físico del paciente
t_enfermedad = input('tiempo_enfermedad '); %Para que el usuario
   escriba el tiempo de enfermedad que lleva el paciente
Prankl = input('Prankl_usuario '); %Para que el usuario escriba el P
   del paciente

z0=[40,40,80,80,800]; %valores iniciales de las constantes incógnita.
   Hacen falta por si el algoritmo genetico falla (ver más adelante)

%restricciones y condiciones para las variables a optimizar
limite_inferior = [10, 10, 30, 30, 720]; %límite inferior para las
   variables del problema de optimización T1 T3 T2 T4 T5
limite_superior = [120, 120, 360, 360, 3600]; %límite superior para las
    variables del problema de optimización
Ares=[]; %matriz de restricciones, si hubiera restricciones entre
   variables
bres=[]; %matriz de restricciones, si hubiera restricciones entre
   variables

%Iteraciones con algoritmo genético
```

```
18
19 for o = 1:15 %número de iteraciones que queramos hacer para quedarnos
       con la mejor solución de todas
20
21       rng('shuffle');
22       T1 = randi([10 120]);
23       T3 = randi([10 120]);
24       T2 = randi([30 360]);
25       T4 = randi([30 360]);
26       T5 = randi([720 3600]);
27
28  PARAMETROS = [T1, T3, T2, T4, T5]';
29
30       optionsGA = gaoptimset('PopulationSize',200,'Generations',250,'
             TolFun',1e-8,'PopInitRange',[0;1]); %opciones del algoritmo gen
             ético
31       ecuacionesga = @(PARAMETROS) fecuacionesga12(PARAMETROS,[Kres;eps;
             t_enfermedad;Prankl]); %ecuaciones a optimizar, pasandole los
             parametros que sean
32       [z,fvalga,exitflag] = ga(ecuacionesga,5,Ares,bres,[],[],
             limite_inferior,limite_superior,[],optionsGA); %llamada al
             algoritmo genético (ECUACIONES,Nº INCOGNITAS, RESTRICCIONES_A,
             RESTRICCIONES_b,...,LOWER BOUND, UPPER BOUND,,OPCIONES)
33
34 end
35
36 salida_ga_ash = funciontrat2000ash4([z(1);z(2);z(3);z(4);z(5);Kres;eps;
       t_enfermedad;Prankl]);
37 salida_ga_BDG = funciontrat2000BDG41([z(1);z(2);z(3);z(4);z(5);Kres;eps;
       t_enfermedad;Prankl]);
38 salida_ga_damage = funciontrat1000damage31([z(1);z(2);z(3);z(4);z(5);
       Kres;eps;t_enfermedad;Prankl]);
39
40
41 X1 = ['La dosis del tratamiento 1 es ', num2str(z(1)), ' mg ', 'en
       intervalos de ', num2str(z(2)), ' dias'];
42 X2 = ['La dosis del tratamiento 2 es ', num2str(z(3)), ' mg ', 'en
       intervalos de ', num2str(z(4)), ' dias'];
43 X3 = ['El tiempo de cambio entre un tratamiento y otro es ', num2str(z
       (5)), ' dias'];
44
45
46 disp('El tratamiento optimo para el paciente es: ');
47 disp(X1);
48 disp(X2);
49 disp(X3);
```