Trabajo de Fin de Grado
Grado en Ingeniería Aeroespacial

Desarrollo de una aplicación móvil para identificar síntomas de enfermedades neurológicas

Autor: Jesús Jurado Ruiz

Tutor: Pedro Galvín Barrera

Dpto. Mecánica de Medios Continuos y Teoría de Estructuras
Escuela Técnica Superior de Ingeniería
Universidad de Sevilla

Sevilla, 2022

Trabajo de Fin de Grado
Grado en Ingeniería Aeroespacial

Desarrollo de una aplicación móvil para identificar síntomas de enfermedades neurológicas

Autor:
Jesús Jurado Ruiz

Tutor:
Pedro Galvín Barrera
Catedrático

Dpto. Mecánica de Medios Continuos y Teoría de Estructuras
Escuela Técnica Superior de Ingeniería
Universidad de Sevilla

Sevilla, 2022

Trabajo de Fin de Grado: Desarrollo de una aplicación móvil para identificar síntomas de enfermedades neurológicas

Autor: Jesús Jurado Ruiz
Tutor: Pedro Galvín Barrera

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 realizar un agradecimiento general a todas las personas que me han acompañado durante el transcurso del grado dado que sin ellos no podría haberlo logrado.

A Pedro, mi tutor, por su gran profesionalidad, su constante disponibilidad y su ayuda durante el desarrollo del presente proyecto.

A mi familia por su confianza, su tiempo y su apoyo incondicional durante todos estos años.

A mis amigos y compañeros de clase por los momentos compartidos.

A Zahira, porque sin su apoyo todo hubiera sido mucho más complicado.

Jesús Jurado Ruiz
Sevilla, 2022

Resumen

El diagnóstico de la naturaleza de los temblores que se manifiestan en las personas no es una tarea sencilla para los médicos debido a la gran cantidad de posibles causas de los mismos. Además, existen numerosos factores que influyen en la magnitud de dichos temblores tales como la edad, el peso, la medicación que toma una determinada persona, el miembro sobre el que se realiza la medición (brazo, pierna, cabeza, etc.) o la posición del miembro en cuestión a la hora de llevar a cabo la medición (puesto que en algunas posiciones pueden darse y en otras no). No obstante, la aparición de nuevas tecnologías ha ayudado enormemente a facilitar esta tarea.

Este proyecto se centra precisamente en el aprovechamiento de dichas tecnologías por medio del desarrollo de una aplicación móvil que permite registrar los temblores de los pacientes a través del empleo del acelerómetro de un teléfono móvil. Esta herramienta permite a los médicos analizar los temblores de cada uno de sus pacientes gracias a las representaciones gráficas de los mismos, especialmente las asociadas al dominio de la frecuencia, puesto que algunos de los parámetros más relevantes en este tipo de análisis son la amplitud y la frecuencia a la que se da el pico máximo del temblor.

Una vez registrados los temblores de varios pacientes, el objetivo es ser capaces de predecir tanto la amplitud como la frecuencia a la que sucede el pico máximo del temblor en el dominio de la frecuencia en función de la edad, el peso, el sexo y la medicación que toma un determinado paciente sin tener que llevar a cabo ninguna medición física. Para resolver esta tarea se ha hecho uso de una red neuronal artificial que, tras pasar por un proceso de entrenamiento basado en datos registrados de pacientes anteriores, es capaz de predecir las características del pico mencionado previamente.

Esta herramienta permite a un médico comparar las características que debería tener el pico del temblor de un paciente según lo predicho por la red neuronal con lo registrado en realidad por medio del acelerómetro de un teléfono móvil y poder así sacar conclusiones acerca de la naturaleza de dicho temblor.

Abstract

Diagnosing the nature of tremors in people is not a simple task for physicians due to the large number of possible causes of tremors. In addition, there are many factors that influence the magnitude of tremors, such as age, weight, medication taken, the type of the limb on which the measurement is taken (arm, leg, head, etc.) or the position of the limb in question when taking the measurement (since in some positions they may occur and in others they may not). However, the emergence of new technologies has greatly facilitated this task.

This project focuses precisely on taking advantage of these technologies by developing a mobile application that allows patients'tremors to be recorded using a mobile phone's accelerometer. This tool allows doctors to analyse the tremors of each of their patients using the accelerometer of a mobile phone. This tool allows doctors to analyse the tremors of their patients thanks to the graphical representations of the tremors, especially those associated with the frequency domain, since some of the most relevant parameters in this type of analysis are the amplitude and the frequency at which the maximum peak of the tremor occurs.

Once the tremors of several patients have been recorded, the aim is to be able to predict both the amplitude and the frequency at which the maximum peak tremor occurs in the frequency domain as a function of age, weight, gender and the medication taken by a given patient without having to make any physical measurements. To solve this task, an artificial neural network has been used which, after undergoing a training process based on data recorded from previous patients, is able to predict the characteristics of the aforementioned peak.

This tool allows the doctor to compare the characteristics that a patient's tremor peak predicted by the neural network should have with what is actually recorded by the accelerometer of a mobile phone, and thus be able to draw conclusions about the nature of the tremor.

Índice Abreviado

Índice

1 Introducción

1.1 El problema de la detección de enfermedades relacionadas con los temblores

Un temblor es una oscilación rítmica que se produce en una determinada parte del cuerpo de una persona (extremidades superiores, cabeza, extremidades inferiores, etc). Se considera como uno de los movimientos involuntarios más comunes realizados por el ser humano. Sin embargo, en ocasiones las características de estos temblores pueden indicar la presencia de anomalías relacionadas con enfermedades de naturaleza neurológica. Por este motivo, surge la necesidad de llevar a cabo un estudio de los mismos.

El diagnóstico de la naturaleza de los temblores que se manifiestan en las personas no es una tarea sencilla para los médicos debido a la gran cantidad de posibles causas de los mismos. Además, existen numerosos factores que influyen en la magnitud de dichos temblores tales como la edad, el peso, la medicación que toma una determinada persona, el miembro sobre el que se realiza la medición (brazo, pierna, cabeza, etc.) o la posición del miembro en cuestión a la hora de llevar a cabo la medición (dado que en algunas posiciones pueden darse y en otras no). No obstante, la aparición de nuevas tecnologías ha ayudado enormemente a facilitar esta tarea.

Este proyecto se centra precisamente en el aprovechamiento de dichas tecnologías por medio del desarrollo de una aplicación móvil que permite registrar los temblores de los pacientes a través del empleo del acelerómetro de un teléfono móvil. Esta herramienta permite a los médicos analizar los temblores de cada uno de sus pacientes gracias a las representaciones gráficas de los mismos, especialmente las asociadas al dominio de la frecuencia, puesto que algunos de los parámetros más relevantes en este tipo de análisis son la amplitud y la frecuencia a la que se da el pico máximo del temblor.

Una vez registrados los temblores de varios pacientes, el objetivo es ser capaces de predecir tanto la amplitud como la frecuencia a la que sucede el pico máximo del temblor en el dominio de la frecuencia en función de la edad, el peso, el sexo y la medicación que toma un determinado paciente sin tener que llevar a cabo ninguna medición física. Para resolver esta tarea se ha hecho uso de una red neuronal artificial que, tras pasar por un proceso de entrenamiento basado en datos registrados de pacientes anteriores, es capaz de predecir las características del pico mencionado previamente.

Esta herramienta permite a un médico comparar las características que debería tener el pico del temblor de un paciente según lo predicho por la red neuronal con lo registrado en realidad por medio del acelerómetro de un teléfono móvil y poder así sacar conclusiones acerca de la naturaleza de dicho temblor.

Para la elaboración de dicha aplicación se ha seguido un desarrollo en paralelo en dos lenguajes de programación: `MATLAB` y `Python`. La finalidad de este doble desarrollo es tener disponible la aplicación en un lenguaje de programación privativo como es `MATLAB` y en uno libre como es `Python` para que sea accesible a todos los interesados.

1.2 Estado del arte

El conocimiento de la naturaleza de los diferentes tipos de temblores manifestados por los seres humanos es de vital importancia para tratar de abordar a tiempo posibles problemas causados por su existencia. En la línea de alcanzar dicho conocimiento se han llevado a cabo estudios como los realizados por Kailash et al. [1] y Louis [2], que tratan de caracterizar con el mayor nivel de detalle posible los distintos tipos de temblores basándose en parámetros como la frecuencia a la que aparecen, partes del cuerpo en las que suceden o la posición de las partes del cuerpo sobre las que ocurren.

El interés por la medición de los temblores manifestados por las personas ha ido creciendo a medida que se ha desarrollado la tecnología disponible para su realización. En primera instancia, las mediciones de este tipo requerían el empleo de equipos complejos y específicos que no se encontraban al alcance de cualquiera y esto hacía que existiera cierta dificultad a la hora de llevar a cabo una monitorización de los pacientes. Sin embargo, con el paso del tiempo, la existencia de herramientas como los smartphones han hecho que dicha tarea sea posible sin necesidad de emplear dispositivos con tanta complejidad. Han sido muchos los estudios realizados con el fin de aprovechar los smartphones para llevar a cabo un registro de los temblores de los pacientes entre los que destacan los realizados por García-Magariño et al. [3], donde se desarrolla una aplicación móvil para detectar temblores producidos en las manos de las personas, y Barrantes et al. [4], donde se emplea el acelerómetro de un smartphone para llevar a cabo un diagnóstico del tipo de temblor sufrido por un paciente. Sin embargo, los smartphones no son las únicas herramientas que se emplean en la actualidad para monitorizar dichos temblores tal como se puede comprobar en Brindha et al. [5], donde se emplean un sensor IMU de 6 ejes que sirve para medir tanto velocidades angulares como aceleraciones lineales y una placa electrónica de tipo NodeMCU que se utiliza para convertir las señales analógicas recogidas en el sensor previamente comentado a digitales.

Dentro de este marco de desarrollo, el presente proyecto pretende realizar un diagnóstico de la naturaleza del temblor sufrido por un determinado paciente diferenciando entre los temblores producidos a causa de la toma de un tipo de medicación en concreto y los temblores consecuencia de la existencia de una enfermedad de tipo neurológico por medio del desarrollo de una aplicación móvil.

1.3 Herramientas móviles disponibles para el desarrollo del proyecto

En esta sección se van a analizar las herramientas disponibles para ejecutar código tanto de `MATLAB` como de `Python` en un teléfono móvil dado que son los dos lenguajes de programación que se van a emplear en la elaboración de la aplicación.

1.3.1 `MATLAB` Mobile

`MATLAB` Mobile es una herramienta que permite a los usuarios con cuenta en MathWorks acceder a las ventajas que ofrece el lenguaje de programación `MATLAB` a través de un dispositivo móvil. Algunas de las tareas que permite realizar esta aplicación móvil son [6]:

- Acceder a `MATLAB` desde la línea de comandos.
- Ver, ejecutar, editar y crear archivos desde el editor.
- Adquirir datos de los sensores del dispositivo.
- Almacenar sus archivos y datos en `MATLAB` Drive.

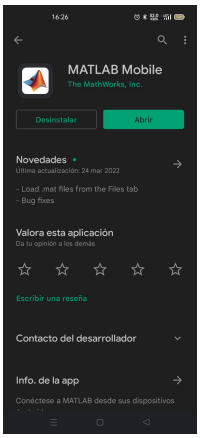

Figura 1.1 `MATLAB` Mobile en Play Store.

Sin embargo, también ofrece ciertas limitaciones con respecto a la versión para ordenador tales como [6]:

- No poder emplear algunas aplicaciones de `MATLAB` como Curve Fitting.
- Imposibilidad de crear aplicaciones con App Designer.
- Incapacidad de interactuar con figuras 3D.
- No ser capaz de abrir o crear modelos empleando el entorno gráfico de Simulink.
- Al tratarse de un lenguaje de programación privativo no es accesible para todos los interesados.

Esto dificulta la implementación de algunas tareas, especialmente en el desarrollo de la red neuronal, como se verá más adelante.

1.3.2 Qpython 3L - Python for Android

Qpython 3L es un motor de `Python` para Android que permite acceder a las ventajas que ofrece el lenguaje de programación `Python` a través de un teléfono móvil. Algunas de las principales ventajas que ofrece esta aplicación móvil son las siguientes [7]:

- Capacidad de ejecutar programas de `Python` sin conexión a internet.
- Tiene incorporadas características SL4A que, entre otras cosas, permiten acceder a los sensores del teléfono.

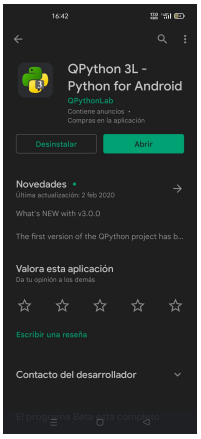

Figura 1.2 QPython 3L en Play Store.

No obstante, esta aplicación lleva un tiempo sin actualizarse y esto provoca ciertos problemas a la hora de la instalación de ciertas librerías básicas y fundamentales para el desarrollo del proyecto como son matplotlib

o numpy, entre otras. Por tanto, únicamente se ha empleado para la toma de datos de los sensores del teléfono a través del módulo SL4A que trae incorporado.

1.3.3 Pydroid3 - IDE for Python 3

Pydroid3 es un entorno de desarrollo integrado de `Python` 3 que permite acceder a las ventajas que ofrece el lenguaje de programación `Python` a través de un teléfono móvil [8].

La principal diferencia con Qpython 3L es que, esta aplicación sí que se encuentra actualizada y esto permite el empleo de las librerías actualizadas de Python previamente mencionadas y tan necesarias en el desarrollo del proyecto. Algunas de las principales ventajas que ofrece esta aplicación móvil son [8]:

- Capacidad de ejecutar programas de `Python` sin conexión a internet.
- Posibilidad de instalación rápida y sencilla de las librerías de Python más actuales.

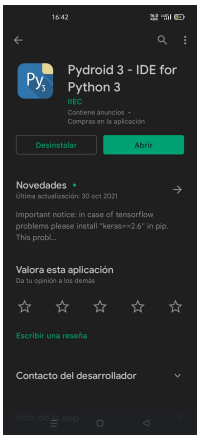

Figura 1.3 Pydroid 3 en Play Store.

Sin embargo, esta aplicación no lleva incorporado el módulo de acceso a los sensores del teléfono que lleva incorporado QPython 3L y, por tanto, no nos permite acceder a los mismos. Por este motivo, para conseguir desarrollar la aplicación móvil del proyecto en cuestión se han tenido que emplear ambas herramientas en conjunto. Por un lado, se ha empleado Qpython 3L para la medida de los temblores por medio del acelerómetro del teléfono y, por el otro, Pydroid3 para el postprocesado de los datos registrados.

1.4 Objetivos del proyecto

En primer lugar, cabe destacar que la elaboración de la aplicación se ha llevado a cabo simultáneamente en `MATLAB` y en `Python`. Por tanto, cada paso de los que se menciona a continuación se ha realizado para los dos lenguajes previamente mencionados. Además, es de vital importancia mencionar que para la resolución de cada una de las tareas demandadas por el proyecto en ambos lenguajes de programación se han desarrollado una serie de ficheros que irán siendo introducidos a lo largo de la presente memoria comentando tanto su denominación como la tarea que desempeñan.

El primer paso en el desarrollo del proyecto es la elaboración de un código que se encargue de la medición de los temblores de los pacientes a través del acelerómetro del teléfono móvil. A continuación, se almacenarán dichas mediciones para poder ser representadas y extraer información clave de ellas. Posteriormente, debido a la falta de mediciones de temblores procedentes de pacientes reales en el momento de la realización del presente proyecto, se generarán temblores sintéticos partiendo de un temblor tomado de un paciente real (del autor del presente proyecto). Cabe destacar que este paso únicamente será necesario mientras no se tenga la cantidad suficiente de registros de temblores obtenidos de mediciones reales. Finalmente, se utilizarán

esos temblores generados sintéticamente para llevar a cabo el entrenamiento de una red neuronal artificial que, una vez entrenada, será capaz de predecir las características del pico máximo del temblor que tendría un determinado paciente según su edad, peso, sexo y las medicaciones que tome. Tras llevar a cabo todas estas tareas se podrá realizar un diagnóstico de la naturaleza del temblor sufrido por un determinado paciente diferenciando entre los temblores producidos a causa de la toma de un tipo de medicación en concreto y los temblores consecuencia de la existencia de una enfermedad de tipo neurológico mediante la comparación del pico máximo del temblor predicho por la red con el medido a través del acelerómetro del smartphone. En caso de que ambos coincidan, el temblor estaría causado por la toma de algún tipo de medicación. Sin embargo, si hay disparidad podría ser causado por algún tipo de enfermedad neurológica.

Cabe destacar que, cuando se tengan el número suficiente de mediciones de temblores procedentes de pacientes reales, habría que modificar el conjunto de entrenamiento y volver a entrenar la red con el fin de obtener predicciones más precisas y realistas.

1.5 Organización de la memoria

En primer lugar, hay un capítulo de introducción en el que se trata brevemente el problema que se va a resolver durante el desarrollo del proyecto y se comenta el estado del arte del mismo, así como las herramientas disponibles actualmente para llevarlo a cabo. A continuación, se comentan tanto los métodos como las soluciones adoptadas durante el desarrollo del proyecto. Posteriormente, se explica la puesta en funcionamiento de las aplicaciones en ambos lenguajes de programación, es decir, en `MATLAB` y en `Python`. Después, existe otro capítulo para comentar los resultados obtenidos. Finalmente, se comentan las conclusiones y los desarrollos futuros del presente proyecto. Además, existen unos apéndices en los que se explican con detalle ciertas tareas que hay que realizar.

2 Metodología y soluciones adoptadas durante el desarrollo del proyecto

El presente capítulo comienza con la presentación del tipo de red neuronal escogido para llevar a cabo la resolución del problema de regresión presente en el proyecto. A continuación, dado que no se dispone de una cantidad suficiente de medidas de temblores procedentes de pacientes reales, se explica la manera en que se han generado datos sintéticos con el fin de completar el proceso de entrenamiento de dicha red neuronal. Finalmente, se comentan las soluciones adoptadas para el desarrollo de la aplicación tanto en `MATLAB` como en `Python`.

2.1 Resolución del problema de regresión

En el presente proyecto se tiene la necesidad de predecir las características del pico máximo asociado al temblor de un paciente (frecuencia a la que aparece el pico máximo y amplitud de dicho pico) en función de la información que se tiene de la persona (edad, sexo, peso y medicaciones que toma). Básicamente, dando los datos del paciente (edad, sexo, peso y medicaciones que toma) como entrada se espera obtener las características del pico máximo del temblor que debería tener dicho paciente como salida. Este tipo de problema se conoce como problema de regresión o predicción. Históricamente, este tipo de problemas se han abordado con modelos estadísticos de regresión. Sin embargo, en los últimos años se ha extendido el uso de las redes neuronales para resolver esta problemática dado que ofrecen un mejor rendimiento. Por esta razón, se resolverá el problema previamente mencionado usando una red neuronal.

2.1.1 Concepto de red neuronal

Una red neuronal es un modelo simplificado que emula el modo en que el cerebro humano procesa la información. Funciona simultaneando un número elevado de unidades de procesamiento interconectadas que parecen versiones abstractas de neuronas [9].

Las unidades de procesamiento se organizan en capas. Hay tres partes normalmente en una red neuronal : una capa de entrada, con unidades que representan los campos de entrada; una o varias capas ocultas; y una capa de salida, con una unidad o unidades que representa el campo o los campos de destino. Las unidades se conectan con fuerzas de conexión variables (o ponderaciones). Los datos de entrada se presentan en la primera capa, y los valores se propagan desde cada neurona hasta cada neurona de la capa siguiente. Al final, se envía un resultado desde la capa de salida [9].

La red aprende examinando los registros individuales, generando una predicción para cada registro y realizando ajustes a las ponderaciones cuando realiza una predicción incorrecta. Este proceso se repite muchas veces y la red sigue mejorando sus predicciones hasta haber alcanzado uno o varios criterios de parada [9].

Al principio, todas las ponderaciones son aleatorias y las respuestas que resultan de la red son, posiblemente, disparatadas. La red aprende a través del entrenamiento. Continuamente se presentan a la red ejemplos para los que se conoce el resultado, y las respuestas que proporciona se comparan con los resultados conocidos. La información procedente de esta comparación se pasa hacia atrás a través de la red, cambiando las ponderaciones

gradualmente. A medida que progresa el entrenamiento, la red se va haciendo cada vez más precisa en la replicación de resultados conocidos. Una vez entrenada, la red se puede aplicar a casos futuros en los que se desconoce el resultado [9].

2.1.2 Ventajas e inconvenientes del uso de redes neuronales en un problema de regresión

Al igual que sucede con todas las posibles soluciones de un problema, el uso de redes neuronales para abordar un problema de regresión presenta sus pros y sus contras. Algunas de las ventajas que ofrece este tipo de solución son su gran capacidad de aprendizaje basándose en los datos proporcionados durante la fase de entrenamiento y su gran velocidad de aprendizaje. Sin embargo, como ocurre con todas las herramientas, las redes neuronales también presentan algunos inconvenientes en la resolución de este tipo de problemas como son la complejidad a la hora de determinar la estructura de dicha red (basada en procedimientos empíricos de prueba y error) o la dificultad a la hora de tratar de identificar la razón por la que la red devuelve unos resultados y no otros (estructura de "caja negra").

2.1.3 Tipo de red neuronal escogida para la resolución del problema

La tipología de red neuronal que se ha seleccionado para resolver el problema de regresión ya comentado ha sido el perceptrón multicapa. El principal motivo de dicha elección es la gran versatilidad que ofrece esta tipología de red neuronal para adaptarse a datos que siguen modelos no lineales, aunque algunas de sus ventajas adicionales son su gran velocidad de aprendizaje y su robustez ante la existencia de ruido en el conjunto de datos suministrado. Sin embargo, sigue siendo de vital importancia conocer las limitaciones que ofrece esta tipología de redes neuronales. Entre las más importantes se encuentran la dificultad asociada al ajuste de una serie de parámetros tales como el número de capas ocultas, el número de neuronas o las funciones de activación, la variabilidad en la precisión del modelo en función de los valores de los pesos iniciales y su incapacidad de extrapolar adecuadamente en caso de entrenarse mal o de forma insuficiente.

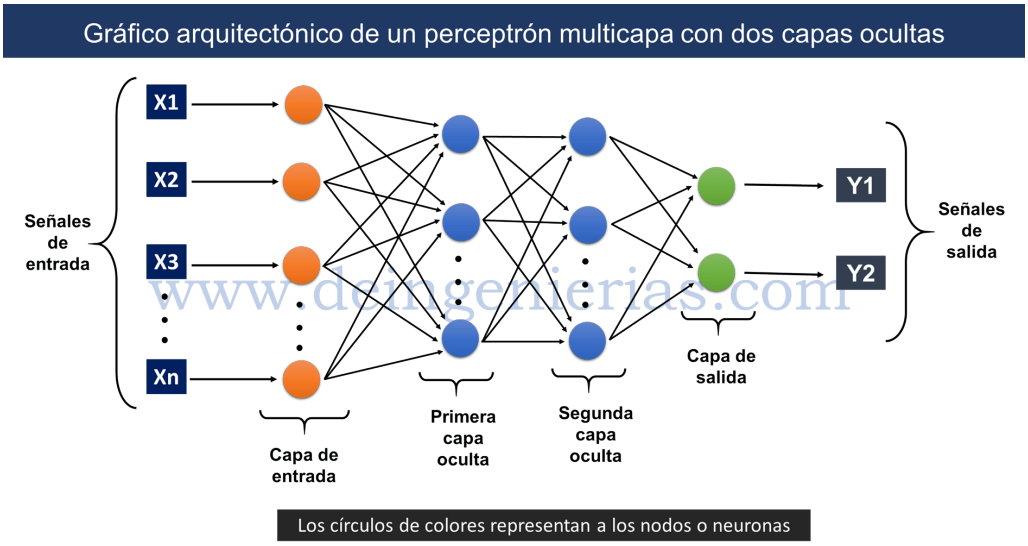

Figura 2.1 Ejemplo de un perceptrón multicapa [10].

Se procederá ahora a comentar algunos de los parámetros más importantes que hay que ajustar en este tipo de redes neuronales. En primer lugar, será necesario realizar una división de los datos empleados para los procesos de entrenamiento y validación. La cantidad de datos empleada para cada uno de los procesos previamente mencionados ha de escogerse con el fin de evitar los fenómenos tanto de underfitting (rendimiento deficiente de la red tanto con los datos de entrenamiento como con los datos de validación) como de overfitting (buen rendimiento de la red con los datos de entrenamiento, pero malo con los datos de validación).

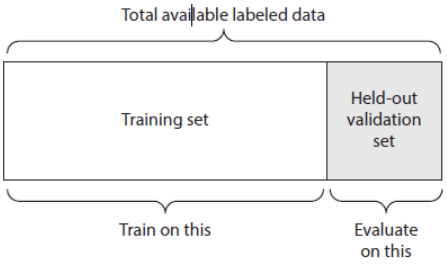

Figura 2.2 División del conjunto de datos [11].

A continuación, será necesario establecer el número de épocas máximo. Este número hace referencia a las veces que la red ve cada valor de entrenamiento en el proceso de entrenamiento. Es muy importante para evitar los fenómenos de underfitting y overfitting anteriormente comentados. Posteriormente, se establecerá el número de capas ocultas. Este número puede aumentar considerablemente según la complejidad de los datos que se van a tratar e influye enormemente en la velocidad de cálculo. Por tanto, tendrá que ser ajustado de tal forma que obtenga los mejores resultados posibles en un tiempo aceptable. Además, cada capa oculta presenta un número de neuronas que deberá ser establecido. Este número será escogido con el objetivo de conseguir evitar los fenómenos tanto de underfitting como de overfitting previamente comentados.

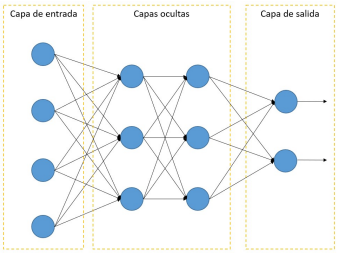

Figura 2.3 Capas ocultas [12].

Por último, otro de los parámetros a establecer es la función de activación. Se trata de una función que determina en qué medida se activa cada neurona dependiendo de su relevancia en la red. Es un parámetro muy importante durante el proceso de entrenamiento de la red neuronal y su correcta elección es clave para la obtención de resultados óptimos. Algunas de las más destacadas son: identidad, sigmoide, tangente hiperbólica y relu.

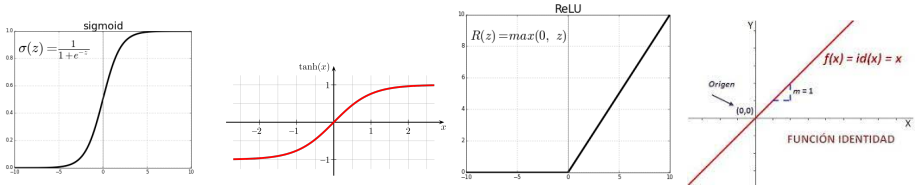

Figura 2.4 Funciones de activación [13, 14, 15, 16] .

2.2 Modelo de generación de datos sintéticos a partir de datos reales

Debido a la falta de registros de temblores procedentes de pacientes reales para llevar a cabo el proceso de entrenamiento de la red neuronal desarrollada en el presente proyecto se decidió generar temblores sintéticos a partir de los temblores procedentes de un paciente real (el autor del presente proyecto). Por tanto, fue necesario llevar a cabo un registro previo de mi temblor para tomarlo como punto de partida.

Para ello se ha decidido generar datos de 1536 pacientes tanto hombres como mujeres con edades comprendidas entre los 20 y los 100 años y con pesos comprendidos entre los 40 y los 160 kg.

El proceso seguido para llevarlo a cabo es el siguiente:

En primer lugar, se añade al temblor real de partida un ruido blanco con una amplitud asociada al 5 % del pico máximo del temblor en cada uno de los ejes. Con la adición de este tipo de ruido se consigue recrear la incertidumbre existente entre los temblores de distintos pacientes en la realidad.

A continuación, se añade el efecto producido por la toma de una medicación determinada. La toma de un medicamento en cuestión provoca la aparición de picos en el temblor a ciertas frecuencias dependiendo del medicamento consumido. Este efecto se ha tratado de conseguir en los datos sintéticos mediante la adición al temblor real en el dominio del tiempo de funciones senoidales con una amplitud correspondiente a un 10 % del pico máximo del temblor en cada uno de los ejes y con una frecuencia equivalente a la frecuencia del pico provocado por la medicación en cuestión. Para los datos sintéticos generados en este caso en particular se ha decidido que la medicación 1 produce un pico en torno a 5 Hz, la medicación 2 en torno a 7 Hz y la medicación 3 en torno a 9 Hz.

Por tanto, el efecto de cada una de las medicaciones se corresponde con los siguientes términos:

$$Medicacion\_1 : 0.1 \cdot Acmax \cdot \sin(2 \cdot \pi \cdot 5 \cdot t) \tag{2.1}$$

$$Medicacion\_2 : 0.1 \cdot Acmax \cdot \sin(2 \cdot \pi \cdot 7 \cdot t) \tag{2.2}$$

$$Medicacion\_3 : 0.1 \cdot Acmax \cdot \sin(2 \cdot \pi \cdot 9 \cdot t) \tag{2.3}$$

Por último se modelizará el crecimiento de la amplitud de los picos de los temblores con la edad y el peso por medio del empleo de una constante de ponderación que crece cuando crecen el peso, la edad o ambas simultáneamente. La constante es la siguiente:

$$k_{ep} = 1 + 0.1 \cdot k_e + 0.1 \cdot k_p \tag{2.4}$$

Donde:

$$k_e = \frac{edad - edadmin}{edadmax - edadmin} \tag{2.5}$$

$$k_p = \frac{peso - pesomin}{pesomax - pesomin} \tag{2.6}$$

Las variables que aparecen en las constantes de ponderación tienen el siguiente significado:

- **edad**: hace referencia a la edad del paciente que se va a generar.
- **edadmin**: hace referencia a la edad mínima del conjunto de pacientes que va a generarse. En el caso particular del presente proyecto, dado que se van a generar pacientes que van de los 20 a los 100 años, el valor de edadmin será 20 años.
- **edadmax**: hace referencia a la edad máxima del conjunto de pacientes que va a generarse. En el caso particular del presente proyecto, dado que se van a generar pacientes que van de los 20 a los 100 años, el valor de edadmax será 100 años.
- **peso**: hace referencia al peso del paciente que se va a generar.
- **pesomin**: hace referencia al peso mínimo del conjunto de pacientes que va a generarse. En el caso particular del presente proyecto, dado que se van a generar pacientes que pesan entre los 40 a los 160 kg, el valor de pesomin será de 40 kg.
- **pesomax**: hace referencia al peso máximo del conjunto de pacientes que va a generarse. En el caso particular del presente proyecto, dado que se van a generar pacientes que pesan entre los 40 a los 160 kg, el valor de pesomax será de 160 kg.

2.2.1 Implementación del modelo de generación de datos sintéticos en MATLAB

Todo lo anterior se consigue en MATLAB con las siguientes líneas de código:

```
ke=(edad-edadmin)/(edadmax-edadmin); %Cte de ponderación de la edad
kp=(peso-pesomin)/(pesomax-pesomin); %Cte de ponderación del peso
kpe=1+0.1*kp+0.1*ke;
k1=medicacion1; %Cte para determinar si toma o no la medicacion 1
k2=medicacion2; %Cte para determinar si toma o no la medicacion 2
k3=medicacion3; %Cte para determinar si toma o no la medicacion 3

%Añadimos ruido para generar datos sintéticos aleatorios
%que servirán para entrenar a la red neuronal.
for i=1:3
a(:,i)=a(:,i)+0.05*acmax(i)*(-1+2*rand(length(a(:,i)),1))+ 0.1*k1*acmax(i)*sin(2*pi*5*t)+ 0.1*k2*
    acmax(i)*sin(2*pi*7*t)+ 0.1*k3*acmax(i)*sin(2*pi*9*t);
a(:,i)=kpe*a(:,i);
end
```

2.2.2 Implementación del modelo de generación de datos sintéticos en Python

Todo lo anterior se consigue en Python con las siguientes líneas de código:

```
ke=(edad-20)/(100-20) #Cte de ponderación de la edad va de 0 a 1
kp=(peso-40)/(160-40) #Cte de ponderación del peso va de 0 a 1
kpe=1+0.1*ke+0.1*kp
k1=medicacion1 #Cte de pico de la medicación 1
k2=medicacion2 #Cte de pico de la medicación 2
k3=medicacion3 #Cte de pico de la medicación 3


#Añadimos ruido aleatorio para generar datos sintéticos.
args1=np.dot(2*pi*5,tiempos)
args2=np.dot(2*pi*7,tiempos)
args3=np.dot(2*pi*9,tiempos)
Acx=Acx+0.05*Acxmax*(-np.ones(L)+2*np.random.random(L))+0.1*k1*Acxmax*np.sin(args1)+0.1*k2*Acxmax
    *np.sin(args2)+0.1*k3*Acxmax*np.sin(args3)
Acy=Acy+0.05*Acymax*(-np.ones(L)+2*np.random.random(L))+0.1*k1*Acymax*np.sin(args1)+0.1*k2*Acymax
    *np.sin(args2)+0.1*k3*Acymax*np.sin(args3)
Acz=Acz+0.05*Aczmax*(-np.ones(L)+2*np.random.random(L))+0.1*k1*Aczmax*np.sin(args1)+0.1*k2*Aczmax
    *np.sin(args2)+0.1*k3*Aczmax*np.sin(args3)
Acx=kpe*Acx
Acy=kpe*Acy
Acz=kpe*Acz
```

2.3 Metodología y soluciones adoptadas durante el desarrollo del proyecto en MATLAB

En esta sección se comentarán los métodos y las soluciones propuestas durante el desarrollo de la aplicación en MATLAB. Cabe mencionar que se han creado una serie de ficheros cuya denominación se irá mencionando a medida que vayan apareciendo las tareas que resuelve cada uno de ellos.

2.3.1 Almacenamiento de los datos para llevar a cabo el entrenamiento de la red neuronal

Con el fin de almacenar los datos que posteriormente se emplearán para entrenar la red neuronal se han creado dos ficheros denominados **if_not_exists** e **if_not_exists_sin** que se ejecutarán como pasos previos a la ejecución de los pasos funcionales de la aplicación. Estos ficheros crearán dos archivos .mat que servirán para almacenar los temblores medidos en pacientes reales y los generados sintéticamente, respectivamente. El primero de ellos se denomina **Dat_train.mat** y el segundo **Dat_train_sin.mat**. Estos archivos se han creado con variables vacías en su interior que se irán completando a medida que se obtienen los datos. La primera de ellas se ha denominado **x** y queda reservada para almacenar los valores asociados a las entradas de la red, es decir, los campos de edad, peso, sexo, toma de la medicación de tipo 1, toma de la medicación de tipo 2 y toma de la medicación de tipo 3 de los pacientes. Por otro lado, la segunda variable creada se ha denominado **y_d** y se emplea para llevar a cabo el almacenamiento de los valores correspondientes a las salidas de la red neuronal, es decir, los campos relacionados con la frecuencia y la amplitud a la que se da el pico máximo del temblor en los pacientes.

2.3.2 Medición de los temblores con el acelerómetro del teléfono móvil

El registro de los temblores con el acelerómetro del teléfono móvil se ha llevado a cabo haciendo uso del fichero desarrollado denominado **paso1** y de la herramienta de Sensores que aparece en el menú de MATLAB Mobile.

Por un lado, en el **paso1** únicamente se emplea el comando **mobiledev** para crear un objeto que sirve para llevar a cabo la lectura de datos de los sensores que se habiliten posteriormente. Por otro lado, la herramienta de Sensores del menú de MATLAB Mobile sirve para habilitar los sensores que se desean utilizar y para especificar la tasa de muestreo con la que se desean llevar a cabo las mediciones. En este caso particular se han habilitado los sensores asociados a la medición de la aceleración y se ha establecido una frecuencia de muestreo de 100 Hz, que es la mayor frecuencia de muestreo soportada actualmente en un smartphone convencional. Esto indica que si la medición se ha llevado a cabo en el dominio del tiempo a 100 Hz, al pasar al dominio de la frecuencia, se tendrá una tasa de muestreo de 50 Hz, es decir, de la mitad. Por tanto, no se podrán detectar temblores con frecuencias superiores a dicho valor.

2.3.3 Postprocesado de los temblores registrados con el acelerómetro del teléfono móvil

Las tareas relacionadas con el postprocesado de los temblores registrados se han resuelto con la elaboración de dos ficheros denominados **paso2** y **paso3**. Según se vayan describiendo dichas tareas se especificará el fichero donde han sido resueltas.

Introducción de la información del paciente registrado

Al ejecutar el archivo denominado **paso2** se piden los datos asociados al paciente al que se le ha realizado la medición del temblor, es decir, será necesario introducir su número asignado, edad, sexo, la medicación que toma, la parte del cuerpo en la que se ha llevado a cabo la medición del temblor y la postura con la que se ha realizado. Todos estos datos se almacenan en la variable de entrada creada en los archivos .mat previamente mencionados para, en un futuro, emplearse en el proceso de entrenamiento de la red neuronal.

Filtrado de los temblores registrados

La tarea del filtrado de los temblores registrados con el acelerómetro del teléfono móvil se ha abordado en el archivo **paso2**.

Los datos asociados a los temblores en el dominio temporal en los tres ejes recogidos a través del acelerómetro de un teléfono móvil no pueden ser tratados directamente dado que presentan ruido que puede afectar a la interpretación de los mismos. Es por este motivo que se necesita que dichos datos pasen por un proceso de filtrado con el fin de que la información que se extraiga de los mismos a posteriori tenga

valor. En el presente proyecto se ha aplicado a los temblores registrados con el acelerómetro del teléfono móvil un filtro de Chebyshev de tipo 1. En MATLAB esta tarea se ha llevado a cabo por medio del uso de los comandos **cheby1** y **filter**. El primero de ellos sirve para establecer los parámetros que va a tener el filtro y pide como parámetros de entrada el tipo y el orden del filtro, el factor de rizado y la frecuencia de corte ofreciendo a la salida el numerador y el denominador de la función de transferencia asociada a dicho filtro. El filtro implementado en este caso se corresponde con un filtro de paso alto, orden 3, factor de rizado 0.1 y una frecuencia de corte de 0.5 Hz. Una vez establecidos dichos parámetros que definen la arquitectura del filtro, se procede a la aplicación del mismo con el comando **filter**. Dicho comando pide como parámetros de entrada el numerador y el denominador de la función de transferencia del filtro previamente calculados con **cheby1** y la aceleración correspondiente al temblor registrado sin filtrar y devuelve a la salida la aceleración correspondiente al temblor registrado ya filtrada.

Paso de los temblores al dominio de la frecuencia

El paso de los temblores registrados del dominio del tiempo al dominio de la frecuencia se ha llevado a cabo en el archivo **paso2**.

Tras pasar por el proceso de filtrado correspondiente, los temblores han de pasarse del dominio temporal al dominio de la frecuencia. Esta tarea se ha llevado a cabo de dos formas, con la transformada rápida de Fourier del temblor haciendo uso del comando **fft** y a través de la estimación de la densidad espectral de potencia del temblor con el comando **pwelch**. El comando **fft** pide como entrada la aceleración asociada al temblor registrado ya filtrada y devuelve como salida la transformada rápida de Fourier de la misma. Por otro lado, el comando **pwelch** pide como entrada tanto la aceleración asociada al temblor registrado ya filtrada como la frecuencia de muestreo y devuelve como salida la estimación de la densidad espectral de potencia de la aceleración asociada al temblor. La estimación de la densidad espectral de potencia de una señal se lleva a cabo realizando una discretización de dicha señal según indique la frecuencia de muestreo facilitada y realizando la transformada rápida de Fourier de cada uno de los segmentos de dicha discretización por separado. De ahí que en el comando **pwelch** se pidan tanto la señal como la frecuencia de muestreo.

Representaciones gráficas y valores críticos de los temblores registrados

Por último, la ejecución del **paso2** devuelve las representaciones gráficas de los temblores registrados tanto en el dominio del tiempo como en el dominio de la frecuencia, así como los valores numéricos asociados a los picos máximos del temblor en ambos dominios. En concreto, las representaciones gráficas devueltas son la aceleración asociada al temblor en cada uno de los ejes frente al tiempo, la transformada rápida de Fourier de dicha aceleración frente a la frecuencia y la estimación de la densidad espectral de potencia de dicha aceleración frente a la frecuencia. Para llevarlas a cabo se ha hecho uso del comando **plot**. Estas representaciones permiten ver los valores críticos asociados a los picos en el dominio de la frecuencia que tanta relevancia tienen en el análisis médico.

Acceso a datos de pacientes anteriores

Al ejecutar el fichero **paso2** se genera un archivo .mat denominado con la fecha de la medición y los datos principales del paciente al que se le ha realizado la medición con el fin de almacenar dichos datos para posteriores consultas. El acceso a dichos archivos se lleva a cabo con la ejecución del fichero **paso3**, el cual se encarga de mostrar por pantalla todos los archivos almacenados en el espacio de MathWorks enumerados y permite seleccionar el archivo deseado mediante la introducción del número asociado al mismo. Tras la selección del archivo asociado al paciente que se desea consultar se muestran las representaciones gráficas y los valores críticos del temblor asociado a dicho paciente al igual que sucedía con la ejecución del **paso2**.

2.3.4 Desarrollo de la red neuronal

Para llevar a cabo el entrenamiento de la red neuronal en el lenguaje de programación MATLAB se ha hecho uso de una serie de ficheros desarrollados durante el presente proyecto cuya denominación se irá mencionando conforme vayan apareciendo los problemas que solventan.

Generación de datos sintéticos

Dada la falta de datos asociados a mediciones de temblores llevadas a cabo en pacientes reales en el momento del desarrollo del presente proyecto, es necesario generar datos sintéticamente siguiendo el proceso que ya se explicó anteriormente en el presente capítulo. Dicha generación de datos junto al almacenamiento de los mismos se lleva a cabo a través de la ejecución del fichero desarrollado **datsin**.

Entrenamiento de la red neuronal

Todo lo referente a la resolución del proceso de entrenamiento de la red neuronal se ha llevado a cabo en el fichero denominado **train_datsin** y se explica a continuación.

En primer lugar, se carga el archivo donde se almacenan los datos de los temblores generados sintéticamente para su utilización en el proceso. A continuación, se normalizan los valores de las entradas para que se encuentren acotados entre 0 y 1 con el fin de facilitar el proceso de entrenamiento. Posteriormente, se diseña la arquitectura de la red neuronal con el comando **fitnet** [17] especificando el número de capas ocultas y el número de neuronas que componen cada capa oculta. Esta arquitectura ha sido diseñada en base a ensayos de tipo prueba y error hasta conseguir el resultado óptimo. Dicho resultado se ha conseguido estableciendo una red neuronal compuesta por 3 capas ocultas, cada una de ellas compuesta por 30, 20 y 10 neuronas respectivamente. Después, se divide el conjunto de datos en valores empleados para el entrenamiento de la red y valores usados para la validación de los resultados arrojados por la red. Esta división se lleva a cabo de forma empírica hasta ver cuál es la óptima. En concreto, en este caso se ha establecido que el 70 % del conjunto de datos se destine a la fase de entrenamiento mientras que el 30 % restante se utilice para la validación de resultados. Por otro lado, se establece el número de épocas máximo en base a ensayos de prueba y error al igual que sucedió con el número de capas ocultas y el número de neuronas de cada capa. Dicho número se ha fijado en 300. Además, se especifican las funciones de activación empleadas en cada capa intermedia de la red neuronal con el fin de optimizar el resultado de la misma. Para la primera capa oculta se ha escogido la función sigmoide mientras que para la segunda y la tercera se ha elegido la función tangente hiperbólica. Una vez completado todo lo anterior, se entrena la red con el comando **train** [18] pasando la arquitectura de la red diseñada y los conjuntos de datos preparados para el entrenamiento. Finalmente, se muestran los resultados ofrecidos por la red neuronal tras completar el proceso de entrenamiento con el fin de apreciar las diferencias entre los valores predichos y los valores que habrían de darse en realidad. Además, se guarda la red entrenada en un archivo denominado **net.mat** para que pueda ser empleada con el fin de predecir valores en un futuro y así poder compararlos con los procedentes de mediciones reales.

Predicciones de la red neuronal

El fichero desarrollado para llevar a cabo las predicciones a través del empleo de la red neuronal entrenada se denomina **pred**. Dicho fichero pide por pantalla los datos del paciente sobre el que se quiere realizar la predicción, es decir, su edad, su peso, su sexo y las medicaciones que toma y los normaliza para que se encuentren acotados entre 0 y 1. Tras esta normalización, los valores entran a la red neuronal entrenada que se guardó en el archivo **net.mat** y esta devuelve tanto la amplitud como la frecuencia a la que debería de darse el pico máximo del temblor de dicho paciente según lo aprendido por la red durante su proceso de entrenamiento.

2.4 Metodología y soluciones adoptadas durante el desarrollo del proyecto en Python

En esta sección se comentarán los métodos y las soluciones propuestas durante el desarrollo de la aplicación en Python. Cabe mencionar que se han creado una serie de ficheros cuya denominación se irá mencionando a medida que vayan apareciendo las tareas que resuelve cada uno de ellos.

2.4.1 Medición de los temblores con el acelerómetro

Con el fin de conseguir registrar los temblores de los pacientes procedentes de las mediciones realizadas con el acelerómetro incorporado en el teléfono móvil se ha empleado el intérprete móvil de Python conocido como QPython 3L. El motivo de escoger este intérprete en concreto es que trae incorporado el módulo **androidhelper** que permite, entre otras cosas, acceder al acelerómetro del teléfono móvil.

En el fichero desarrollado denominado **paso1** se emplea el módulo **androidhelper** para la lectura de los temblores con el acelerómetro del teléfono móvil. La frecuencia de muestreo se ha establecido en 100 Hz al ser esta la máxima soportada en la actualidad por un smartphone convencional, por tanto, habrá que tener en cuenta que al pasar al dominio de la frecuencia la tasa de muestreo se verá reducida a la mitad, es decir, a 50 Hz, impidiendo detectar temblores con una frecuencia superior. La ejecución de dicho fichero demanda introducir por pantalla la duración de la medición a realizar y, una vez cumplido el tiempo establecido, se almacenan todos los datos registrados del temblor durante ese período en unos archivos .txt que, posteriormente, podrán ser consultados en el postprocesado.

2.4.2 Postprocesado de los temblores registrados con el acelerómetro

Los ficheros desarrollados para llevar a cabo la resolución de esta parte del proyecto se denominan **paso2**, **paso3** y **paso4** y se irán comentando según vayan apareciendo los problemas que solventan.

Introducción de la información del paciente registrado

Al ejecutar el archivo denominado **paso2** se piden los datos asociados al paciente al que se le ha realizado la medición del temblor, es decir, será necesario introducir su número asignado, edad, sexo, la medicación que toma, la parte del cuerpo en la que se ha llevado a cabo la medición del temblor y la postura con la que se ha realizado.

Filtrado de los datos tomados del acelerómetro de un teléfono móvil

La tarea del filtrado de los temblores registrados con el acelerómetro del teléfono móvil se ha abordado en el fichero **paso2**.

Los datos asociados a los temblores en el dominio temporal en los tres ejes recogidos a través del acelerómetro de un teléfono móvil no pueden ser tratados directamente dado que presentan ruido que puede afectar a la interpretación de los mismos. Es por este motivo que se necesita que dichos datos pasen por un proceso de filtrado con el fin de que la información que se extraiga de los mismos a posteriori tenga valor. En el presente proyecto se ha aplicado a los temblores registrados con el acelerómetro del teléfono móvil un filtro de Chebyshev de tipo 1. En Python esta tarea se ha llevado a cabo por medio del uso de las funciones **signal.cheby1** y **signal.filtfilt** procedentes de la librería **scipy**. La primera de ellas sirve para establecer los parámetros que va a tener el filtro y pide como parámetros de entrada el tipo y el orden del filtro, el factor de rizado y la frecuencia de corte ofreciendo a la salida el numerador y el denominador de la función de transferencia asociada a dicho filtro. El filtro implementado en este caso se corresponde con un filtro de paso alto, orden 3, factor de rizado 0.1 y una frecuencia de corte de 0.5 Hz. Una vez establecidos dichos parámetros que definen la arquitectura del filtro, se procede a la aplicación del mismo con la función **signal.filtfilt**. Dicha función pide como parámetros de entrada el numerador y el denominador de la función de transferencia del filtro previamente calculados con **signal.cheby1** y la aceleración correspondiente al temblor registrado sin filtrar y devuelve a la salida la aceleración correspondiente al temblor registrado ya filtrada.

Paso de los temblores al dominio de la frecuencia

El paso de los temblores registrados del dominio del tiempo al dominio de la frecuencia se ha llevado a cabo en el archivo **paso2**.

Tras pasar por el proceso de filtrado correspondiente, los temblores han de pasarse del dominio temporal al dominio de la frecuencia. Esta tarea se ha llevado a cabo de dos formas, con la transformada rápida de Fourier del temblor haciendo uso de la función **fftpack.fourier.fft** de la librería **scipy** y a través de la estimación de la densidad espectral de potencia del temblor con la función **signal.welch** de la librería **scipy**. Por un lado, la función **fftpack.fourier.fft** pide como entrada la aceleración asociada al temblor registrado ya filtrada y devuelve como salida la transformada rápida de Fourier de la misma. Por otro lado, la función **signal.welch** pide como entrada tanto la aceleración asociada al temblor registrado ya filtrada como la frecuencia de muestreo y devuelve como salida la estimación de la densidad espectral de potencia de la aceleración asociada al temblor. La estimación de la densidad espectral de potencia de una señal se lleva a cabo realizando una discretización de dicha señal según indique la frecuencia de muestreo facilitada y realizando la transformada rápida de Fourier de cada uno de los segmentos de dicha discretización por separado. De ahí que en la función **signal.welch** se pidan tanto la señal como la frecuencia de muestreo.

Valores críticos de los temblores registrados

Por último, la ejecución del **paso2** devuelve los valores numéricos asociados a los picos máximos del temblor tanto en el dominio de la frecuencia como en el dominio del tiempo.

Acceso a datos de pacientes anteriores y representaciones gráficas asociadas

Al ejecutar el fichero **paso2** se genera un archivo .pkl denominado con la fecha de la medición y los datos principales del paciente al que se le ha realizado la medición con el fin de almacenar dichos datos para posteriores consultas. El acceso a dichos archivos se lleva a cabo con la ejecución de los ficheros desarrollados conocidos como **paso3** y **paso4**. El primero de ellos se encarga de mostrar por pantalla todos los archivos almacenados en la carpeta **qpython** enumerados y permite seleccionar el archivo deseado mediante la introducción del número asociado al mismo. El segundo, se encarga de mostrar las representaciones gráficas y los valores críticos del temblor asociado a dicho paciente

En concreto, las representaciones gráficas devueltas son la aceleración asociada al temblor del paciente en cada uno de los ejes frente al tiempo, la transformada rápida de Fourier de dicha aceleración frente a la frecuencia y la estimación de la densidad espectral de potencia de dicha aceleración frente a la frecuencia. Para llevarlas a cabo se ha hecho uso de la función **pyplot** de la librería **matplotlib**. Estas representaciones permiten ver los valores críticos asociados a los picos del temblor en el dominio de la frecuencia que tanta relevancia tienen en el análisis médico.

2.4.3 Solución de la problemática asociada al uso simultáneo de matplotlib y la función input en Pydroid 3

Durante el desarrollo de la aplicación en `Python` se observó que Pydroid 3 presentaba una problemática a la hora de ejecutar un fichero donde se emplease simultáneamente la función input de `Python` necesaria para que el usuario introduzca valores por pantalla y la librería matplotlib necesaria para llevar a cabo cualquier representación gráfica en `Python`. Por este motivo, se decidió solventar el problema acontecido por medio de la separación de la función y la librería previamente mencionadas en distintos ficheros. Por tanto, todo lo asociado a la introducción de valores por pantalla se puso en un fichero y todo lo asociado a las representaciones gráficas en otro. Sin embargo, la necesidad de utilizar ambos en el mismo fichero en ocasiones era inevitable. Por tanto, se decidió tomar la siguiente solución:

Por un lado, en un fichero se introducen los valores por pantalla necesarios y se almacenan en una base de datos provisional. Por otro lado, en otro fichero distinto se lleva a cabo la lectura de los valores que se introdujeron en la base de datos provisional comentada anteriormente y se representan gráficamente con el empleo de la librería **matplotlib**. De esta manera se ha conseguido comunicar indirectamente los valores que se introdujeron por pantalla en el primer fichero y las representaciones gráficas que el usuario demanda en el segundo.

Un ejemplo donde se ha seguido este proceso ha sido para llevar a cabo la consulta de los datos asociados al temblor de un paciente almacenados previamente. Primero, es necesario ejecutar el fichero desarrollado denominado **paso3** en Pydroid 3 y, una vez hecho esto, aparece un listado con los archivos disponibles para la consulta enumerados. Además, se pide por pantalla la introducción del número asociado al archivo que se desea consultar. Tras la introducción de dicho número, este se almacena en la base de datos provisional anteriormente comentada. A continuación, con la ejecución del fichero desarrollado denominado **paso4** se lleva a cabo una consulta a la base de datos provisional donde quedó almacenado el número del archivo a consultar y se representan gráficamente los valores de interés presentes en el archivo seleccionado.

2.4.4 Almacenamiento de los datos para llevar a cabo el entrenamiento de la red neuronal

Con la finalidad de llevar a cabo el almacenamiento de los datos que se emplearán en el proceso de entrenamiento de la red neuronal se ha llevado a cabo la creación de dos bases de datos distintas dependiendo del tipo de datos que se almacenan en las mismas. Una orientada al almacenamiento de datos procedentes de las mediciones realizadas a pacientes reales y otra para almacenar datos generados sintéticamente.

Las creaciones de dichas bases de datos van implícitas en la ejecución de los ficheros desarrollados conocidos como **paso2** y **datsin**, respectivamente. En ellas se almacenan los datos correspondientes a las entradas de la red (edad, peso, sexo, medicación 1, medicación 2, medicación 3) y a las salidas de la misma (frecuencia del pico máximo del temblor, amplitud del pico máximo del temblor).

Para llevar a cabo la creación y la cumplimentación de las bases de datos previamente mencionadas se ha empleado el módulo **sqlite3** [19] .

2.4.5 Desarrollo de la red neuronal en Python

Para llevar a cabo el entrenamiento de la red neuronal en el lenguaje de programación Python se ha hecho uso de una serie de ficheros desarrollados durante el presente proyecto cuya denominación se irá mencionando conforme vayan apareciendo los problemas que solventan.

Generación de datos sintéticos

Dada la falta de datos asociados a mediciones de temblores llevadas a cabo en pacientes reales en el momento del desarrollo del presente proyecto, es necesario generar datos sintéticamente siguiendo el proceso que ya se explicó anteriormente en el presente capítulo. Dicha generación de datos junto al almacenamiento de los mismos se lleva a cabo a través de la ejecución del fichero desarrollado denominado **datsin**.

Entrenamiento de la red neuronal

Todo lo referente a la resolución del proceso de entrenamiento de la red neuronal se ha llevado a cabo en el fichero desarrollado denominado **train_datsin** y se explica a continuación.

En primer lugar, se importa la librería **sklearn** y se realiza una consulta a la base de datos donde se almacenan los datos de los temblores generados sintéticamente y se almacenan en un Dataframe para poder ser empleados en el proceso de entrenamiento de la red. A continuación, se divide el conjunto de datos en valores utilizados para la fase de entrenamiento de la red y valores empleados para la validación de los resultados arrojados por la red. Esta división se realiza de forma empírica hasta alcanzar la configuración óptima. En este caso en particular se ha dividido el conjunto de datos de tal forma que el 80 % de ellos se han empleado en la fase de entrenamiento mientras que el 20 % restante se ha usado en la fase de validación. Una vez hecho todo lo anterior, se diseña la arquitectura de la red con el empleo de la función **neural_network.MLPRegressor** de la librería **sklearn** . Cabe destacar que en esta fase se observó que lo más óptimo era separar el proceso de entrenamiento en dos. Por un lado, se llevaría a cabo el entrenamiento de una red encargada de predecir el valor de la frecuencia del pico máximo del temblor de un determinado paciente. Por otro lado, se entrenaría otra red distinta con el fin de predecir el valor de la amplitud de dicho pico. Por un lado, se decidió que la arquitectura de la red encargada de predecir la frecuencia del pico máximo del temblor estuviese compuesta por 3 capas ocultas cada una de ellas compuesta por 30, 20 y 10 neuronas respectivamente. Además, se estableció un número máximo de épocas de 600 y se empleó la función de activación relu en cada una de las capas ocultas. Por otro lado, la arquitectura de la red encargada de predecir la amplitud del pico máximo del temblor se diseñó de tal manera que tuviese 4 capas ocultas cada una de ellas compuesta por 30, 30, 20 y 20 neuronas respectivamente. Además, el número máximo de épocas se fijó en 600 y la función de activación escogida en este caso fue la sigmoide en todas las capas ocultas. Finalmente, se guardan ambas redes entrenadas en dos archivos denominados **red_entranada_1.sav** y **red_entranada_2.sav**, respectivamente, para que se puedan emplear con un fin predictivo a posteriori.

Predicciones de la red neuronal

El fichero desarrollado para llevar a cabo las predicciones a través del empleo de las redes neuronales entrenadas se denomina **salida_pred**. Dicho fichero pide por pantalla los datos del paciente sobre el que se quiere realizar la predicción, es decir, su edad, su peso, su sexo y las medicaciones que toma. Tras su introducción, los valores entran a las redes neuronales entrenadas que se guardaron en los archivos **red_entranada_1.sav** y **red_entranada_2.sav** y estas devuelven tanto la amplitud como la frecuencia a la que

debería de darse el pico máximo del temblor de dicho paciente según lo aprendido durante sus procesos de entrenamiento.

3 Puesta en funcionamiento de la aplicación

Una vez completada la fase de desarrollo del proyecto es necesario conocer los pasos que hay que seguir para que la aplicación lleve a cabo las tareas para las que se ha desarrollado. Por tanto, el objetivo del presente capítulo es explicar el proceso de la puesta en funcionamiento de la aplicación tanto en MATLAB como en Python.

3.1 Puesta en funcionamiento de la aplicación en MATLAB

3.1.1 Pasos previos a la ejecución de los códigos

El primer paso a llevar a cabo será la instalación de MATLAB Mobile a través de Play Store. El enlace a la instalación queda facilitado en [6] en la bibliografía del presente proyecto. A continuación, al abrir MATLAB Mobile será necesario crear una cuenta de MathWorks en caso de que no se posea una ya. Posteriormente, habrá que asociar una licencia vigente del servicio de mantenimiento de software de MathWorks con su cuenta de MathWorks para tener acceso a otros productos complementarios y conseguir 5 GB de almacenamiento en la nube para MATLAB Drive [6]. La siguiente tarea a realizar es la de subir todos los scripts desarrollados durante el proyecto al espacio de MATLAB Drive asociado a la cuenta de MathWorks en cuestión. Esta tarea queda detallada en el Apéndice A. Por último, crearemos los archivos que servirán de almacén para los datos asociados a los temblores tanto reales como sintéticos por separado para luego emplearlos en el proceso de entrenamiento de la red neuronal. Para ello, será necesario dirigirse a la ventana de comandos de MATLAB Mobile. Esto se realiza abriendo MATLAB Mobile y dirigiéndose al menú desplegable accesible a través de las tres líneas horizontales paralelas que se encuentran en la esquina superior izquierda al abrir Matlab Mobile y seleccionando la opción Comandos.

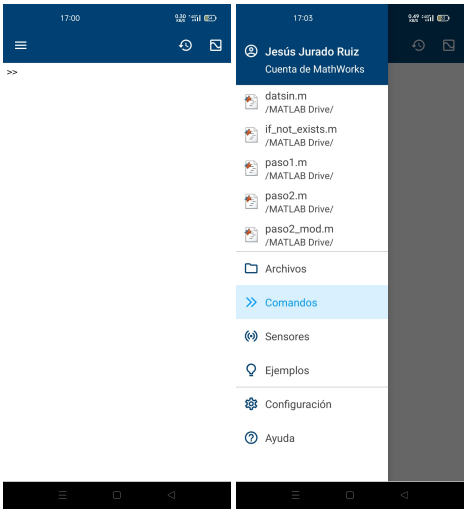

Figura 3.1 Ventana de comandos y menú desplegable de MATLAB Mobile.

Una vez allí, para almacenar toda la información relevante asociada a los temblores de los pacientes a los que se les va a realizar las mediciones, será necesario crear un archivo .mat. Este archivo se genera escribiendo en la ventana de comandos **if_not_exists**. Los datos almacenados en este archivo se podrán utilizar para el entrenamiento de la red neuronal cuando exista un número suficiente de los mismos. Este paso sólo ha de ejecutarse la primera vez que se usa la aplicación dado que su única función es la creación de un archivo que servirá de almacén para los datos asociados a los temblores de pacientes reales y que se irá completando a medida que se realicen las mediciones a los pacientes. A continuación, de forma análoga a como se ha hecho con los datos asociados a los temblores de pacientes reales, se ejecutará en la ventana de comandos el paso **if_not_exists_sin** para generar un archivo .mat que servirá de almacén para los datos de los temblores generados sintéticamente y que se emplearán en el entrenamiento de la red neuronal de forma provisional hasta que se tenga el número suficiente de datos procedentes de pacientes reales.

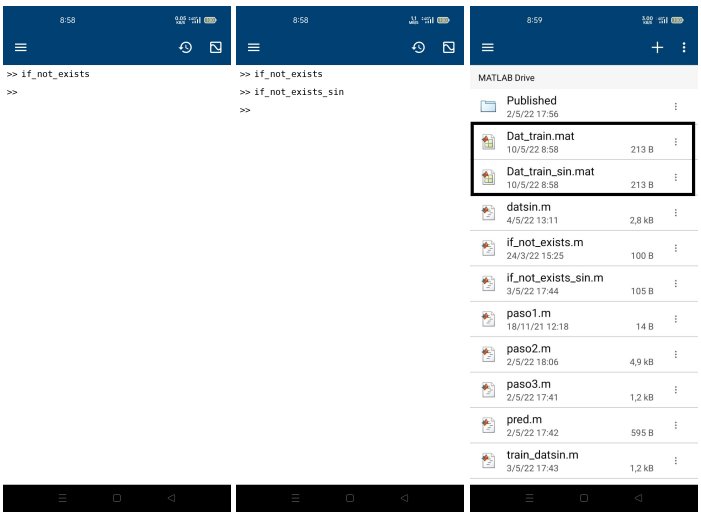

Figura 3.2 Ejecución de los ficheros **if_not_exists** y **if_not_exists_sin** y archivos .mat generados.

3.1.2 Puesta a punto de la obtención y extracción de información de los temblores registrados

Para llevar a cabo el registro del temblor mediante el uso del acelerómetro de un teléfono móvil a través de `MATLAB` Mobile será necesario dirigirse a la ventana de comandos tal como se explicó anteriormente. Una vez situados en ella el primer paso será escribir en dicha ventana **paso1**, que es el fichero encargado de llevar a cabo la comunicación con los sensores del teléfono móvil.

Figura 3.3 Ejecución del archivo **paso1**.

Una vez ejecutado el **paso1** se procede a la selección de los sensores que se desean emplear para la toma de la muestra. Para ello, habrá que volver a abrir el menú desplegable comentado anteriormente y dirigirse al apartado Sensores. En este módulo será necesario poner la tasa de muestreo en 100 Hz (la máxima soportada por un smartphone convencional en la actualidad) y activar los sensores asociados a la aceleración, que son los que ocupan el interés de este proyecto.

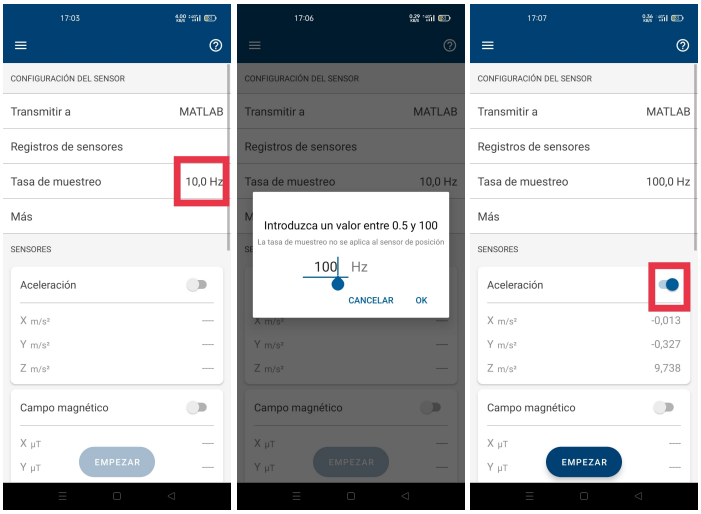

Figura 3.4 Modificación de los parámetros de los sensores.

Cuando el paciente se encuentre preparado para realizar la medición se pulsará el botón azul de Empezar localizado en la parte inferior central de la pantalla y se llevará a cabo la toma de la muestra durante el tiempo que sea necesario. Una vez pasado dicho período de tiempo se pulsará el botón rojo de parar. Por lo general, se estima que con una muestra de aproximadamente 10 segundos es más que suficiente para sacar conclusiones.

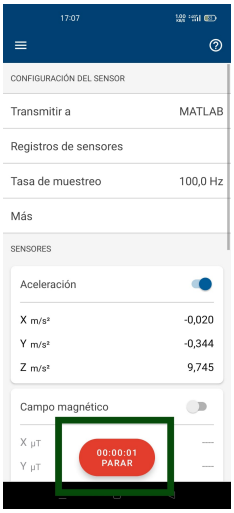

Figura 3.5 Parada de la medición.

El siguiente paso está relacionado con el tratamiento posterior del temblor registrado para que se pueda obtener la mayor cantidad de información posible del mismo. Con el fin de eliminar el ruido del temblor registrado es necesario llevar a cabo el filtrado de dicha señal. Una vez llevada a cabo esta tarea se procede a la extracción de los valores de interés del temblor y se lleva a cabo una representación gráfica del mismo tanto en el dominio de la frecuencia como en el dominio temporal.

Todo lo comentado en el párrafo anterior se consigue con la ejecución en la ventana de comandos del **paso2**.

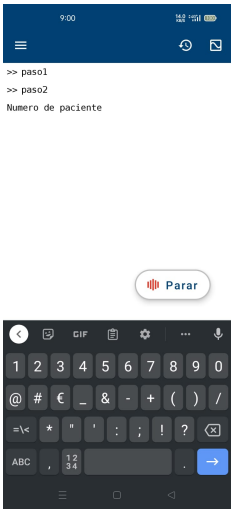

Figura 3.6 Ejecución del archivo **paso2**.

En este paso se pedirá al usuario información acerca del paciente al que se le están realizando las mediciones para ser almacenada y posteriormente utilizada. Con la ejecución del mismo se lleva a cabo la representación del temblor registrado tanto en el dominio de la frecuencia como en el dominio temporal y se obtienen también detalles sobre los picos de dichos temblores en ambos dominios. Del dominio de la frecuencia es del que se obtiene la información más relevante, la amplitud y la frecuencia a la que se da el pico máximo del temblor del paciente al que se le ha realizado la medición (datos muy importantes para la determinación temprana de una enfermedad de naturaleza neurológica).

Los datos almacenados en el **paso2** podrán ser consultados posteriormente con la ejecución del **paso3**, el cual mostrará la lista de ficheros disponibles en el espacio de MATLAB Drive denominados de forma ordenada según la fecha en que se tomase la muestra. Una vez seleccionado un archivo de la lista se mostrarán por pantalla los valores críticos del paciente en cuestión así como las representaciones gráficas en los dominios temporal y frecuencial previamente comentados.

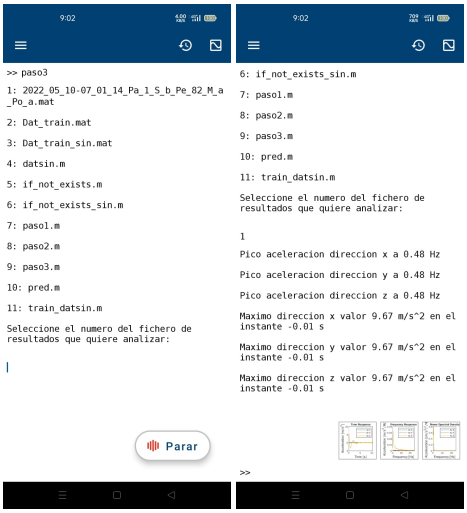

Figura 3.7 Ejecución y resultados obtenidos del archivo **paso3**.

3.1.3 Puesta a punto de la red neuronal

Generación de datos sintéticos para el entrenamiento

Para poder llevar a cabo la fase de entrenamiento de una red neuronal son necesarios datos que sigan un determinado patrón que la red pueda identificar para así poder cumplir con su propósito predictivo. Sin embargo, dadas las dificultades que en muchas ocasiones se presentan para la obtención del número de datos que hacen falta para esta tarea, es necesario llevar a cabo una generación de datos sintéticos que se asemejen lo máximo posible a los datos que se darían en la realidad. El primer paso para la generación sintética de datos es partir de datos reales, por tanto, será necesario ejecutar el **paso1** y ajustar los parámetros de los sensores igual que se hizo anteriormente. Tras realizar dichos ajustes se registró mi temblor para ser utilizado como punto de partida y, en lugar de ejecutar el **paso2**, se procederá con la ejecución del fichero **datsin** en la ventana de comandos.

Figura 3.8 Ejecución del archivo **datsin**.

Una vez ejecutado este código se generan una serie de datos que se almacenan en el archivo **Dat_train_sin.mat** y que, posteriormente, serán tomados del mismo para su utilización en el entrenamiento de la red neuronal. Este paso únicamente será necesario cuando no se disponga de un número de mediciones de temblores realizadas en pacientes reales suficientes para llevar a cabo el proceso. Cabe destacar que la ejecución de este código puede llevar algunos minutos puesto que se lleva a cabo la generación de una gran cantidad de datos.

Entrenamiento de la red neuronal

Debido a la incapacidad de MATLAB Mobile para emplear los módulos relacionados con el entrenamiento de redes neuronales esta fase se llevará a cabo en el ordenador. Para ello, una vez obtenidos los datos sintéticos que se emplearán en el proceso, habrá que descargar el archivo **Dat_train_sin.mat** generado en el espacio de MATLAB Drive así como el archivo **train_datsin** y abrir MATLAB en el ordenador.

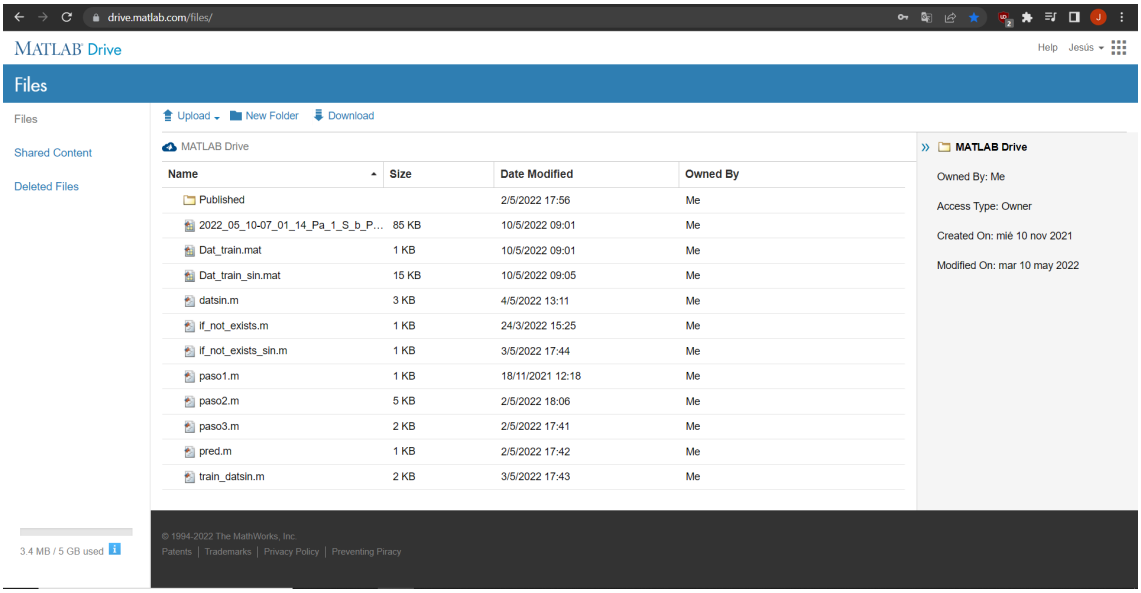

Figura 3.9 Archivos almacenados en MATLAB Drive.

Una vez en el ordenador será necesario abrir la carpeta en la que se encuentren los archivos descargados desde MATLAB Drive mencionados anteriormente y ejecutar el fichero **train_datsin** en la ventana de comandos de MATLAB.

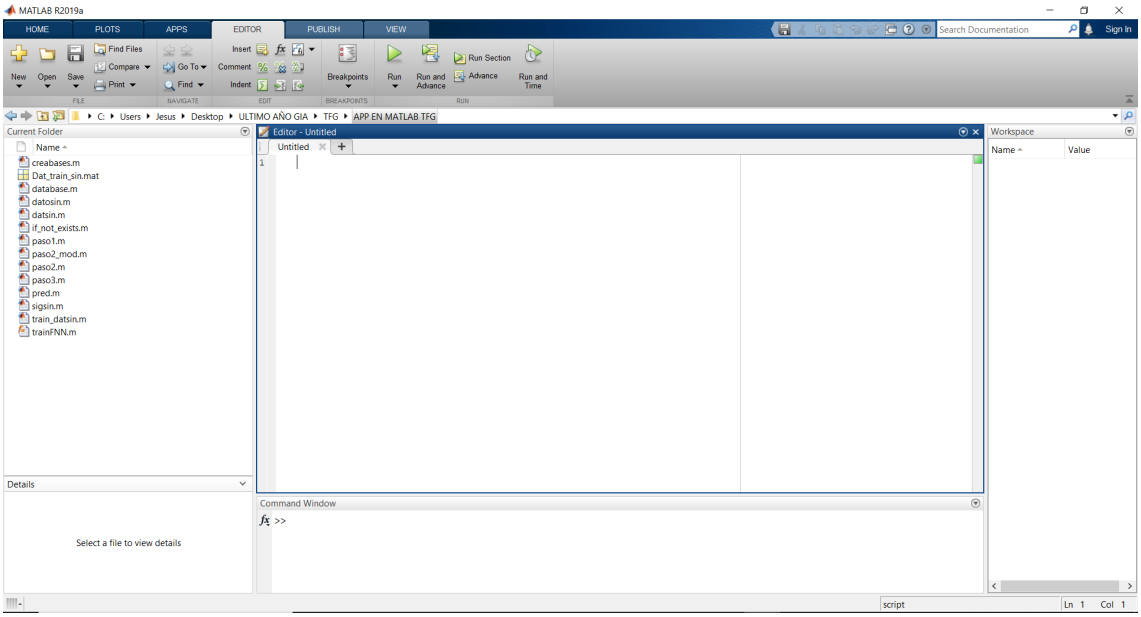

Figura 3.10 Entorno de MATLAB en el ordenador.

Tras llevar a cabo este paso se completará el entrenamiento de la red neuronal y se mostrarán gráficamente los resultados del proceso que permitirán conocer el grado de precisión de la misma y, por tanto, el alcance

de su capacidad predictiva. Además, se generará un archivo denominado **net.mat** en el que se almacena la red ya entrenada y lista para emplearse en predicciones.

El siguiente paso, sería subir a MATLAB Drive el archivo **net.mat** de la red entrenada para que posteriormente pueda ser empleada en el teléfono móvil desde MATLAB Mobile. Por último, mencionar que todas las tareas relacionadas con la subida y descarga de archivos desde MATLAB Drive se explican en el Apéndice A.

Predicción de la red neuronal entrenada

Finalmente, una vez abierto Matlab Mobile habrá que ejecutar en la ventana de comandos el archivo **pred** e introducir los valores de la persona que se está analizando. La ejecución de este archivo permite obtener los resultados que debería tener el paciente de acuerdo a lo aprendido por la red y permite compararlos con los tomados en la realidad por medio del acelerómetro del teléfono móvil.

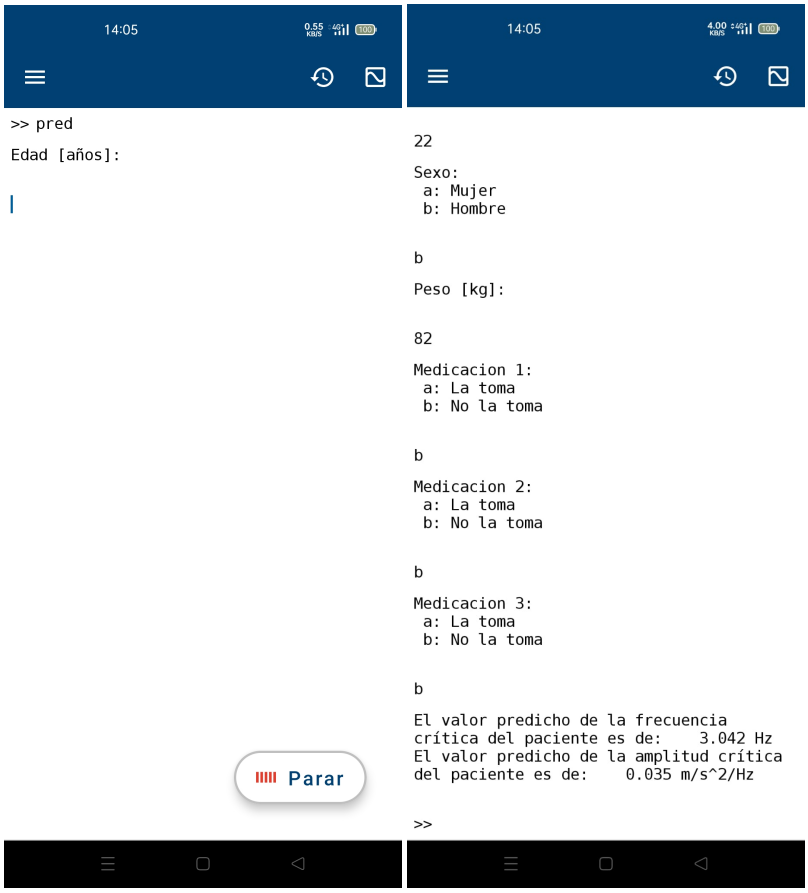

Figura 3.11 Ejecución y resultados del paso **pred** en MATLAB Mobile.

Conclusiones acerca del proceso llevado a cabo con la red neuronal

Comentar que el entrenamiento de la red es algo que únicamente se realizará cuando existan cambios sustanciales en el conjunto de entrenamiento. Mientras tanto, este proceso se puede omitir pasando directamente al paso **pred** para llevar a cabo las predicciones directamente si ya se tiene un archivo **net.mat** con una red entrenada.

3.2 Puesta en funcionamiento de la aplicación en `Python`

3.2.1 Pasos previos a la ejecución de códigos

En primer lugar, será necesario llevar a cabo la instalación desde Play Store de las aplicaciones QPython 3L y Pydroid 3. El enlace a la instalación de QPython 3L queda facilitado en [7] y el de la instalación de Pydroid 3 en [8].

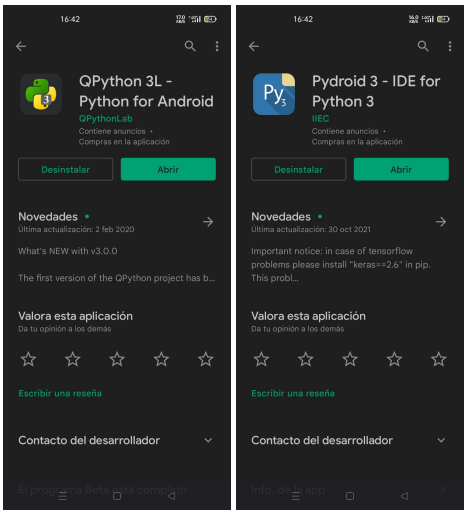

Figura 3.12 QPython 3L y Pydroid 3 en Play Store.

Una vez instalados ambos intérpretes de código habrá que proceder a la instalación, también desde Play Store, de Pydroid repository plugin y de Pydroid permissions plugin, que son un complemento de repositorio de Pydroid que proporciona un repositorio de instalación rápida con paquetes precompilados que contienen librerías nativas y un complemento de Pydroid para obtener permisos adicionales, respectivamente.

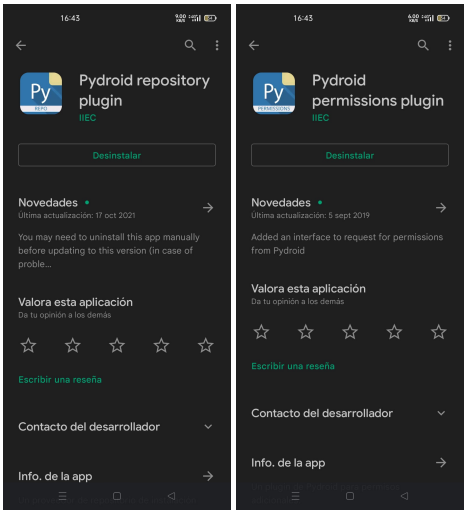

Figura 3.13 Complementos de Pydroid 3 necesarios de instalar.

A continuación, se llevará a cabo el proceso de instalación de las librerías que van a ser necesarias durante el proyecto en la aplicación Pydroid 3. Se detallará en el Anexo B el proceso de instalación de librerías en Pydroid 3 y ese mismo procedimiento tendrá que repetirse con las siguientes librerías: Numpy [20], Matplotlib [21], Scipy [22], Dill [23], Pandas [24], Sklearn [25] y Pickle [26].

Por último, será necesario descargar los archivos que contienen los scripts del proyecto en la carpeta **qpython** que se crea por defecto en **Almacenamiento del teléfono** tras la instalación de QPython 3L.

3.2.2 Puesta a punto de la medición de los temblores

Como se comentó previamente, la parte de la obtención de los datos de los sensores se llevará a cabo mediante el empleo de la aplicación QPython 3L. En primer lugar, será necesario abrir el Editor que aparece en la pantalla inicial y dirigirse al icono de la carpeta que aparece en la parte superior derecha. Esto nos llevará a la carpeta de qpython en la que previamente se han guardado los ficheros desarrollados durante el proyecto.

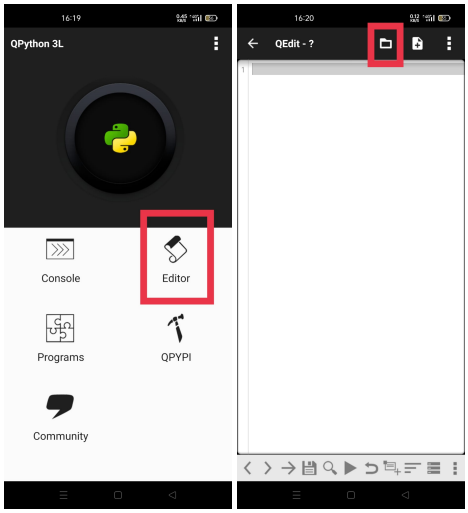

Figura 3.14 Apertura del editor y de la carpeta qpython en QPython 3L.

Una vez allí será necesario seleccionar el archivo **paso1.py** y, tras la selección, se ejecutará con el botón triangular que aparece en la parte inferior central del editor. Tras presionar dicho botón el programa pedirá por pantalla el número de segundos que debe durar la toma de la muestra y, una vez introducido el tiempo de duración deseado, comenzará a registrar el temblor de un determinado paciente empleando el acelerómetro del teléfono móvil. Los datos del temblor se almacenan en archivos .txt en la carpeta qpython para posteriormente ser tratados desde Pydroid 3.

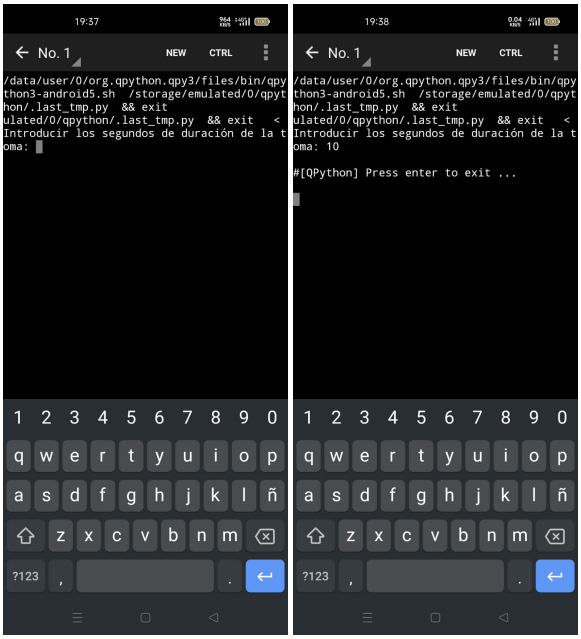

Figura 3.15 Introducción de los segundos de duración de la toma tras la ejecución del **paso1**.

3.2.3 Puesta a punto del tratamiento de los temblores registrados

A partir de este momento la aplicación a emplear será Pydroid 3, tal como se mencionó previamente.

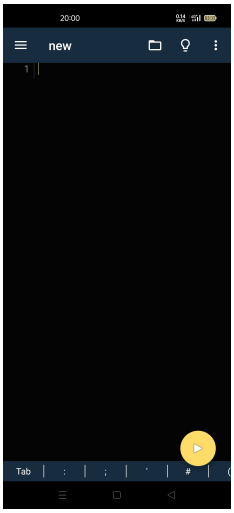

Figura 3.16 Entorno de la aplicación Pydroid 3.

El primer paso será abrir la aplicación Pydroid 3 y dirigirse al icono de la carpeta que aparece en la parte superior derecha. Una vez abierto el desplegable, habrá que seleccionar el subapartado Open y explorar el almacenamiento interno del teléfono para abrir la carpeta qpython en la que se encuentran los ficheros del presente proyecto. Tras la apertura de dicha carpeta será necesario seleccionar el fichero **paso2.py**. Después de abrir dicho archivo habrá que ejecutar el código asociado al mismo pulsando el botón de ejecución amarillo que aparece en la parte inferior derecha e introducir los valores pedidos por pantalla relacionados con el paciente al que se le ha realizado la medición. Una vez recibidos dichos valores, el programa mostrará por pantalla algunos de los valores de interés asociados a los picos del temblor registrado y almacenará los datos en un archivo denominado con la fecha de generación y algunos de los datos del paciente en cuestión en la carpeta qpython.

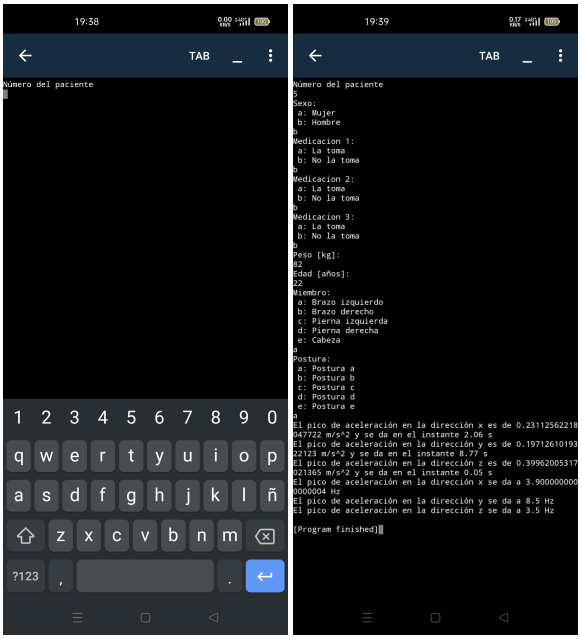

Figura 3.17 Ejecución y resultados del **paso2** en Pydroid 3.

Posteriormente, con el fin de consultar los datos previamente almacenados y ver representaciones gráficas del temblor de un paciente en cuestión habrá que ejecutar el **paso3.py** de forma análoga a como se ha procedido con el **paso2.py**. Tras la ejecución de dicho paso aparecerá una lista con todos los ficheros disponibles en la carpeta qpython y se pedirá por pantalla un número asociado al archivo que se desea consultar.

Figura 3.18 Ejecución del **paso3** en Pydroid 3.

Tras la introducción de dicho número será necesario ejecutar el **paso4.py**, el cual mostrará por pantalla las gráficas de interés asociadas al temblor del paciente que se desea consultar. Además, a través del panel **Graphical program output** de Pydroid 3 se muestran los valores numéricos de interés del temblor del paciente que se mostraban tras la ejecución del **paso2**.

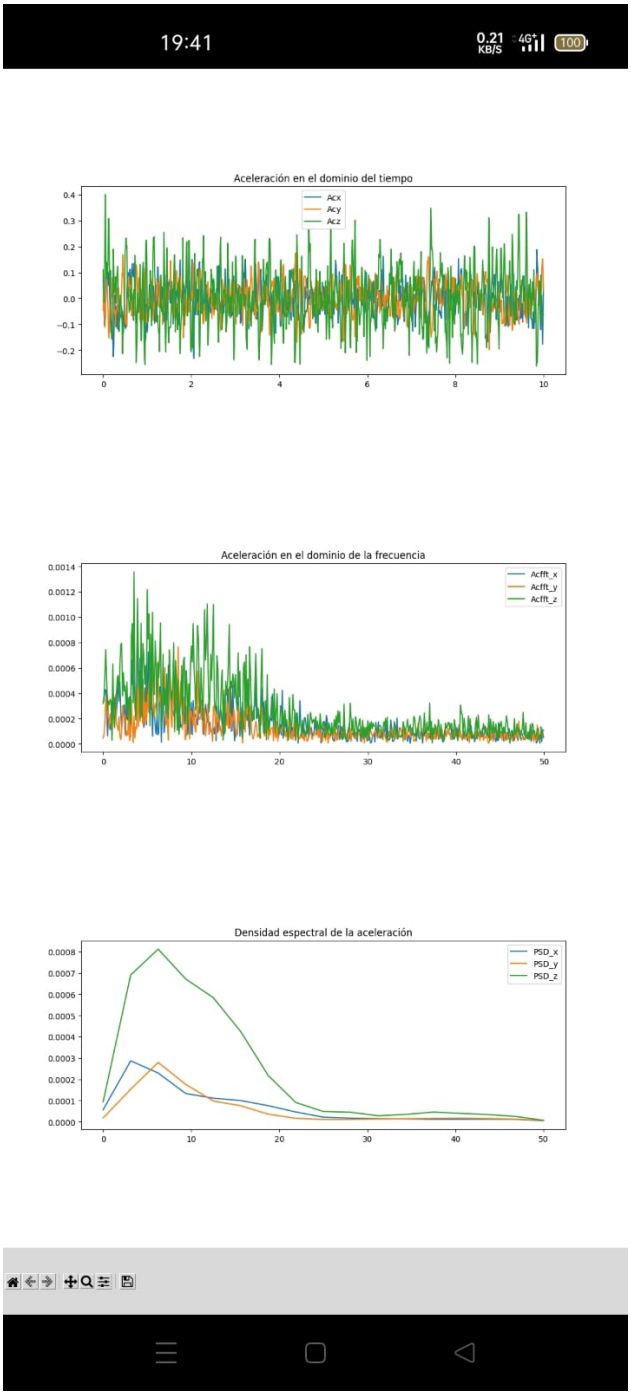

Figura 3.19 Resultados gráficos del **paso4** en Pydroid 3.

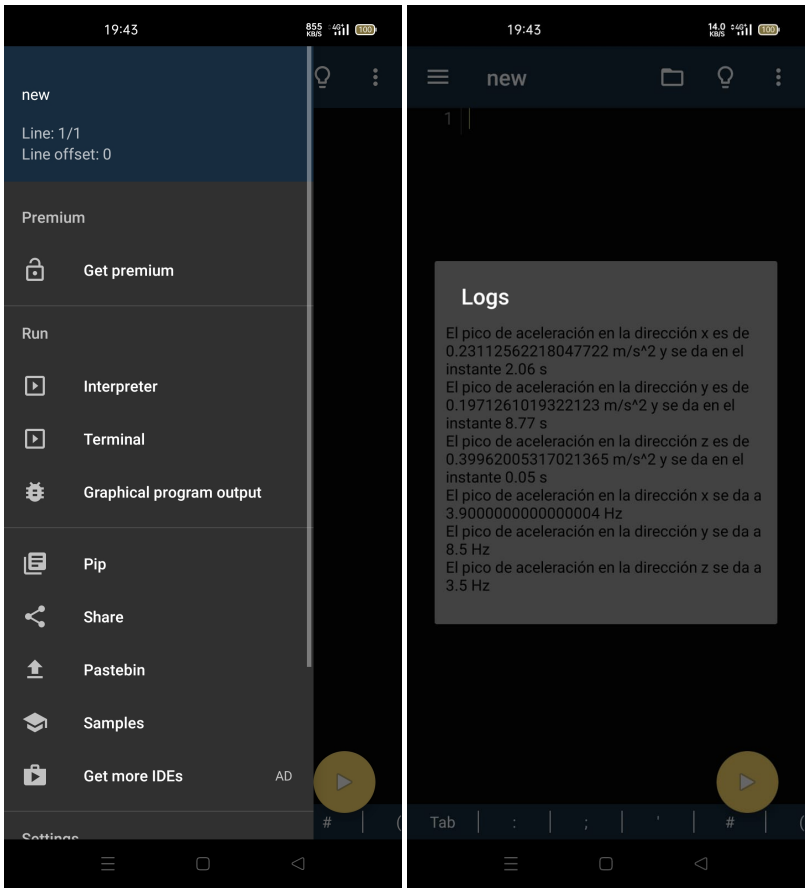

Figura 3.20 Resultados numéricos del **paso4** en Pydroid 3.

3.2.4 Puesta a punto de la red neuronal

Generación de datos sintéticos para el entrenamiento

Tal como se llevó a cabo en MATLAB, ante la escasez de datos procedentes de mediciones de temblores procedentes de pacientes reales será necesaria la generación de datos de naturaleza sintética. Para ello, en primer lugar, será necesario partir de datos procedentes de una medición real ejecutando el **paso1** en QPython 3L tal como ya se explicó previamente y llevando a cabo la medición de mi temblor en particular para emplearlo como punto de partida. Una vez hecho esto se procederá a ejecutar en Pydroid 3 el archivo **datsin**. Este archivo tiene un tiempo de ejecución de algunos minutos puesto que genera una gran cantidad de datos que tratan de emular el patrón ocurrido en la realidad. Sin embargo, debido a que es algo que no se tiene que estar ejecutando continuamente no es un gran problema. Al terminar la ejecución, estos datos quedan almacenados en una base de datos de sqlite, quedando así disponibles para el posterior proceso de entrenamiento de la red en cuestión.

Entrenamiento de la red neuronal

A diferencia de lo que ocurría en MATLAB Mobile, en este caso el proceso de entrenamiento puede completarse desde el teléfono móvil sin tener que recurrir a un ordenador. Para ello, será necesario ejecutar desde Pydroid 3 el archivo **train_datsin** que se encuentra en la carpeta qpython. Cuando se lleva a cabo este paso se muestran gráficamente los resultados del proceso de entrenamiento de la red neuronal y se generan los archivos asociados a la misma ya entrenada que se emplearán para las predicciones posteriores.

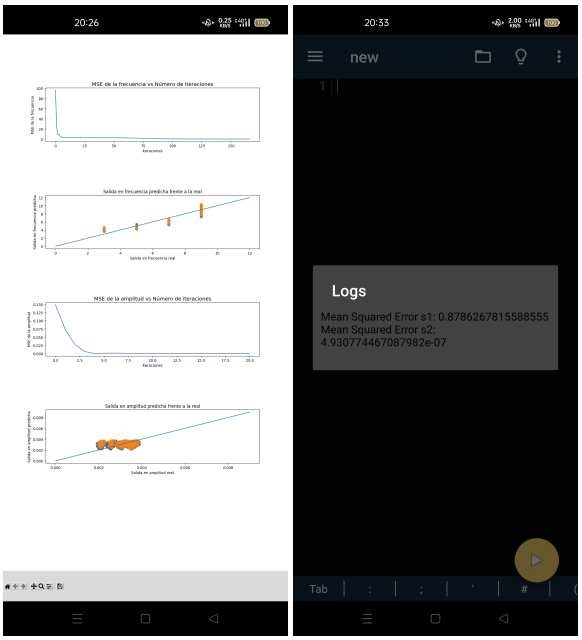

Figura 3.21 Resultados del entrenamiento de la red neuronal.

Predicciones de la red neuronal entrenada

Por último, mediante la ejecución del archivo **salida_pred** y la introducción por pantalla de los datos de la persona que se está analizando se lleva a cabo la predicción de las características del pico máximo del temblor que esa persona, atendiendo a sus características, debería tener. Una vez obtenidos dichos valores se comparan con los obtenidos a través de la medición del temblor del paciente a través del acelerómetro del teléfono móvil.

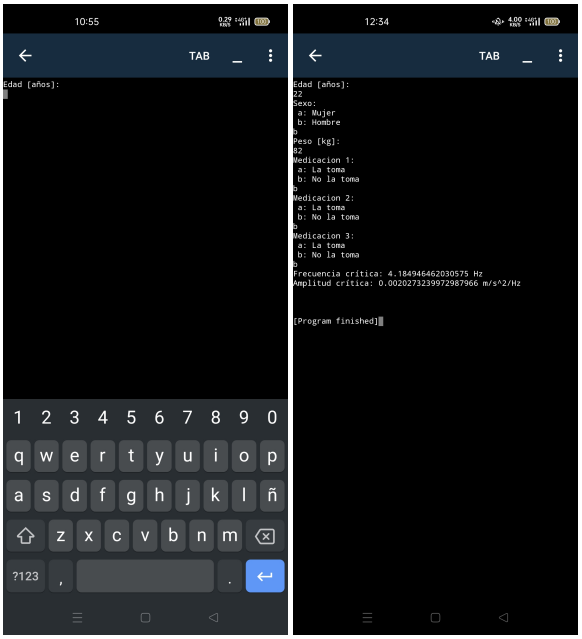

Figura 3.22 Ejecución y resultados del archivo **salida_pred** en Pydroid 3.

4 Resultados obtenidos y discusión de los mismos

En este capítulo se comentarán los resultados ofrecidos por la aplicación desarrollada a lo largo del presente proyecto en ambos lenguajes de programación.

4.1 Resultados obtenidos en MATLAB

4.1.1 Resultados obtenidos tras la ejecución del paso2

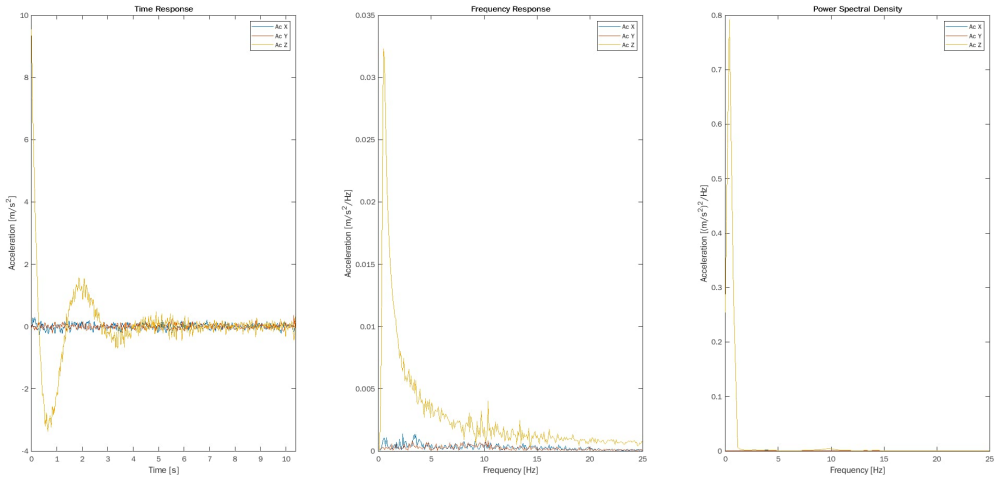

Figura 4.1 Gráficos obtenidos con el paso2 en MATLAB Mobile.

Tras la ejecución del fichero **paso2** en MATLAB Mobile se obtienen los gráficos asociados a la medición del temblor de un determinado paciente tanto en el dominio del tiempo como en el dominio de la frecuencia. Comentar que el paso al dominio de la frecuencia se ha llevado a cabo de dos maneras diferentes, con la transformada rápida de Fourier como puede verse en el gráfico central y con la densidad espectral de potencia tal como se observa en la gráfica de la derecha. Esto permite observar las características del pico máximo del temblor en el dominio de la frecuencia y facilita a un médico el análisis de dicho temblor con el fin de determinar su naturaleza.

Las características del temblor varían con la edad, el peso y la medicación que toma el paciente al que se le realizan las mediciones. Por un lado, un aumento de edad, de peso o de ambas simultáneamente produciría un aumento de la amplitud del temblor. Por otro lado, la toma de una medicación modificaría la frecuencia a

la que se da el pico máximo del temblor. Todo ello podría verse reflejado en los gráficos facilitados por la herramienta.

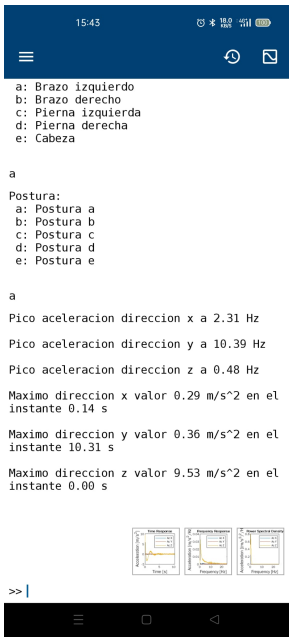

Figura 4.2 Valores críticos capturados en el paso2 en MATLAB Mobile.

Además, la ejecución del fichero **paso2** también devuelve los valores numéricos relacionados con el pico máximo del temblor tanto en el dominio de la frecuencia como en el dominio del tiempo tal como se ve en la imagen anterior. Estos valores serán posteriormente comparados con los predichos por la red neuronal con el fin de sacar conclusiones sobre la naturaleza del temblor tal como se verá más adelante en la presente sección.

4.1.2 Resultados obtenidos tras el entrenamiento de la red neuronal en MATLAB

La arquitectura de la red neuronal diseñada en MATLAB se presenta en la siguiente figura.

Figura 4.3 Arquitectura de la red neuronal diseñada en MATLAB.

Dicha arquitectura ha sido diseñada siguiendo un proceso de prueba y error hasta lograr conseguir unos resultados con errores admisibles. Para ello, se han ido modificando el número de capas ocultas de la red, así como el número de neuronas que componen dichas capas y se han ajustado otros parámetros como el número de épocas o el tipo de funciones de activación en cada capa hasta optimizar el resultado como ya se comentó en capítulos anteriores de la presente memoria.

El gráfico mostrado a continuación representa la caída del error cuadrático medio con el aumento del número de épocas, es decir, del número de pasadas de los datos por la red para ajustar los pesos de ponderación. Esta caída se produce hasta que en una cierta época se alcanza el valor mínimo del error y se detiene la ejecución. Siempre hay que tener presente en este gráfico que se produzca la caída del error tanto para los datos empleados en la fase de entrenamiento como en la fase de validación. Esto está relacionado con evitar la aparición del fenómeno conocido como overfitting, en el cual el error decae cada vez más en la fase de entrenamiento pero a costa de aumentar el error en la fase de validación.

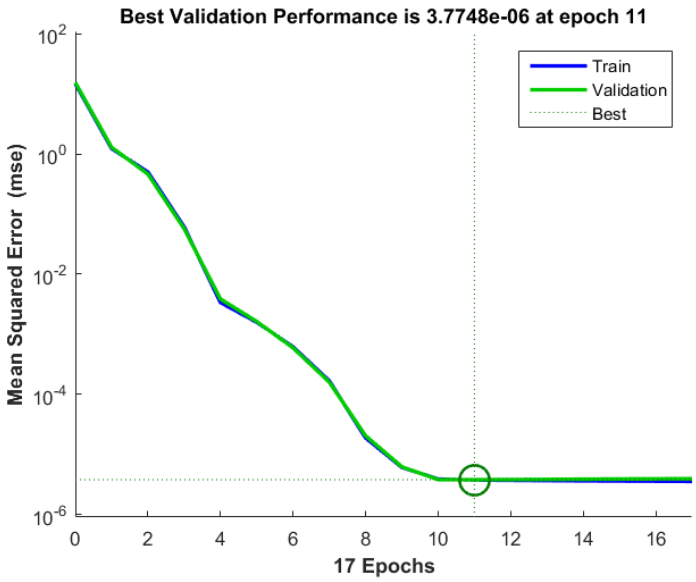

Figura 4.4 Error cuadrático medio vs Número de iteraciones en MATLAB.

El gráfico de la siguiente figura representa el valor de la frecuencia a la que se da el pico máximo del temblor en el dominio de la frecuencia que se obtiene en la realidad frente al que predice la red neuronal tras pasar por el proceso de entrenamiento. Lo ideal sería que ambas coincidiesen siguiendo la tendencia de la recta representada. Sin embargo, la existencia de errores hace que las predicciones no se correspondan exactamente con la realidad y esto provoca que no se siga exactamente la tendencia previamente mencionada.

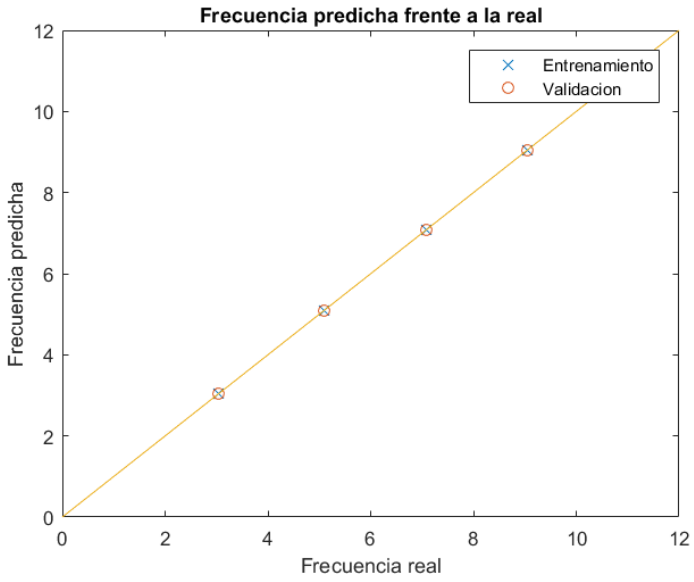

Figura 4.5 Frecuencia predicha por la red vs Frecuencia real facilitada a la red.

Análogamente a lo representado en el gráfico comentado en el párrafo previo, en el último gráfico se representa la otra salida predicha, en este caso la amplitud del pico máximo del temblor en el dominio de la frecuencia.

Por último, comentar que los errores son de un orden de magnitud tan pequeño que se pueden tomar las predicciones como una representación muy cercana a la realidad.

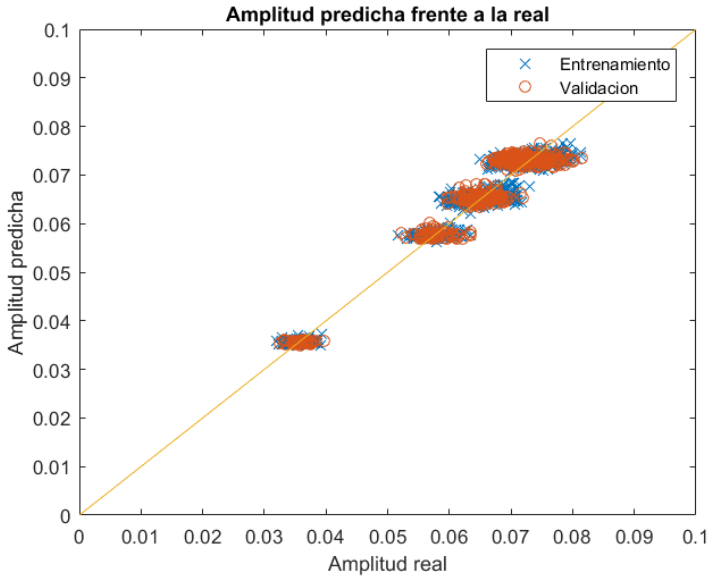

Figura 4.6 Amplitud predicha por la red vs Amplitud real facilitada a la red.

4.1.3 Resultados obtenidos en las predicciones de la red neuronal en MATLAB Mobile

Tal como puede verse en la figura, al recibir los datos asociados a la edad, el peso, el sexo y las medicaciones tomadas por el paciente (1,2,3), la red neuronal es capaz de devolver la frecuencia a la que se da el pico máximo del temblor y la amplitud del mismo con un error asumible.

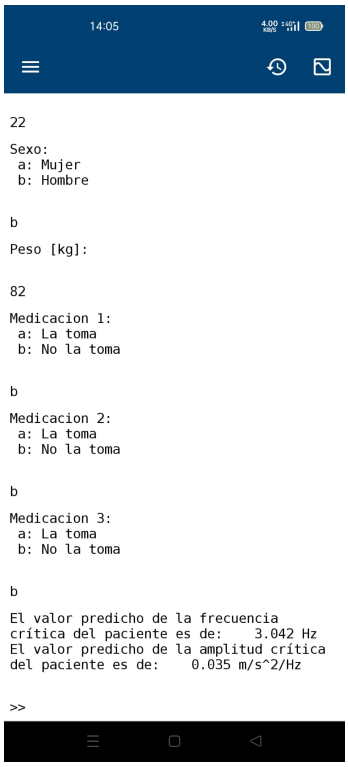

Figura 4.7 Resultados obtenidos en las predicciones de la red neuronal en MATLAB Mobile.

Comentar que al haber llevado a cabo el proceso de entrenamiento de la red neuronal empleando datos de naturaleza sintética, es decir, datos generados a través de la adición de ruido sobre datos procedentes de pacientes reales, los resultados arrojados por la misma no tienen una interpretación física lógica. Sin embargo, cuando se disponga de la cantidad suficiente de datos procedentes de las mediciones de los temblores de pacientes reales, se podrá llevar a cabo un nuevo proceso de entrenamiento tras el cual se podrán obtener valores útiles a la hora de analizar la naturaleza de los temblores de un determinado paciente.

4.1.4 Resultados predichos vs Resultados obtenidos tras mediciones en MATLAB Mobile

Con el fin de ilustrar la utilidad de dicha aplicación en la detección de la naturaleza de los temblores de los pacientes se presenta el siguiente caso.

Por un lado, en la imagen de la izquierda se pueden ver los resultados de una medición realizada a un paciente de 22 años de edad, de género masculino y 82 kg de peso que toma únicamente la medicación de tipo 1. En ella se puede apreciar que la frecuencia del pico máximo del temblor asociado a dicho paciente es de 4.79 Hz. Por otro lado, la imagen de la derecha presenta los resultados de la predicción del temblor de dicho paciente llevada a cabo por la red neuronal desarrollada en el presente proyecto. En ella se puede ver que la frecuencia predicha del pico máximo del temblor asociado a dicho paciente es de 5.095 Hz. Por tanto, se puede concluir que, dado que ambas frecuencias son prácticamente idénticas, la causa del temblor en dicho paciente es consecuencia de la toma de la medicación de tipo 1. Sin embargo, si existiera una disparidad evidente entre ambos valores, esto podría indicar que la presencia de dicho temblor se debe a la existencia de algún tipo de enfermedad neurológica.

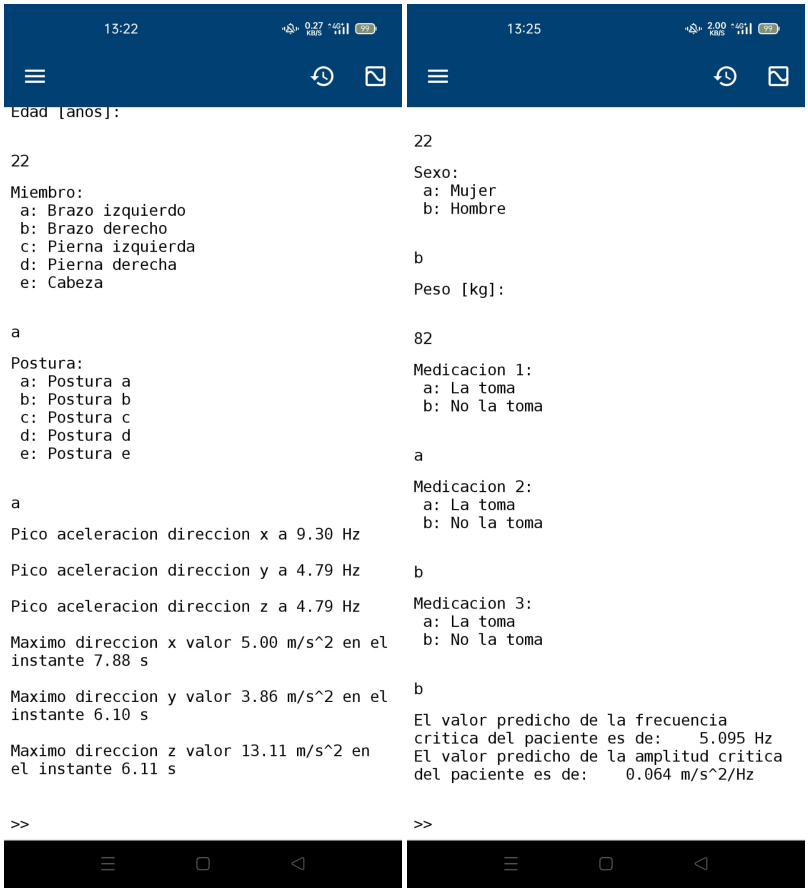

Figura 4.8 Resultados obtenidos tras mediciones vs Resultados predichos .

4.2 Resultados obtenidos en Python

4.2.1 Resultados numéricos obtenidos tras la ejecución del paso2

Como se observa en la figura, una vez que se lleva a cabo la ejecución del fichero **paso2** se obtienen valores numéricos relacionados con los picos máximos del temblor del paciente al que se le realiza la medición tanto en el dominio del tiempo como en el dominio de la frecuencia.

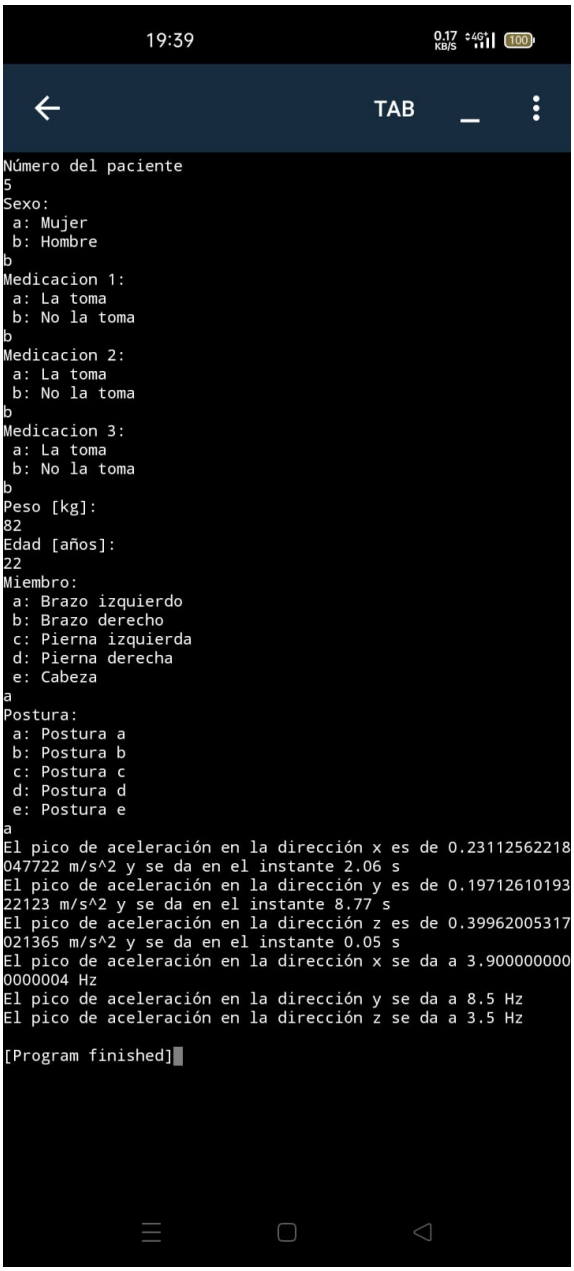

Figura 4.9 Resultados numéricos obtenidos tras la ejecución del **paso2** en Pydroid 3.

4.2.2 Resultados gráficos obtenidos tras la ejecución del paso4

Tras la ejecución del fichero **paso4** en Pydroid 3 se obtienen los gráficos asociados al temblor registrado a través del acelerómetro del teléfono móvil tras realizar la medición al paciente tanto en el dominio del tiempo como en el dominio de la frecuencia. Comentar que el paso al dominio de la frecuencia se ha llevado a cabo de dos maneras diferentes, con la transformada rápida de Fourier como puede verse en el gráfico central y con la densidad espectral de potencia tal como se observa en la gráfica inferior. Esto permite observar las características del pico máximo del temblor en el dominio de la frecuencia y facilita a un médico el análisis de dicho temblor con el fin de determinar su naturaleza.

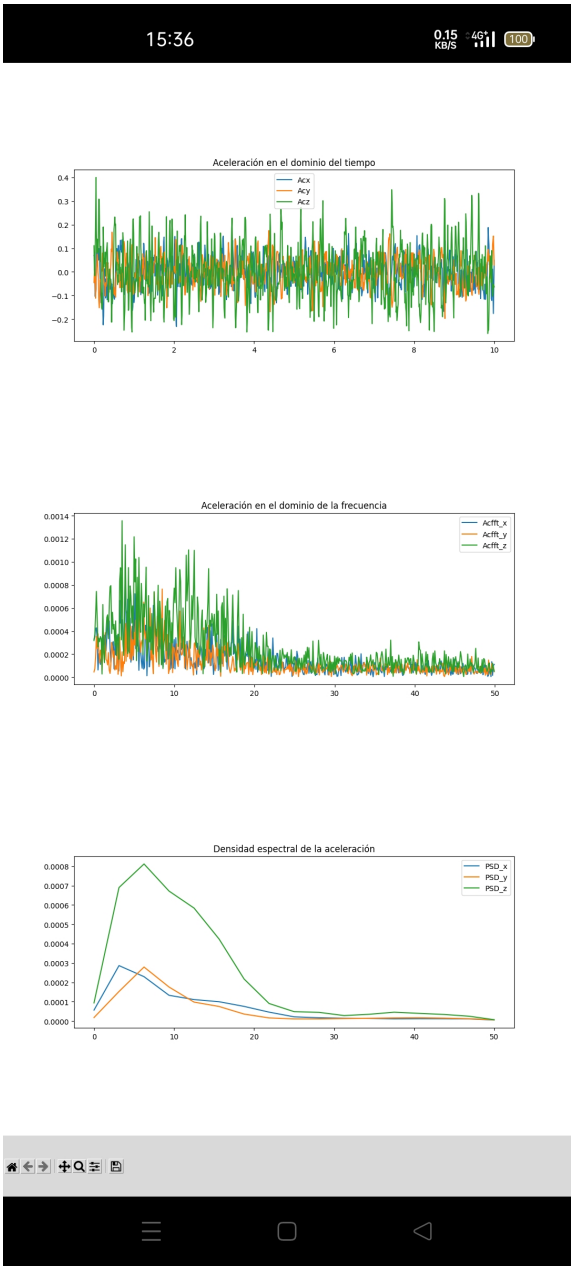

Figura 4.10 Resultados gráficos obtenidos tras la ejecución del **paso4** en Pydroid 3.

Las características de dicho temblor varían con la edad, el peso y la medicación que toma el paciente al que se le realizan las mediciones. Por un lado, un aumento de edad, de peso o de ambas simultáneamente produciría un aumento de la amplitud del temblor. Por otro lado, la toma de una medicación modificaría la frecuencia a la que se da el pico máximo del temblor.

4.2.3 Resultados obtenidos tras el entrenamiento de la red neuronal en Pydroid 3

Con el fin de mejorar las predicciones realizadas por la red neuronal en `Python` se llevó a cabo una división del problema. Por un lado, se llevó a cabo el proceso de entrenamiento de una red neuronal con un cierto número de capas ocultas y con un tipo determinado de función de activación con el fin de mejorar la predicción de la frecuencia asociada al pico del temblor de un determinado paciente. Por otro lado, se realizó una fase de entrenamiento de otra red neuronal con un número de capas y un tipo de función de activación distintas para llevar a cabo una mejora en las predicciones relacionadas con la amplitud del pico del temblor de un determinado paciente. Cada una de estas redes, una vez entrenadas, se guardaron con el fin de realizar predicciones a posteriori. El comportamiento de cada una de las redes neuronales tras ser entrenadas se puede apreciar en la imagen de la izquierda. Como puede verse, en ambos casos, el error cuadrático medio disminuye con el número de iteraciones hasta alcanzar un orden de magnitud aceptable. Por último, comentar que en la imagen de la derecha se puede ver una estimación de los errores cometidos en cada una de las predicciones por separado. A pesar de que no son idénticamente cero, son errores asumibles para la aplicación del proyecto en cuestión.

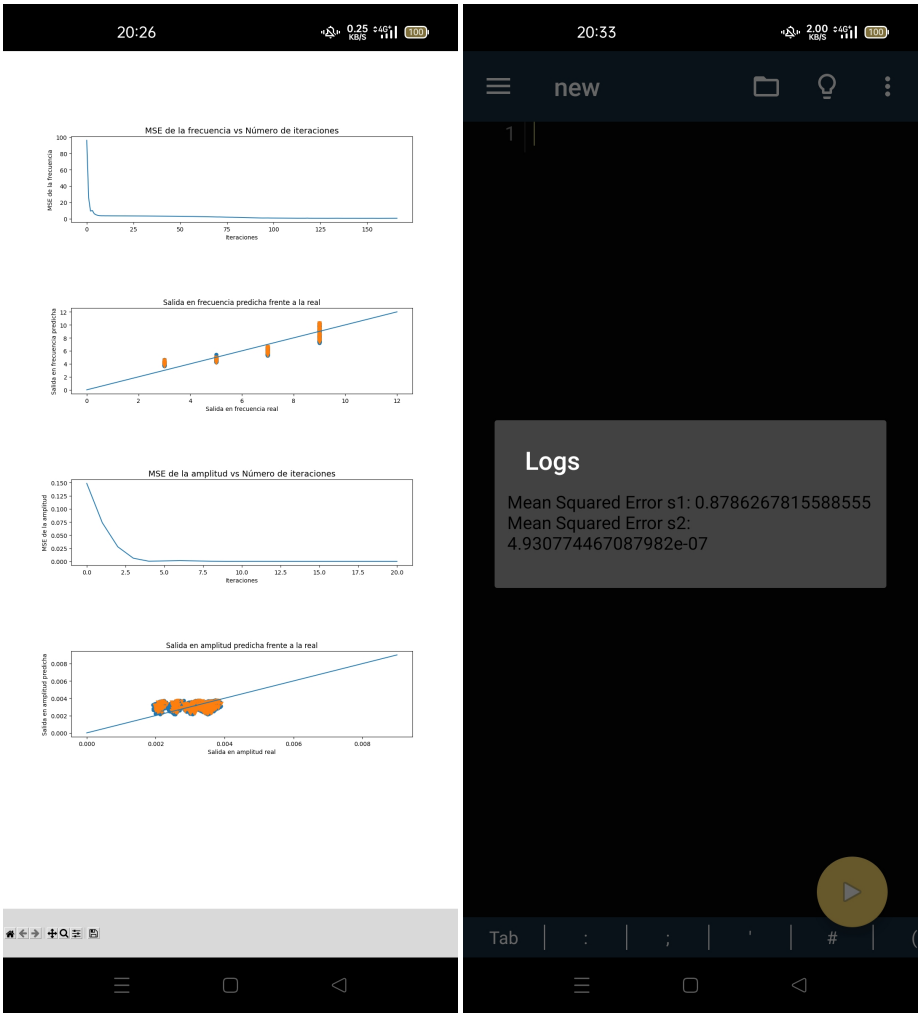

Figura 4.11 Resultados obtenidos tras el entrenamiento de la red neuronal en Pydroid 3.

4.2.4 Resultados obtenidos en las predicciones de la red neuronal en Pydroid 3

Tal como puede verse en la figura, al recibir los datos asociados a la edad, el peso, el sexo y y las medicaciones que toma el paciente (1,2,3), las redes neuronales entrenadas anteriormente, son capaces de devolver la frecuencia a la que se da el pico máximo del temblor y la amplitud del mismo con un error asumible.

Figura 4.12 Resultados obtenidos en las predicciones de la red neuronal en Pydroid 3.

Comentar que al haber llevado a cabo el proceso de entrenamiento de la red neuronal empleando datos de naturaleza sintética, es decir, datos generados a través de la adición de ruido sobre datos procedentes de pacientes reales, los resultados arrojados por la misma no tienen una interpretación física lógica. Sin embargo, cuando se disponga de la cantidad suficiente de datos procedentes de las mediciones de los temblores de pacientes reales, se podrá llevar a cabo un nuevo proceso de entrenamiento tras el cual se podrán obtener valores útiles a la hora de analizar la naturaleza de los temblores de un determinado paciente.

4.2.5 Resultados predichos vs Resultados obtenidos tras mediciones en Pydroid 3

Con el fin de ilustrar la utilidad de dicha aplicación en la detección de la naturaleza de los temblores de los pacientes se presenta el siguiente caso.

Por un lado, en la imagen de la izquierda se pueden ver los resultados de una medición realizada a un paciente de 22 años de edad, de género masculino y 82 kg de peso que toma únicamente la medicación de tipo 1. En ella se puede apreciar que la frecuencia del pico máximo del temblor asociado a dicho paciente es de 4.5 Hz. Por otro lado, la imagen de la derecha presenta los resultados de la predicción del temblor de dicho

paciente llevada a cabo por la red neuronal desarrollada en el presente proyecto. En ella se puede ver que la frecuencia predicha del pico máximo del temblor asociado a dicho paciente es de 4.55 Hz. Por tanto, se puede concluir que, dado que ambas frecuencias son prácticamente idénticas, la causa del temblor en dicho paciente es consecuencia de la toma de la medicación de tipo 1. Sin embargo, si existiera una disparidad evidente entre ambos valores, esto podría indicar que la presencia de dicho temblor se debe a la existencia de algún tipo de enfermedad neurológica.

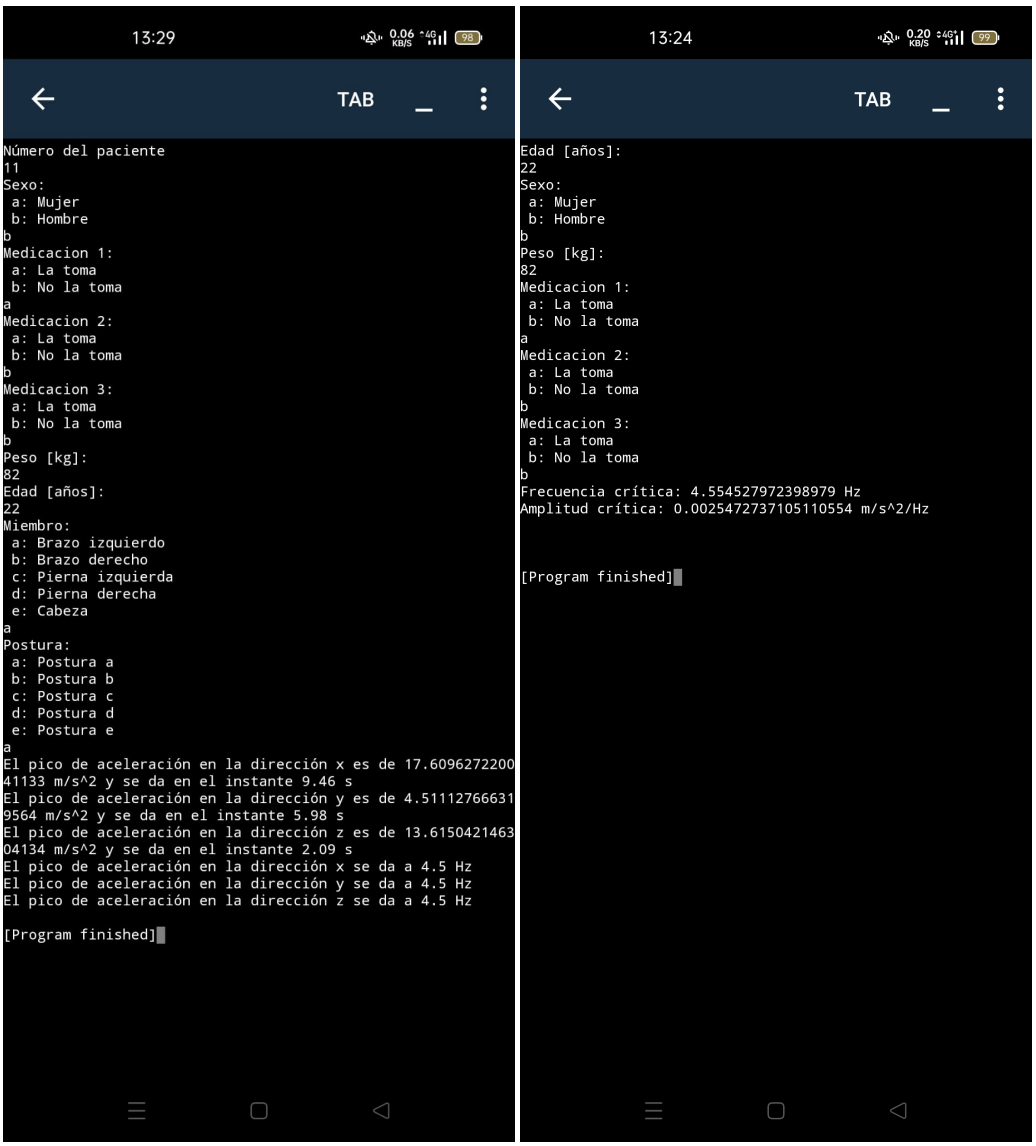

Figura 4.13 Resultados obtenidos tras mediciones vs Resultados predichos .

5 Conclusiones y desarrollos futuros

Finalmente, en este capítulo se comentarán las conclusiones extraídas durante el desarrollo del proyecto y se plantearán posibles mejoras y desarrollos de cara al futuro.

Durante el desarrollo del presente proyecto se ha podido comprobar que las aplicaciones móviles empleadas para ejecutar código tanto en `MATLAB` como en `Python` presentan ciertas limitaciones que han hecho necesario la adopción de soluciones técnicas distintas a las que se hubiesen planteado en un desarrollo completo en ordenador.

El primero de los grandes objetivos a alcanzar durante este proyecto era la realización de una aplicación que fuese capaz de medir los temblores de un paciente a través del acelerómetro de un teléfono móvil para después almacenarlos y extraer información de los mismos. Por otro lado, el segundo de los objetivos estaba relacionado con la realización de una red neuronal que fuera capaz de predecir los valores de interés del temblor de un paciente determinado. Ambos objetivos se han desarrollado adecuadamente durante el proyecto. Sin embargo, en ambos existen posibilidades de mejora que se comentarán a continuación.

Una de las mejoras futuras a introducir está relacionada con el perfeccionamiento de la capacidad predictiva de la red neuronal desarrollada en el presente proyecto. Dado que no se disponía de la cantidad suficiente de datos de temblores medidos a pacientes reales en el momento en que se llevó a cabo el proceso de entrenamiento de la red neuronal se tuvo que recurrir a la generación de temblores sintéticos, por tanto, una vez obtenida la cantidad necesaria de datos procedente de temblores medidos a pacientes reales habría que proceder a entrenar de nuevo la red neuronal con la finalidad de afinar su capacidad de predecir los valores de interés asociados a los temblores de los pacientes analizados. Además, sería conveniente introducir en la red más parámetros de entrada que también afectan a la magnitud de los temblores tales como el miembro sobre el que se lleva a cabo la medición o la posición de dicho miembro a la hora de realizarla.

Otra mejora a implementar en un futuro sería la generación de un archivo .apk que integre todos los códigos desarrollados en el presente proyecto de tal forma que facilite el empleo de la aplicación por parte del usuario. De esta forma, el usuario únicamente tendría que descargar el archivo .apk que ya traería incorporados todos los códigos y llevar a cabo una única instalación sin tener que preocuparse por descargar todos los archivos por separado e introducirlos en un lugar determinado o por la ejecución individual de cada uno de los pasos.

Por último, sería conveniente estandarizar la forma en la que se realizan las mediciones a los pacientes con el fin de evitar que factores externos a la medida influyan negativamente en la interpretación de los resultados. Por tanto, sería de gran interés realizar el ensayo siempre con el mismo tipo de smartphone para evitar posibles efectos inducidos por la variación del peso del mismo así como unificar las posturas de los miembros del cuerpo sobre los que se realizan dichas mediciones, es decir, asociar las posiciones de cada miembro a unos ángulos determinados en los que se sepa que existe tendencia de aparición de temblores basándose en la experiencia previa.

Apéndice A
Subida y descarga de archivos desde MATLAB Drive

Para llevar a cabo la subida y descarga de archivos desde MATLAB Drive desde un ordenador será necesario acceder a [27] e iniciar sesión.

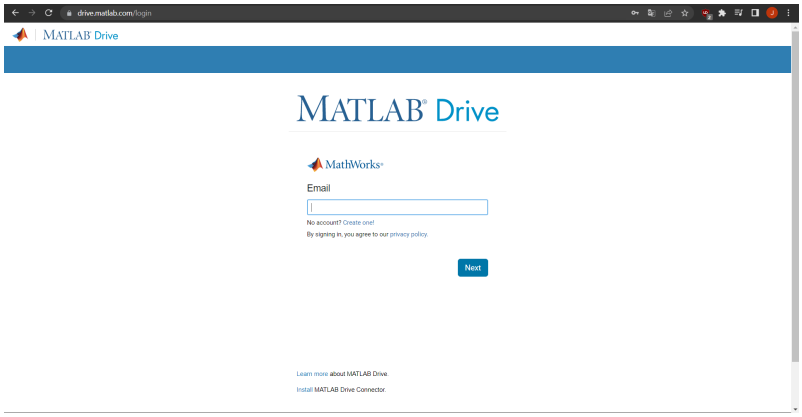

Figura A.1 Inicio de sesión en MATLAB Drive desde un ordenador.

Una vez llevado a cabo el inicio de sesión aparecerán los archivos ya almacenados en el espacio personal tal como se muestra en la figura.

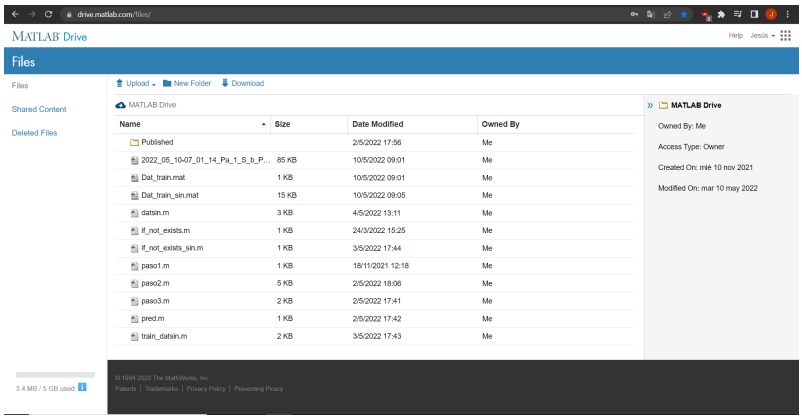

Figura A.2 Archivos almacenados en MATLAB Drive.

Para llevar a cabo la subida de un archivo a MATLAB Drive desde el ordenador simplemente habrá que dirigirse al icono representado con una flecha azul apuntando hacia arriba que dice Upload y explorar la carpeta donde se encuentre el archivo a subir. En cuanto a lo relacionado con la descarga de archivos desde

MATLAB Drive existen dos opciones según lo que se desee llevar a cabo. Por un lado, si lo que se desea es descargar todos los archivos almacenados en el espacio de MATLAB Drive lo más conveniente es dirigirse al icono representado con una flecha azul apuntando hacia abajo que dice Download. Una vez seleccionado dicho icono se llevará a cabo la descarga de una carpeta comprimida con todos los archivos que en ese momento se encuentren subidos a MATLAB Drive. Por otro lado, si lo que se desea es descargar en el ordenador un único archivo de los que se encuentran en MATLAB Drive simplemente habría que hacer click derecho sobre el archivo en cuestión y seleccionar Download.

Apéndice B
Instalación de librerías en Pydroid 3

En primer lugar, será necesario abrir la aplicación Pydroid 3. Al hacerlo, aparecerá la interfaz que aparece en la figura.

Figura B.1 Panel de inicio de Pydroid 3.

Una vez hecho esto habrá que abrir el menú desplegable que aparece en la parte superior izquierda tal como se muestra en la imagen.

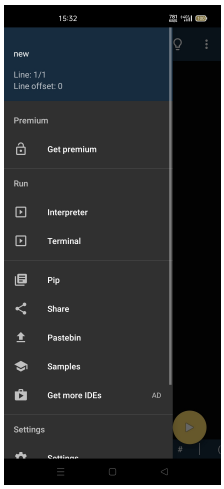

Figura B.2 Menú desplegable de Pydroid 3.

A continuación, será necesario dirigirse al subapartado que dice Pip, seleccionar SEARCH LIBRARIES e introducir por pantalla el nombre de la librería que desea instalarse.

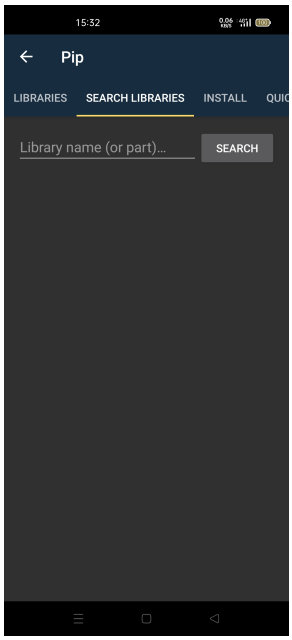

Figura B.3 Panel de búsqueda de librerías en Pydroid 3.

Para ilustrar el proceso se llevará a cabo la instalación de la librería numpy a modo de ejemplo. Para ello, se introducirá numpy por pantalla en el panel de búsqueda mostrado en la imagen.

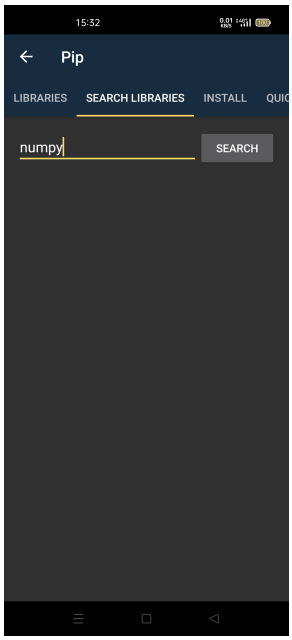

Figura B.4 Ejemplo de búsqueda de una librería en Pydroid 3.

Tras llevar a cabo lo anterior se pulsa el botón SEARCH y aparecerá el listado mostrado en la imagen.

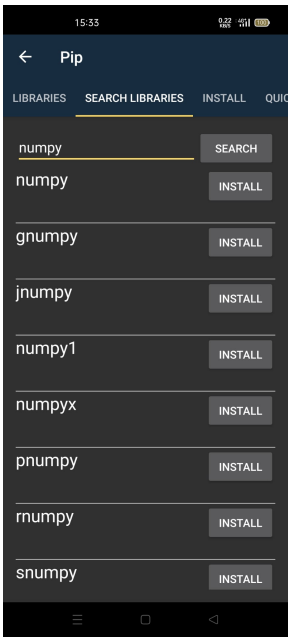

Figura B.5 Resultado de la búsqueda de una librería en Pydroid 3.

Finalmente, se pulsa el botón INSTALL para llevar a cabo la instalación de la librería en cuestión, en este ejemplo, de numpy. Una vez hecho esto aparecerá por pantalla el mensaje Complete que indica que la instalación se ha llevado a cabo con éxito.

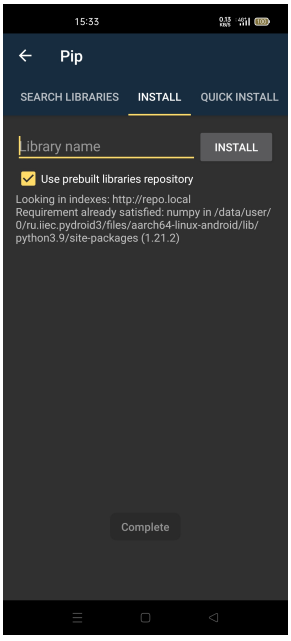

Figura B.6 Mensaje de instalación completada de una librería en Pydroid 3.

Apéndice C
Códigos desarrollados durante el proyecto

C.1 Códigos desarrollados en MATLAB

```
%Paso if_not_exists en Matlab
x=[];
y_d=[];
mat_file=matfile('Dat_train.mat','Writable',true);
save('Dat_train.mat','x','y_d');
```

Código C.1 Archivo **if_not_exists** en MATLAB.

```
%Paso if_not_exists_sin en Matlab
x=[];
y_d=[];
mat_file=matfile('Dat_train_sin.mat','Writable',true);
save('Dat_train_sin.mat','x','y_d');
```

Código C.2 Archivo **if_not_exists_sin** en MATLAB.

```
%paso1 en Matlab
m=mobiledev();
```

Código C.3 Archivo **paso1** en MATLAB.

```
%paso2 en Matlab
load('Dat_train.mat');
prompt = 'Numero de paciente\n';
num = input(prompt);

if isempty(num)
    error('Debe introducir el numero de paciente')
end

if exist('num2','var') == 0
    num2=num;
end

if exist('sexo','var') == 0||num~=num2
    prompt = 'Sexo:\n a: Mujer \n b: Hombre\n';

    sexo = input(prompt,'s');

    if sexo=='a'
        sexo2=0;
    elseif sexo=='b'
        sexo2=1;
    else
        sexo2=2;
    end

    if (0>sexo2)||(sexo2>1)
        error('Debe introducir un codigo segun la lista anterior')
    end
end
if exist('medicacion1','var') == 0||num~=num2
    prompt = 'Medicacion 1:\n a: La toma \n b: No la toma\n';

    medicacion1 = input(prompt,'s');

    if medicacion1=='a'
        medicacion12=1;
    elseif medicacion1=='b'
        medicacion12=0;
    else
        medicacion12=2;
    end

    if (0>medicacion12)||(medicacion12>1)
        error('Debe introducir un codigo segun la lista anterior')
    end
end

if exist('medicacion2','var') == 0||num~=num2
    prompt = 'Medicacion 2:\n a: La toma \n b: No la toma\n';

    medicacion2 = input(prompt,'s');

    if medicacion2=='a'
        medicacion22=1;
    elseif medicacion2=='b'
        medicacion22=0;
    else
        medicacion22=2;
    end

    if (0>medicacion22)||(medicacion22>1)
        error('Debe introducir un codigo segun la lista anterior')
    end
end

if exist('medicacion3','var') == 0||num~=num2
    prompt = 'Medicacion 3:\n a: La toma \n b: No la toma\n';

    medicacion3 = input(prompt,'s');

```

```
    if medicacion3=='a'
        medicacion32=1;
    elseif medicacion3=='b'
        medicacion32=0;
    else
        medicacion32=2;
    end

    if (0>medicacion32)||(medicacion32>1)
        error('Debe introducir un codigo segun la lista anterior')
    end
end

if exist('peso','var') == 0||num~=num2
    prompt = 'Peso [kg]:\n';

    peso = input(prompt);
    peso=round(peso);
end


if exist('edad','var') == 0||num~=num2
    prompt = 'Edad [años]:\n';

    edad = input(prompt);
end

if num~=num2
    num2=num;

end
prompt = 'Miembro:\n a: Brazo izquierdo\n b: Brazo derecho\n c: Pierna izquierda\n d: Pierna
     derecha\n e: Cabeza\n';

org = input(prompt,'s');

if org=='a'
    org2=1;
elseif org=='b'
    org2=2;
elseif org=='c'
    org2=3;
elseif org=='d'
    org2=4;
elseif org=='e'
    org2=5;
else
    org2=0;
end

if (1>org2)||(org2>5)
    error('Debe introducir un codigo segun la lista anterior')
end

prompt = 'Postura:\n a: Postura a\n b: Postura b\n c: Postura c\n d: Postura d\n e: Postura e\n';

pos = input(prompt,'s');

if pos=='a'
    pos2=1;
elseif pos=='b'
    pos2=2;
elseif pos=='c'
    pos2=3;
elseif pos=='d'
    pos2=4;
elseif pos=='e'
    pos2=5;
else
    pos2=0;
end
```

```
142
143 if (1>pos2)||(pos2>5)
144     error('Debe introducir un codigo segun la lista anterior')
145 end
146
147
148 %Aceleracion
149 [a,t]=accellog(m);
150 fs=1/(t(2)-t(1));
151
152 ChebyOrder=3;
153 ChebyRipple=0.1;
154 ChebyFreq=0.5;
155 [Bf,Af]=cheby1(ChebyOrder,ChebyRipple,ChebyFreq/fs*2,'high');
156 a=filter(Bf,Af,a);
157
158 af=fft(a)/fs;
159 nfreq=size(af,1);
160 df=fs/nfreq;
161 freq=df:df:fs;
162
163 [pxx,f] = pwelch(a,[],[],[],fs);
164
165 %picos acelerometro
166 dir=['x';'y';'z'];
167 data=af;
168 for idir=1:3
169     [~,loc]=findpeaks(abs(squeeze(data(:,idir))));
170     [ampmax,locmax]=max(abs(squeeze(data(loc,idir))));
171     freqaux=freq(loc);
172     freqmax=freqaux(locmax);
173     freqmaxvector(idir)=freqmax;
174     ampmaxvector(idir)=ampmax/fs;
175     disp(sprintf('Pico aceleracion direccion %s a %4.2f Hz\n',dir(idir,:),freqmax))
176 end
177
178 %Hallamos la frecuencia crítica y amplitud crítica del paciente sintético en cuestión
179 freqcritica=max(freqmaxvector);
180 ampcritica=max(ampmaxvector);
181 clear data
182
183 data=a;
184 tiempo=t;
185 for idir=1:3
186     [maxa,locmax]=max(abs(squeeze(data(:,idir))));
187     disp(sprintf('Maximo direccion %s valor %4.2f m/s^2 en el instante %4.2f s\n',dir(idir,:),
      maxa,tiempo(locmax)))
188 end
189 clear tiempo data
190
191 filename=sprintf(datestr(now,'yyyy_mm_dd-HH_MM_SS'));
192
193 save(fullfile(strcat(filename,'_Pa_',int2str(num),'_S_',sexo,'_Pe_',int2str(peso),'_M_',org,'_Po_'
      ,pos)))
194
195 if exist('m','var') == 1
196     clear m
197     m=mobiledev();
198 end
199
200 close all
201
202 figure(1)
203 plot(t,a);
204 xlabel('Time [s]','FontSize',12);
205 ylabel('Acceleration [m/s^2]','FontSize',12)
206 title('Time Response')
207 legend('Ac X','Ac Y','Ac Z')
208 xlim([0 t(end)])
209
210 figure(2)
```

```
plot(freq(1:(nfreq)),abs(squeeze(af(1:(nfreq),:)))/fs);
xlabel('Frequency [Hz]','FontSize',12);
ylabel('Acceleration [m/s^2/Hz]','FontSize',12)
title('Frequency Response')
legend('Ac X','Ac Y','Ac Z')
xlim([0 25])

figure(3)
plot(f,abs(pxx));
xlabel('Frequency [Hz]','FontSize',12);
ylabel('Acceleration [(m/s^2)^2/Hz]','FontSize',12);
title('Power Spectral Density')
legend('Ac X','Ac Y','Ac Z')
xlim([0 25])

%Armamos nuestros vectores de datos adecuadamente convertidos.
x=[x;[edad,sexo2,peso,medicacion12,medicacion22,medicacion32]];
y_d=[y_d;[freqcritica,ampcritica]];

save('Dat_train.mat','x','y_d','-append');
```

Código C.4 Archivo **paso2** en MATLAB.

```
%paso3 en Matlab
clear

files = dir;
fileIndex = find(~[files.isdir]);

name = cell(length(fileIndex),1);

for iname=1:length(fileIndex)
    name{iname}=char({strcat(files(fileIndex(iname)).name)});
end

for iname=1:length(fileIndex)
    disp(sprintf('%s: %s\n',int2str(iname),char(name(iname))))
end

prompt = 'Seleccione el numero del fichero de resultados que quiere analizar:\n';
num = input(prompt);

load(char(name(num)))

for idir=1:3
    disp(sprintf('Pico aceleracion direccion %s a %4.2f Hz\n',dir(idir,:),freqmax))
end

for idir=1:3
    disp(sprintf('Maximo direccion %s valor %4.2f m/s^2 en el instante %4.2f s\n',dir(idir,:),
     maxa,t(locmax)))
end
close all

figure(1)
plot(t,a);
xlabel('Time [s]','FontSize',12);
ylabel('Acceleration [m/s^2]','FontSize',12)
title('Time Response')
legend('Ac X','Ac Y','Ac Z')
xlim([0 t(end)])

figure(2)
plot(freq(1:(nfreq)),abs(squeeze(af(1:(nfreq),:)))/fs);
xlabel('Frequency [Hz]','FontSize',12);
ylabel('Acceleration [m/s^2/Hz]','FontSize',12)
title('Frequency Response')
legend('Ac X','Ac Y','Ac Z')
xlim([0 25])

figure(3)
plot(f,abs(pxx));
xlabel('Frequency [Hz]','FontSize',12);
ylabel('Acceleration [(m/s^2)^2/Hz]','FontSize',12);
title('Power Spectral Density')
legend('Ac X','Ac Y','Ac Z')
xlim([0 25])
```

Código C.5 Archivo **paso3** en MATLAB.

```
%Paso datsin en Matlab
for sexo=0:1
    for medicacion1=0:1
        for medicacion2=0:1
            for medicacion3=0:1
                for edadmin=20:10:90
                    for pesomin=40:10:150
load('Dat_train_sin.mat');
%Aceleracion
[a,t]=accellog(m);
fs=1/(t(2)-t(1));
ChebyOrder=3;
ChebyRipple=0.1;
ChebyFreq=0.5;
[Bf,Af]=cheby1(ChebyOrder,ChebyRipple,ChebyFreq/fs*2,'high');
a=filter(Bf,Af,a);
%Calculo de los picos del acelerometro.
dir=['x';'y';'z'];
data=a;
tiempo=t;
for idir=1:3
    [maxa,locmax]=max(abs(squeeze(data(:,idir))));
    acmax(idir)=maxa;
end
clear tiempo data

edadmax=edadmin+10;
pesomax=pesomin+10;

secedad=edadmin:edadmax;
edad=randsample(secedad,1);

secpeso=pesomin:pesomax;
peso=randsample(secpeso,1);

ke=(edad-edadmin)/(edadmax-edadmin); %Cte de ponderacion de la edad
kp=(peso-pesomin)/(pesomax-pesomin); %Cte de ponderacion del peso
kpe=1+0.1*kp+0.1*ke;
k1=medicacion1; %Cte de pico de la medicacion 1
k2=medicacion2; %Cte de pico de la medicacion 2
k3=medicacion3; %Cte de pico de la medicacion 3


%Añadimos ruido para generar datos sinteticos que servirán para entrenar a la red neuronal.
for i=1:3
a(:,i)=a(:,i)+0.05*acmax(i)*(-1+2*rand(length(a(:,i)),1))+ 0.1*k1*acmax(i)*sin(2*pi*5*t)+ 0.1*k2*
     acmax(i)*sin(2*pi*7*t)+ 0.1*k3*acmax(i)*sin(2*pi*9*t);
a(:,i)=kpe*a(:,i);
end
%Pasamos al dominio de la frecuencia esos datos sinteticos.
af=fft(a)/fs;
nfreq=size(af,1);
df=fs/nfreq;
freq=df:df:fs;
[pxx,f] = pwelch(a,[],[],[],fs);
data=af;
for idir=1:3
    [~,loc]=findpeaks(abs(squeeze(data(:,idir))));
    [ampmax,locmax]=max(abs(squeeze(data(loc,idir))));
    freqaux=freq(loc);
    freqmax=freqaux(locmax);
    freqmaxvector(idir)=freqmax;
    ampmaxvector(idir)=ampmax/fs;
end
%Hallamos la frecuencia critica y amplitud critica del paciente sintetico en cuestion
freqcritica=max(freqmaxvector);
ampcritica=max(ampmaxvector);
clear data
%Rellenamos los campos de edad, sexo y medicacion.
edadv(cont)=edad;
pesov(cont)=peso;
```

```
sexov(cont)=sexo;
medicacion1v(cont)=medicacion1;
medicacion2v(cont)=medicacion2;
medicacion3v(cont)=medicacion3;
freqmax=freqcritica;
freqmaxv(cont)=freqmax;
ampmax=ampcritica;
ampmaxv(cont)=ampmax;

%Armamos nuestros vectores de datos adecuadamente convertidos.
x=[x;[edadv',sexov',pesov',medicacion1v',medicacion2v',medicacion3v']];
y_d=[y_d;[freqmaxv',ampmaxv']];




save('Dat_train_sin.mat','x','y_d','-append');
                    end
                end
            end
        end
    end
end
```

Código C.6 Archivo **datsin** en MATLAB.

```
%Paso train_datsin en Matlab
load('Dat_train_sin.mat');

%Normalizamos las entradas
for i=1:6
    xn(:,i)=(x(:,i)-min(x(:,i)))/(max(x(:,i))-min(x(:,i)));
end


%Entrenamiento de la red neuronal
ttr=y_d'; %Hay que proporcionar los vectores como vectores fila y no como vectores columna
xtr=xn';


net=fitnet([30,20,10]);
net.divideParam.trainRatio=70/100;
net.divideParam.valRatio=30/100;
net.divideParam.testRatio=0/100;
net.trainParam.min_grad=1e-20;
net.trainParam.epochs=300;
net.layers{1}.transferFcn = 'logsig';
net.layers{2}.transferFcn = 'tansig';
[net,tr]=train(net,xtr,ttr);

%Comportamiento de la red
y_ent=net(xtr(:,tr.trainInd));
t_ent=ttr(:,tr.trainInd);
y_val=net(xtr(:,tr.valInd));
t_val=ttr(:,tr.valInd);
figure(1)
plot(t_ent(1,:),y_ent(1,:),'x')
hold on
plot(t_val(1,:),y_val(1,:),'o')
plot(0:12,0:12)
title('Frecuencia predicha frente a la real')
xlabel('Frecuencia real')
ylabel('Frecuencia predicha')
legend('Entrenamiento','Validacion')
hold off

figure(2)
plot(t_ent(2,:),y_ent(2,:),'x')
hold on
plot(t_val(2,:),y_val(2,:),'o')
plot(0:0.01:0.1,0:0.01:0.1)
title('Amplitud predicha frente a la real')
xlabel('Amplitud real')
ylabel('Amplitud predicha')
legend('Entrenamiento','Validacion')
hold off

%Guardado de la red para usos futuros
save('net.mat','net')
```

Código C.7 Archivo **train_datsin** en MATLAB.

```
%Paso pred en Matlab
load('net.mat')
%Introducimos un valor para llevar a cabo la prediccion.
prompt = 'Edad [años]:\n';
edad = input(prompt);

if (20>edad)||(edad>100)
    error('Debe introducir una edad entre los 20 y los 100 años')
end

prompt = 'Sexo:\n a: Mujer \n b: Hombre\n';
sexo = input(prompt,'s');
if sexo=='a'
    sexo=0;
elseif sexo=='b'
    sexo=1;
else
    sexo=2;
end

if (0>sexo)||(sexo>1)
    error('Debe introducir un codigo de la lista segun la lista anterior')
end
prompt = 'Peso [kg]:\n';

peso = input(prompt);
peso=round(peso);

if (40>peso)||(peso>160)
    error('Debe introducir un peso entre los 40 y los 160 kg')
end

prompt = 'Medicacion 1:\n a: La toma \n b: No la toma\n';
medicacion1 = input(prompt,'s');

if medicacion1=='a'
    medicacion1=1;
elseif medicacion1=='b'
    medicacion1=0;
else
    medicacion1=2;
end

if (0>medicacion1)||(medicacion1>1)
    error('Debe introducir un codigo de la lista segun la lista anterior')
end

prompt = 'Medicacion 2:\n a: La toma \n b: No la toma\n';
medicacion2 = input(prompt,'s');

if medicacion2=='a'
    medicacion2=1;
elseif medicacion2=='b'
    medicacion2=0;
else
    medicacion2=2;
end

if (0>medicacion2)||(medicacion2>1)
    error('Debe introducir un codigo de la lista segun la lista anterior')
end

prompt = 'Medicacion 3:\n a: La toma \n b: No la toma\n';
medicacion3 = input(prompt,'s');

if medicacion3=='a'
    medicacion3=1;
elseif medicacion3=='b'
    medicacion3=0;
else
    medicacion3=2;
```

```
end

if (0>medicacion3)||(medicacion3>1)
    error('Debe introducir un codigo de la lista segun la lista anterior')
end

edad=(edad-20)/(100-20);
peso=(peso-40)/(160-40);
xn=[edad,sexo,peso,medicacion1,medicacion2,medicacion3]';

salida=net(xn);
fprintf('El valor predicho de la frecuencia critica del paciente es de: %8.3f Hz\n',salida(1))
fprintf('El valor predicho de la amplitud critica del paciente es de: %8.3f m/s^2/Hz\n',salida(2)
    )
```

Código C.8 Archivo **pred** en MATLAB.

C.2 Códigos desarrollados en Python

```python
#paso1 en Python
import androidhelper
import time

cursor = androidhelper.Android()
dt = 10 #10ms entre mediciones, equivalente a una frecuencia de 100 Hz.
duracion = int(input('Introducir los segundos de duracion de la toma: '))*1000 #Duracion de la
    toma
tiempo=-40 #Inicialización del tiempo.
cursor.startSensingTimed(2,dt) #El 2 indica que vamos a leer los datos del acelerómetro
# y dt indica la frecuencia con la que lo vamos a hacer.
Acx=[]
Acy=[]
Acz=[]
tiempos=[]
while tiempo <= duracion:
    acelerometro=cursor.sensorsReadAccelerometer()
    tiempo+=dt
    tiempos.append(tiempo/1000)
    Acx.append(acelerometro[1][0])
    Acy.append(acelerometro[1][1])
    Acz.append(acelerometro[1][2])
    time.sleep(dt/1000)
cursor.stopSensing()

AcxT=open('Acx.txt', 'w')
AcyT=open('Acy.txt', 'w')
AczT=open('Acz.txt', 'w')
tiemposT=open('tiempos.txt', 'w')

for x in range(3,len(Acx)-3):
    AcxT.write(str(Acx[x])+',')
    AcyT.write(str(Acy[x])+',')
    AczT.write(str(Acz[x])+',')
    tiemposT.write(str(tiempos[x])+',')
AcxT.write(str(Acx[len(Acx)-2]))
AcyT.write(str(Acy[len(Acx)-2]))
AczT.write(str(Acz[len(Acx)-2]))
tiemposT.write(str(tiempos[len(Acx)-2]))

AcxT.close()
AcyT.close()
AczT.close()
tiemposT.close()
```

Código C.9 Archivo **paso1** en Python.

```
#paso2 en Python
import numpy as np
from scipy import signal
from scipy.signal import welch
from scipy.signal.windows import hann
import scipy.fftpack as fourier
import sqlite3
from datetime import datetime
import dill


prompt='Número del paciente \n'
num=input(prompt)

if bool(num)==False:
    print('Error, debe introducir un número')



prompt='Sexo:\n a: Mujer \n b: Hombre\n'
sexo=input(prompt)
if sexo=='a':
    sexo2=0
elif sexo=='b':
    sexo2=1
else:
    sexo2=2
if 0>sexo2 or sexo2>1:
    print('Error, debe introducir un código según la lista anterior')


prompt='Medicacion 1:\n a: La toma \n b: No la toma\n'
medicacion1=input(prompt)
if medicacion1=='a':
    medicacion12=1
elif medicacion1=='b':
    medicacion12=0
else:
    medicacion12=2
if 0>medicacion12 or medicacion12>1:
    print('Error, debe introducir un código según la lista anterior')


prompt='Medicacion 2:\n a: La toma \n b: No la toma\n'
medicacion2=input(prompt)
if medicacion2=='a':
    medicacion22=1
elif medicacion2=='b':
    medicacion22=0
else:
    medicacion22=2
if 0>medicacion22 or medicacion22>1:
    print('Error, debe introducir un código según la lista anterior')


prompt='Medicacion 3:\n a: La toma \n b: No la toma\n'
medicacion3=input(prompt)
if medicacion3=='a':
    medicacion32=1
elif medicacion3=='b':
    medicacion32=0
else:
    medicacion32=2
if 0>medicacion32 or medicacion32>1:
    print('Error, debe introducir un código según la lista anterior')


prompt='Peso [kg]:\n'
peso=input(prompt)
peso=int(peso)

```

```
72
73 prompt='Edad [años]:\n'
74 edad=input(prompt)
75 edad=int(edad)
76
77
78 prompt = 'Miembro:\n a: Brazo izquierdo\n b: Brazo derecho\n c: Pierna izquierda\n d: Pierna
       derecha\n e: Cabeza\n'
79 org=input(prompt)
80
81 if org=='a':
82     org2=1
83 elif org=='b':
84     org2=2
85 elif org=='c':
86     org2=3
87 elif org=='d':
88     org2=4
89 elif org=='e':
90     org2=5
91 else:
92     org2=0
93 if 1>org2 or org2>5:
94     print('Error, debe introducir un código según la lista anterior')
95
96 prompt = 'Postura:\n a: Postura a\n b: Postura b\n c: Postura c\n d: Postura d\n e: Postura e\n';
97 pos=input(prompt)
98
99 if pos=='a':
100     pos2=1
101 elif pos=='b':
102     pos2=2
103 elif pos=='c':
104     pos2=3
105 elif pos=='d':
106     pos2=4
107 elif pos=='e':
108     pos2=5
109 else:
110     pos2=0
111 if 1>pos2 or pos2>5:
112     print('Error, debe introducir un código según la lista anterior')
113
114
115
116
117 Acx=[]
118 Acy=[]
119 Acz=[]
120 tiempos=[]
121
122
123 AcxR=open('Acx.txt','r')
124 AcyR=open('Acy.txt','r')
125 AczR=open('Acz.txt','r')
126 tiemposR=open('tiempos.txt','r')
127
128
129 for acx in AcxR.readline().split(','):
130     if acx != ' ':
131       Acx.append(float(acx))
132 for acy in AcyR.readline().split(','):
133     if acy != ' ':
134       Acy.append(float(acy))
135 for acz in AczR.readline().split(','):
136     if acz != ' ':
137       Acz.append(float(acz))
138 for tiempo in tiemposR.readline().split(','):
139   if tiempo != ' ':
140     tiempos.append(float(tiempo))
141
```

```
142
143
144 L=len(tiempos)
145 dt=tiempos[1]-tiempos[0]
146 fs=1/dt #Frecuencia de muestreo
147
148
149 N=3 #Orden del filtro
150 rp=0.1 #El factor de rizado del filtro
151 wn=0.5 #Frecuencia crítica del filtro
152
153 b, a=signal.cheby1(N, rp, wn/(fs/2), btype='highpass', analog=False, output='ba')
154 Acx = signal.filtfilt(b, a, Acx)
155 Acy = signal.filtfilt(b, a, Acy)
156 Acz = signal.filtfilt(b, a, Acz)
157
158
159 Acxmax=max(abs(Acx))
160 indicexmax=np.argmax(abs(Acx))
161 tAcxmax=tiempos[indicexmax]
162 Acymax=max(abs(Acy))
163 indiceymax=np.argmax(abs(Acy))
164 tAcymax=tiempos[indiceymax]
165 Aczmax=max(abs(Acz))
166 indicezmax=np.argmax(abs(Acz))
167 tAczmax=tiempos[indicezmax]
168 print(f'El pico de aceleración en la dirección x es de {Acxmax} m/s^2 y se da en el instante {
      tAcxmax} s')
169 print(f'El pico de aceleración en la dirección y es de {Acymax} m/s^2 y se da en el instante {
      tAcymax} s')
170 print(f'El pico de aceleración en la dirección z es de {Aczmax} m/s^2 y se da en el instante {
      tAczmax} s')
171
172
173
174 acfft_x = fourier.fft(Acx)/fs
175 Acfft_x=abs(acfft_x)
176 Acfft_x=Acfft_x[0:L//2]
177 acfft_y = fourier.fft(Acy)/fs
178 Acfft_y=abs(acfft_y)
179 Acfft_y=Acfft_y[0:L//2]
180 acfft_z = fourier.fft(Acz)/fs
181 Acfft_z=abs(acfft_z)
182 Acfft_z=Acfft_z[0:L//2]
183 freq = (fs/L)*np.arange(0,L//2)
184
185
186 indicexfft_xmax=np.argmax(Acfft_x)
187 facxfftmax=freq[indicexfft_xmax]
188 ampxfftmax=Acfft_x[indicexfft_xmax]/fs
189 indiceyfft_ymax=np.argmax(Acfft_y)
190 facyfftmax=freq[indiceyfft_ymax]
191 ampyfftmax=Acfft_y[indiceyfft_ymax]/fs
192 indicezfft_zmax=np.argmax(Acfft_z)
193 faczfftmax=freq[indicezfft_zmax]
194 ampzfftmax=Acfft_z[indicezfft_zmax]/fs
195 facfftmaxv=[facxfftmax,facyfftmax,faczfftmax]
196 ampfftmaxv=[ampxfftmax,ampyfftmax,ampzfftmax]
197 ampfftmax=max(ampfftmaxv)
198 indiceampfft_max=np.argmax(ampfftmaxv)
199 facfftmax=facfftmaxv[indiceampfft_max]
200
201 print(f'El pico de aceleración en la dirección x se da a {facxfftmax} Hz')
202 print(f'El pico de aceleración en la dirección y se da a {facyfftmax} Hz')
203 print(f'El pico de aceleración en la dirección z se da a {faczfftmax} Hz')
204
205
206 nblock = 32
207 overlap = 0
208 win = hann(nblock, False)
209 f, Pxxacxf= welch(Acx,fs, window=win, noverlap=overlap, nfft=nblock, return_onesided=True)
```

```
f, Pxxacyf= welch(Acy,fs, window=win, noverlap=overlap, nfft=nblock, return_onesided=True)
f, Pxxaczf= welch(Acz,fs, window=win, noverlap=overlap, nfft=nblock, return_onesided=True)


now=datetime.now()
filename=now.strftime("%Y_%m_%d-%H_%M_%S")
filename=filename+'_Pa_'+str(num)+'_S_'+sexo+'_Pe_'+str(peso)+'_M_'+org+'_Po_'+pos
filename=f'{filename}.pkl'
dill.dump_session(filename)

#Hacemos la conexion a la base de datos
conexion=sqlite3.connect('pacientes_entrenamiento_red.db')

cursor=conexion.cursor() #Creamos un cursor

#Crear una tabla
cursor.execute("""CREATE TABLE IF NOT EXISTS pacientes(
    num INTEGER PRIMARY KEY,
    edad INTEGER,
    sexo INTEGER,
    peso INTEGER,
    medicacion1 INTEGER,
    medicacion2 INTEGER,
    medicacion3 INTEGER,
    freqmax REAL,
    ampmax REAL
)""")

conexion.commit() #Guardamos los cambios

#Insertar datos en la tabla creada
cursor.execute("INSERT INTO pacientes(num,edad,sexo,peso,medicacion1,medicacion2,medicacion3,
    freqmax,ampmax) VALUES (?,?, ?, ?, ?,?,?,?,?)",(num,edad,sexo2,peso,medicacion12,
    medicacion22,medicacion32,facfftmax,ampfftmax))

conexion.commit() #Guardamos los cambios

conexion.close() #Cerramos la conexion


AcxR.close()

AcyR.close()

AczR.close()

tiemposR.close()
```

Código C.10 Archivo **paso2** en Python.

```python
#paso3 en Python
import dill
import os
import sqlite3

dir = '/storage/emulated/0/qpython'
with os.scandir(dir) as ficheros:
    ficheros = [fichero.name for fichero in ficheros if fichero.is_file()]
for iname in range(0,len(ficheros)):
    print(f'{iname}: {ficheros[iname]}')


prompt = 'Seleccione el número del fichero de resultados que quiere analizar:\n'
num = input(prompt)

#Almacenamiento de número introducido en una base de datos provisional

#Hacemos la conexion a la base de datos
conexion=sqlite3.connect('seleccion.db')

cursor=conexion.cursor() #Creamos un cursor

#Crear una tabla
cursor.execute("""CREATE TABLE IF NOT EXISTS seleccion(
    num INTEGER
);""")

conexion.commit() #Guardamos los cambios

#Borramos todo el contenido previo que pudiera haber puesto que no nos interesa
cursor.execute('DELETE FROM seleccion')
conexion.commit() #Guardamos los cambios

#Insertar datos en la tabla creada
cursor.execute("INSERT INTO seleccion(num) VALUES (?)",(num,))
conexion.commit() #Guardamos los cambios

conexion.close() #Cerramos la conexion
```

Código C.11 Archivo **paso3** en Python.

```python
#paso4 en Python
import matplotlib.pyplot as plt
import dill
import os
import sqlite3

dir = '/storage/emulated/0/qpython'
with os.scandir(dir) as ficheros:
    ficheros = [fichero.name for fichero in ficheros if fichero.is_file()]

conexion=sqlite3.connect('seleccion.db') #Hacemos la conexion a la misma

cursor=conexion.cursor() #Creamos un cursor

cursor.execute("SELECT * FROM seleccion")
opc_seleccion=cursor.fetchall()
seleccion=opc_seleccion[0][0]




conexion.close() #Cerramos la conexion


filename=f'{ficheros[seleccion]}'
dill.load_session(filename)
print(f'El pico de aceleración en la dirección x es de {Acxmax} m/s^2 y se da en el instante {tAcxmax} s')
print(f'El pico de aceleración en la dirección y es de {Acymax} m/s^2 y se da en el instante {tAcymax} s')
print(f'El pico de aceleración en la dirección z es de {Aczmax} m/s^2 y se da en el instante {tAczmax} s')
print(f'El pico de aceleración en la dirección x se da a {facxfftmax} Hz')
print(f'El pico de aceleración en la dirección y se da a {facyfftmax} Hz')
print(f'El pico de aceleración en la dirección z se da a {faczfftmax} Hz')

fig,ax=plt.subplots(3,1)
ax[0].set_title('Aceleración en el dominio del tiempo')
ax[0].plot(tiempos,Acx, label='Acx')
ax[0].plot(tiempos,Acy, label='Acy')
ax[0].plot(tiempos,Acz, label='Acz')
ax[1].set_title('Aceleración en el dominio de la frecuencia')
ax[1].plot(freq,Acfft_x/fs, label='Acfft_x')
ax[1].plot(freq,Acfft_y/fs, label='Acfft_y')
ax[1].plot(freq,Acfft_z/fs, label='Acfft_z')
ax[2].set_title('Densidad espectral de la aceleración')
ax[2].plot(f, Pxxacxf, label='PSD_x')
ax[2].plot(f, Pxxacyf, label='PSD_y')
ax[2].plot(f, Pxxaczf, label='PSD_z')
ax[0].legend()
ax[1].legend()
ax[2].legend()
plt.subplots_adjust(left=0.125,bottom=0.1, right=0.9, top=0.9, wspace=0.2, hspace=1)
plt.show()
```

Código C.12 Archivo **paso4** en Python.

```python
#Paso datsin en Python
import numpy as np
import matplotlib.pyplot as plt
from scipy import signal
from scipy.signal import welch
from scipy.signal.windows import hann
import scipy.fftpack as fourier
import sqlite3
import random
from math import pi

#Creamos la conexion a la base de datos.
conexion=sqlite3.connect('pacientes_sin_entrenamiento_red.db') #Hacemos la conexion a la misma
cursor=conexion.cursor() #Creamos un cursor

#Crear una tabla
cursor.execute("""CREATE TABLE IF NOT EXISTS pacientes_sin(
    num INTEGER PRIMARY KEY AUTOINCREMENT,
    edad INTEGER,
    sexo INTEGER,
    peso INTEGER,
    medicacion1 INTEGER,
    medicacion2 INTEGER,
    medicacion3 INTEGER,
    freqmax REAL,
    ampmax REAL
)""")

conexion.commit() #Guardamos los cambios

for medicacion1 in range(0,2): #Medicacion1 puede tomar 0 o 1
    for medicacion2 in range(0,2): #Medicacion2 puede tomar 0 o 1
        for medicacion3 in range(0,2): #Medicacion3 puede tomar 0 o 1
            for sexo in range(0,2): #Sexo
                for edadmin in range(20,100,10): #Edadmin va de 20 a 90
                    for pesomin in range(40,160,10): #Pesomin va de 40 a 150 kg
                        Acx=[]
                        Acy=[]
                        Acz=[]
                        tiempos=[]


                        AcxR=open('Acx.txt','r')
                        AcyR=open('Acy.txt','r')
                        AczR=open('Acz.txt','r')
                        tiemposR=open('tiempos.txt','r')


                        for acx in AcxR.readline().split(','):
                                if acx != ' ':
                                    Acx.append(float(acx))
                        for acy in AcyR.readline().split(','):
                                    if acy != ' ':
                                        Acy.append(float(acy))
                        for acz in AczR.readline().split(','):
                                if acz != ' ':
                                    Acz.append(float(acz))
                        for tiempo in tiemposR.readline().split(','):
                            if tiempo != ' ':
                                tiempos.append(float(tiempo))

                        L=len(tiempos)
                        dt=tiempos[1]-tiempos[0]
                        fs=1/dt #Frecuencia de muestreo


                        N=3 #Orden del filtro
                        rp=0.1 #El factor de rizado del filtro
                        wn=0.5 #Frecuencia crítica del filtro

```

```
                        b, a=signal.cheby1(N, rp, wn/(fs/2), btype='highpass', analog=False,
    output='ba')
                        Acx = signal.filtfilt(b, a, Acx)
                        Acy = signal.filtfilt(b, a, Acy)
                        Acz = signal.filtfilt(b, a, Acz)


                        Acxmax=max(Acx)
                        indicexmax=np.argmax(Acx)
                        tAcxmax=tiempos[indicexmax]
                        Acymax=max(Acy)
                        indiceymax=np.argmax(Acy)
                        tAcymax=tiempos[indiceymax]
                        Aczmax=max(Acz)
                        indicezmax=np.argmax(Acz)
                        tAczmax=tiempos[indicezmax]


                        edadmax=edadmin+10
                        pesomax=pesomin+10
                        edad=random.randrange(edadmin,edadmax,1)
                        peso=random.randrange(pesomin,pesomax,1)
                        sexo=sexo
                        medicacion1=medicacion1
                        medicacion2=medicacion2
                        medicacion3=medicacion3
                        ke=(edad-20)/(100-20) #Cte de ponderación de la edad va de 0 a 1
                        kp=(peso-40)/(160-40) #Cte de ponderación del peso va de 0 a 1
                        kpe=1+0.1*ke+0.1*kp
                        k1=medicacion1 #Cte de pico de la medicación 1
                        k2=medicacion2 #Cte de pico de la medicación 2
                        k3=medicacion3 #Cte de pico de la medicación 3


                        #Añadimos ruido para generar datos sintéticos.
                        args1=np.dot(2*pi*5,tiempos)
                        args2=np.dot(2*pi*7,tiempos)
                        args3=np.dot(2*pi*9,tiempos)
                        Acx=Acx+0.05*Acxmax*(-np.ones(L)+2*np.random.random(L))+0.1*k1*Acxmax*np.
    sin(args1)+0.1*k2*Acxmax*np.sin(args2)+0.1*k3*Acxmax*np.sin(args3)
                        Acy=Acy+0.05*Acymax*(-np.ones(L)+2*np.random.random(L))+0.1*k1*Acymax*np.
    sin(args1)+0.1*k2*Acymax*np.sin(args2)+0.1*k3*Acymax*np.sin(args3)
                        Acz=Acz+0.05*Aczmax*(-np.ones(L)+2*np.random.random(L))+0.1*k1*Aczmax*np.
    sin(args1)+0.1*k2*Aczmax*np.sin(args2)+0.1*k3*Aczmax*np.sin(args3)
                        Acx=kpe*Acx
                        Acy=kpe*Acy
                        Acz=kpe*Acz



                        #Pasamos los datos sintéticos al dominio de la frecuencia.
                        acfft_x = fourier.fft(Acx)/fs
                        Acfft_x=abs(acfft_x)
                        Acfft_x=Acfft_x[0:L//2]
                        acfft_y = fourier.fft(Acy)/fs
                        Acfft_y=abs(acfft_y)
                        Acfft_y=Acfft_y[0:L//2]
                        acfft_z = fourier.fft(Acz)/fs
                        Acfft_z=abs(acfft_z)
                        Acfft_z=Acfft_z[0:L//2]
                        freq = (fs/L)*np.arange(0,L//2)


                        indicexfft_xmax=np.argmax(Acfft_x)
                        facxfftmax=freq[indicexfft_xmax]
                        ampxfftmax=Acfft_x[indicexfft_xmax]/fs
                        indiceyfft_ymax=np.argmax(Acfft_y)
                        facyfftmax=freq[indiceyfft_ymax]
                        ampyfftmax=Acfft_y[indiceyfft_ymax]/fs
                        indicezfft_zmax=np.argmax(Acfft_z)
                        faczfftmax=freq[indicezfft_zmax]
```

```
                        ampzfftmax=Acfft_z[indicezfft_zmax]/fs
                        facfftmaxv=[facxfftmax,facyfftmax,faczfftmax]
                        ampfftmaxv=[ampxfftmax,ampyfftmax,ampzfftmax]
                        ampfftmax=max(ampfftmaxv)
                        indiceampfft_max=np.argmax(ampfftmaxv)
                        facfftmax=facfftmaxv[indiceampfft_max]

                        #Insertar datos en la tabla creada
                        cursor.execute("INSERT INTO pacientes_sin(num,edad,sexo,peso,medicacion1,
    medicacion2,medicacion3,freqmax,ampmax) VALUES (null,?, ?, ?, ?,?,?,?,?)",(edad,sexo,peso,
    medicacion1,medicacion2,medicacion3,facfftmax,ampfftmax))

                        conexion.commit() #Guardamos los cambios

conexion.close() #Cerramos la conexion


AcxR.close()

AcyR.close()

AczR.close()

tiemposR.close()
```

Código C.13 Archivo **datsin** en Python.

```
import pandas as pd
import numpy as np
from numpy import random
import matplotlib.pyplot as plt
from sklearn.neural_network import MLPRegressor
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import MinMaxScaler
from sklearn import metrics
from sklearn.model_selection import GridSearchCV
import sqlite3
import pickle


conexion=sqlite3.connect('pacientes_sin_entrenamiento_red.db')

cursor=conexion.cursor()

cursor.execute("SELECT * FROM pacientes_sin")
pacientes_sin=cursor.fetchall()

edadv=[]
sexov=[]
pesov=[]
medicacion1v=[]
medicacion2v=[]
medicacion3v=[]
freqmaxv=[]
ampmaxv=[]

for paciente in pacientes_sin:
    edadv.append(paciente[1])
    sexov.append(paciente[2])
    pesov.append(paciente[3])
    medicacion1v.append(paciente[4])
    medicacion2v.append(paciente[5])
    medicacion3v.append(paciente[6])
    freqmaxv.append(paciente[7])
    ampmaxv.append(paciente[8])


conexion.close()

df=pd.DataFrame({
    'xo': edadv,
    'x1': sexov,
    'x3': pesov,
    'x4': medicacion1v,
    'x5': medicacion2v,
    'x6': medicacion3v,
    'y0': freqmaxv,
    'y1': ampmaxv
})
x=df.iloc[:,0:6].values
y1=df.iloc[:,6:7].values
y2=df.iloc[:,7:8].values

trainX, testX, trainY1, testY1 = train_test_split(x,y1,test_size=0.2,random_state=1)

model1=MLPRegressor(hidden_layer_sizes=(30,20,10),max_iter=600,activation='relu',solver='adam')

model1.fit(trainX,trainY1.ravel())

#Guardar la red entrenada de la salida en frecuencia
filename = 'red_entrenada1.sav'
pickle.dump(model1, open(filename, 'wb'))

#Cargamos la red entrenada
loaded_model1 = pickle.load(open(filename, 'rb'))

y_pred_train1=loaded_model1.predict(trainX)

```

```
y_pred_test1=loaded_model1.predict(testX)


salida1train_pred=y_pred_train1
salida1test_pred=y_pred_test1

t1_train=trainY1
t1_test=testY1

trainX, testX, trainY2, testY2 = train_test_split(x,y2,test_size=0.2,random_state=1)


model2=MLPRegressor(hidden_layer_sizes=(30,30,20,20),max_iter=600,activation='logistic',solver='
    adam')

model2.fit(trainX,trainY2.ravel())

#Guardar la red entrenada de la salida en amplitud
filename = 'red_entrenada2.sav'
pickle.dump(model2, open(filename, 'wb'))

#Cargamos la red entrenada
loaded_model2 = pickle.load(open(filename, 'rb'))

y_pred_train2=loaded_model2.predict(trainX)

y_pred_test2=loaded_model2.predict(testX)

salida2train_pred=y_pred_train2
salida2test_pred=y_pred_test2

t2_train=trainY2
t2_test=testY2




print('Mean Squared Error s1:', metrics.mean_squared_error(t1_test, salida1test_pred))

print('Mean Squared Error s2:', metrics.mean_squared_error(t2_test, salida2test_pred))

fig,ax=plt.subplots()
ax1= plt.subplot(411)
plt.plot(loaded_model1.loss_curve_)
plt.title("MSE de la frecuencia vs Número de iteraciones", fontsize=14)
plt.xlabel('Iteraciones')
plt.ylabel('MSE de la frecuencia')

ax2= plt.subplot(412)
plt.scatter(t1_train,salida1train_pred,marker='o')
plt.scatter(t1_test,salida1test_pred,marker='v')
plt.plot(np.arange(0,13),np.arange(0,13))
plt.title('Salida en frecuencia predicha frente a la real')
plt.xlabel('Salida en frecuencia real')
plt.ylabel('Salida en frecuencia predicha')

ax3= plt.subplot(413)
plt.plot(loaded_model2.loss_curve_)
plt.title("MSE de la amplitud vs Número de iteraciones", fontsize=14)
plt.xlabel('Iteraciones')
plt.ylabel('MSE de la amplitud')

ax4= plt.subplot(414)
plt.scatter(t2_train,salida2train_pred,marker='o')
plt.scatter(t2_test,salida2test_pred,marker='v')
plt.plot(np.arange(0,0.01,0.001),np.arange(0,0.01,0.001))
plt.title('Salida en amplitud predicha frente a la real')
plt.xlabel('Salida en amplitud real')
plt.ylabel('Salida en amplitud predicha')

plt.subplots_adjust(left=0.4,bottom=0.1, right=0.6, top=0.9, wspace=0.2, hspace=0.5)
```

```
142 plt.show()
```

Código C.14 Archivo **train_datsin** en Python.

```python
#Paso salida_pred en Python
import pandas as pd
import pickle

prompt='Edad [años]:\n'
edadx=input(prompt)
edadx=int(edadx)
if 20>edadx or edadx>100:
    print('Error, debe introducir una edad entre los 20 y los 100 años')
prompt='Sexo:\n a: Mujer \n b: Hombre\n'
sexox=input(prompt)
if sexox=='a':
    sexox=0
elif sexox=='b':
    sexox=1
else:
    sexox=2
if 0>sexox or sexox>1:
    print('Error, debe introducir un código según la lista anterior')

prompt='Peso [kg]:\n'
pesox=input(prompt)
pesox=int(pesox)
if 40>pesox or pesox>160:
    print('Error, debe introducir un peso entre los 40 y los 160 kg')


prompt='Medicacion 1:\n a: La toma \n b: No la toma\n'
medicacion1x=input(prompt)
if medicacion1x=='a':
    medicacion1x=1
elif medicacion1x=='b':
    medicacion1x=0
else:
    medicacion1x=2
if 0>medicacion1x or medicacion1x>1:
    print('Error, debe introducir un código según la lista anterior')


prompt='Medicacion 2:\n a: La toma \n b: No la toma\n'
medicacion2x=input(prompt)
if medicacion2x=='a':
    medicacion2x=1
elif medicacion2x=='b':
    medicacion2x=0
else:
    medicacion2x=2
if 0>medicacion2x or medicacion2x>1:
    print('Error, debe introducir un código según la lista anterior')


prompt='Medicacion 3:\n a: La toma \n b: No la toma\n'
medicacion3x=input(prompt)
if medicacion3x=='a':
    medicacion3x=1
elif medicacion3x=='b':
    medicacion3x=0
else:
    medicacion3x=2
if 0>medicacion3x or medicacion3x>1:
    print('Error, debe introducir un código según la lista anterior')
entrada=pd.DataFrame({
    'xo': edadx,
    'x1': sexox,
    'x3': pesox,
    'x4': medicacion1x,
    'x5': medicacion2x,
    'x6': medicacion3x,
},index=[0])


```

```
#Cargar las redes entrenadas
filename = 'red_entrenada1.sav'
loaded_model1 = pickle.load(open(filename, 'rb'))

filename = 'red_entrenada2.sav'
loaded_model2 = pickle.load(open(filename, 'rb'))

salida_pred1=loaded_model1.predict(entrada.values)

salida_pred2=loaded_model2.predict(entrada.values)


print("Frecuencia crítica:", salida_pred1[0], "Hz")
print("Amplitud crítica:", salida_pred2[0], "m/s^2/Hz")
print("\n")
```

Código C.15 Archivo **salida_pred** en Python.

Índice de Figuras

Índice de Códigos

Bibliografía

[1] K. P. Bhatia, P. Bain, N. Bajaj, R. J. Elble, M. Hallett, E. D. Louis, J. Raethjen, M. Stamelou, C. M. Testa, and G. Deuschl, "Consensus statement on the classification of tremors. from the task force on tremor of the international parkinson and movement disorder society," *Movement Disorders*, vol. 33, pp. 75–87, 1 2018.

[2] E. D. Louis, "Diagnosis and management of tremor," *CONTINUUM Lifelong Learning in Neurology*, vol. 22, pp. 1143–1158, 8 2016. [Online]. Available: https://journals.lww.com/continuum/Fulltext/2016/08000/Diagnosis_and_Management_of_Tremor.11.aspx

[3] I. García-Magariño, C. Medrano, I. Plaza, and B. Oliván-Blázquez, "A smartphone-based system for detecting hand tremors in unconstrained environments," *Personal and Ubiquitous Computing*, vol. 20, 11 2016.

[4] S. Barrantes, A. J. S. Egea, H. A. G. lez Rojas, M. J. Martí, Y. Compta, F. Valldeoriola, E. S. Mezquita, E. Tolosa, and J. Valls-Solè, "Differential diagnosis between parkinson's disease and essential tremor using the smartphone's accelerometer," 2017. [Online]. Available: https://doi.org/10.1371/journal.pone.0183843

[5] A. Brindha, K. Sunitha, and S. R. Wilson, "TREMOR CLASSIFICATION USING WEARABLE IOT BASED SENSORS," *IOP Conference Series: Materials Science and Engineering*, vol. 1219, no. 1, p. 012024, jan 2022. [Online]. Available: https://doi.org/10.1088/1757-899x/1219/1/012024

[6] "Matlab mobile - apps en google play." [Online]. Available: https://play.google.com/store/apps/details?id=com.mathworks.matlabmobile&hl=es_419&gl=US

[7] "Qpython 3l - python for android - apps en google play." [Online]. Available: https://play.google.com/store/apps/details?id=org.qpython.qpy3&hl=es_419&gl=US

[8] "Pydroid 3 - ide for python 3 - aplicaciones en google play." [Online]. Available: https://play.google.com/store/apps/details?id=ru.iiec.pydroid3&hl=es&gl=US

[9] "El modelo de redes neuronales - documentación de ibm." [Online]. Available: https://www.ibm.com/docs/es/spss-modeler/SaaS?topic=networks-neural-model

[10] "La-mejor-imagen-de-redes-neuronales-artificiales.png (1883×1005)." [Online]. Available: https://deingenierias.com/wp-content/uploads/La-Mejor-imagen-de-Redes-Neuronales-Artificiales.png

[11] "Machine learning: Cómo desarrollar un modelo desde cero | by victor roman | ciencia y datos | medium." [Online]. Available: https://medium.com/datos-y-ciencia/machine-learning-c%C3%B3mo-desarrollar-un-modelo-desde-cero-cc17654f0d48

[12] "Tipos de capas | interactive chaos." [Online]. Available: https://interactivechaos.com/es/manual/tutorial-de-deep-learning/tipos-de-capas

[13] "sigmoid.png (400×300)." [Online]. Available: https://ml4a.github.io/images/figures/sigmoid.png

[14] “Tangente hiperbólica - wikipedia, la enciclopedia libre.” [Online]. Available: https://es.wikipedia.org/wiki/Tangente_hiperb%C3%B3lica

[15] “relu.png (347×280).” [Online]. Available: https://ml4a.github.io/images/figures/relu.png

[16] “Función identidad.” [Online]. Available: https://www.universoformulas.com/matematicas/analisis/funcion-identidad/

[17] “Function fitting neural network - matlab fitnet - mathworks españa.” [Online]. Available: https://es.mathworks.com/help/deeplearning/ref/fitnet.html;jsessionid=8b0aefc51835ba973aee0ad4be9f

[18] “Train shallow neural network - matlab train - mathworks españa.” [Online]. Available: https://es.mathworks.com/help/deeplearning/ref/network.train.html

[19] “Sqlite documentation.” [Online]. Available: https://www.sqlite.org/docs.html

[20] “Numpy user guide — numpy v1.22 manual.” [Online]. Available: https://numpy.org/doc/stable/user/index.html#user

[21] “Users guide — matplotlib 3.5.2 documentation.” [Online]. Available: https://matplotlib.org/stable/users/index

[22] “Scipy user guide — scipy v1.9.0.dev0+2034.211380d manual.” [Online]. Available: https://scipy.github.io/devdocs/tutorial/index.html#user-guide

[23] “dill module documentation — dill 0.3.5.dev0 documentation.” [Online]. Available: https://dill.readthedocs.io/en/latest/dill.html

[24] “User guide — pandas 1.4.2 documentation.” [Online]. Available: https://pandas.pydata.org/pandas-docs/stable/user_guide/index.html#user-guide

[25] “User guide: contents — scikit-learn 1.1.0 documentation.” [Online]. Available: https://scikit-learn.org/stable/user_guide.html

[26] “pickle — python object serialization — python 3.10.4 documentation.” [Online]. Available: https://docs.python.org/3/library/pickle.html

[27] “Sign in to matlab drive - matlab simulink - mathworks.” [Online]. Available: https://drive.matlab.com/login