Trabajo de Fin de Grado
Grado en Ingeniería Electrónica, Robótica y Mecatrónica

Movimiento de un brazo robótico con ondas cerebrales

Autora: Laura Escobar Moralejo
Tutor: Alfredo Pérez Vega-Leal

Dpto. de Ingeniería Electrónica
Escuela Técnica Superior de Ingeniería
Universidad de Sevilla

Sevilla, 2019

Trabajo de Fin de Grado
Grado en Ingeniería Electrónica, Robótica y Mecatrónica

Movimiento de un brazo robótico con ondas cerebrales

Autora:
Laura Escobar Moralejo

Tutor:
Alfredo Pérez Vega-Leal
Profesor Contratado Doctor

Dpto. de Ingeniería Electrónica
Escuela Técnica Superior de Ingeniería
Universidad de Sevilla
Sevilla, 2019

Trabajo de Fin de Grado: Movimiento de un brazo robótico con ondas cerebrales

Autora: Laura Escobar Moralejo

Tutor: Alfredo Pérez Vega-Leal

El tribunal nombrado para juzgar el Proyecto arriba indicado, compuesto por los siguientes miembros:

Presidente:

Vocales:

Secretario:

Acuerdan otorgarle la calificación de:

Sevilla, 2019

El Secretario del Tribunal

A mi familia

A mis amigos

Agradecimientos

Quería aprovechar estas pocas líneas para agradecer a mi familia por todo el apoyo y la gran paciencia en estos duros y estresantes años de carrera.

Quería mencionar también a todos mis amigos, a los de fuera de la ETSI y a los de aquí. Los de fuera, que me han ayudado a evadirme, a relajarme, a divertirme y a sacar mi cabeza del ordenador y de los libros, que me han aguantado en mis días de mayor estrés y que me han ayudado a conseguir lo que me propusiera. Los de dentro, que me han ayudado en tantos trabajos, exámenes y prácticas, que han hecho de mis largos días en la facultad unos días geniales; aquellos que se han convertido en mi familia de la ETSI.

Agradecer también a todos los profesores que he encontrado en mi camino hasta llegar hasta aquí y en especial, a mi tutor, Alfredo, por darme ánimos y ayuda cuando lo he necesitado.

A todos, gracias.

.

Laura Escobar Moralejo

Escuela Técnica Superior de Ingeniería

Sevilla, 2019

Resumen

En este documento se muestra el experimento realizado que utiliza ondas cerebrales para mover un brazo robótico.

En primer lugar, se muestra una breve introducción del contexto y la motivación de este trabajo, acabando con una conclusión del alcance real al que podría llegar este proyecto.

A continuación, se parte del estudio de la onda cerebral y de la red neuronal creada por la compañera Nagore Peirotén en su TFG para, seguidamente, mostrar los cambios realizados en esta RNA para mejorar su precisión.

Finalmente, se muestra la conexión entre la red neuronal y el brazo robótico, demostrando así el funcionamiento de este proyecto.

Abstract

In this document we can find the experiment produced which uses brain waves to move a robotic arm.

Firstly, it shows a brief introduction of the context and the motivation of this project, ending with a conclusion of the real significance which this project could reach.

Next, we are based in the colleague Nagore Peirotén López de Arbina's investigation of the brain wave and the artificial neuronal network (ANN) she created to show the changes made to improve this ANN's precision.

Finally, it shows the conexions between the ANN and the robotic arm to demonstrate its operation.

Índice

Índice de Tablas

Índice de Figuras

Notación

RNA (ANN)	Red Neuronal Artificial (Atificial Neuronal Network)
EEG	Electroencefalograma
BCI	Interfaz Cerebro-Computadora (Brain-Computer Interface)
SNC	Sistema Nervioso Central
EcOG	Electrocorticografía
MEG	Magnetoencefalografía
fMRI	Imágenes de resonancia magnético funcional
PET	Termografía por emisión de positrones
SCP	Potenciales Corticales Lentos (Slow Cortical Potentials)
CNV	Variación contingente negativa
IA	Inteligencia Artificial

1 INTRODUCCIÓN

El complejo mundo de las prótesis tuvo un gran avance durante todo el siglo XX. Aunque datan prótesis desde la época egipcia, con las numerosas guerras producidas el siglo pasado fue cuando la humanidad se vio en la necesidad de estudiar y avanzar en este ámbito. Durante muchos años y aún en la actualidad, las prótesis ortopédicas funcionan como un órgano «pasivo» que permite a la persona amputada retomar gran parte de su movilidad e independencia, pero que no tiene movilidad propia.

Sin embargo, desde los años ochenta y sobre todo, desde el 2000 en adelante, con el avance de la robótica y la inteligencia artificial, se han empezado a investigar las neuroprótesis. Es decir, el manejo de prótesis ortopédicas con señales neuronales, a través de un chip insertado en el cerebro del paciente.

1.1 Estado del arte

Actualmente se investigan neuroprótesis que apoyan y asisten a personas con movilidad reducida o nula, reconectando sus nervios con su cerebro a pesar de los daños que puedan tener en la médula espinal. Como ejemplos tenemos la investigación realizada por un equipo de la Universidad Case Western Reserve (Ohio, EEUU) [1], que ha conseguido que una persona tetrapléjica beba y coma por sí sola; o el ensayo realizado por los investigadores de Caltech (California, EEUU) [2], donde un paciente tetrapléjico ha movido un brazo robótico externo a su cuerpo y lo ha utilizado como si fuera propio, simplemente pensando en la acción que quería llevar a cabo. Encontramos otros ejemplos testados en primates como el estudio del Centro Médico de la Duke University [3], en el que han controlado brazos mecánicos sin recurrir a movimientos, solo con señales cerebrales. Estos estudios nos demuestran que es posible realizar movimientos externos al cuerpo solo con la intención de hacerlos.

En España, más de 90 000 personas han sufrido alguna amputación [4] y, sin embargo, la Seguridad Social española solo cubre una prótesis básica y pasiva, con tan solo una función estética. Solo 400 personas en el mundo tienen prótesis con acoplamiento mecánico y oseointegración, es decir, con conexión mecánica a los huesos, músculos y nervios. De estas 400 personas, gran parte se encuentran en países con mayor inversión en medicina y sanidad como Suecia o Francia. [5] Sin embargo, las neuroprótesis aún son proyectos de investigación y pocas personas en el mundo han podido probar alguna.

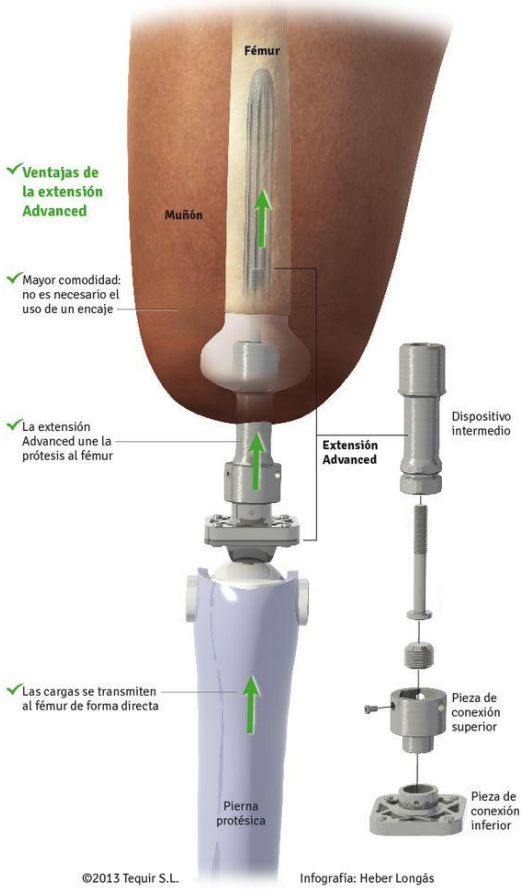

Ilustración 1: Prótesis de Oseointegración

1.2 Motivación

Este trabajo comienza con un proyecto realizado con el departamento de Ingeniería Electrónica en la ETSI, *Neurotronics*, que se basa en la construcción de un EEG desde cero. A partir de aquí surgió la idea de mover un brazo robótico con las ondas cerebrales obtenidas a partir de un electroencefalograma, dando esto lugar a, por el momento, dos trabajos de fin de grado. En primer lugar, el TFG que se comenta en repetidas ocasiones en esta memoria *Diseño de una red neuronal en Matlab para análisis de señales de electroencefalograma* de Nagore Peirotén López de Arbina, utilizado como base para este proyecto pues realiza un amplio estudio de las características de las ondas cerebrales necesarias para detectar movimientos corporales; y en segundo lugar este TFG.

Este proyecto está motivado por la que considero mi vocación, que es ayudar a personas con necesidad de una prótesis y conseguir que tengan una vida con las mínimas dificultades y lo más normalizada posible.

Por lo tanto, este trabajo se traduce en un prototipo que pueda convertirse, en el día de mañana, en un dispositivo inalámbrico que las personas discapacitadas puedan utilizar para mover miembros robóticos como si fueran sus propias articulaciones.

1.3 Objetivo

Crear una interfaz cerebro-computadora (BCI por sus siglas en inglés) con el fin de mover un brazo robótico mediante el mero hecho de pensar en moverlo.

En este trabajo de fin de grado se hará un primer prototipado de una idea mayor: una prótesis robótica, adherida al cuerpo del paciente, que pueda moverse a través de sus ondas cerebrales, captadas y analizadas por un chip insertado en su cerebro. Esto sería posible en pacientes que han sufrido amputaciones y no en pacientes nacidos sin algún miembro, pues las ondas cerebrales que se utilizarían para su movimiento son las que se producen en el cerebro al pensar en mover el miembro; si nunca se ha tenido este miembro, podemos suponer que estas ondas no serán iguales.

1.4 Prototipo

Se utilizará un brazo robótico básico (externo a cualquier ser vivo) que se controlará a partir de ondas cerebrales captadas por un electroencefalograma ordinario y analizadas por una red neuronal artificial. En este trabajo se diseñará una red neuronal artificial con el objetivo de diferenciar las ondas cerebrales que se producen al mover el brazo izquierdo de las producidas al realizar otros movimientos como los del brazo derecho, la lengua o los pies. A continuación, se moverá el brazo robótico hacia una posición si se detectan movimientos en el brazo izquierdo o se mantendrá en reposo en cualquier otro caso.

1.4.1 Posibles mejoras y viabilidad

El método que se propone se trata de un BCI no invasivo pues, para el paciente, solo requiere de unos electrodos situados en el cuero cabelludo. Sin embargo, este prototipo no tendría sentido llevarlo a la vida real tal y como es ahora mismo pues puede resultar poco viable hacer una vida normal de esta forma y esto es, al fin y al cabo, lo que busca este proyecto: facilitar y mejorar la calidad de vida de personas con dificultades físicas.

Actualmente existen también métodos invasivos en los que se implantan electrodos directamente en la corteza cerebral (para medir actividad de neuronas concretas) o en el córtex (para grupos de neuronas) mediante una intervención quirúrgica. No obstante, por los riesgos que conlleva esta operación suele realizarse solo en animales con fines de estudio.

Gracias a los avances en medicina y tecnología, en tan solo unos años esta operación podría reducir sus riesgos y aumentar su viabilidad, convirtiendo este proyecto en una realidad.

1.5 Hardware y Software

1.5.1 Montaje hardware

Tras la salida de la red neuronal artificial se ha construido un montaje hardware consistente en un brazo robótico de aluminio de seis grados de libertad (hombro, antebrazo, codo, brazo, muñeca y pinza) movido por servomotores MG996R.

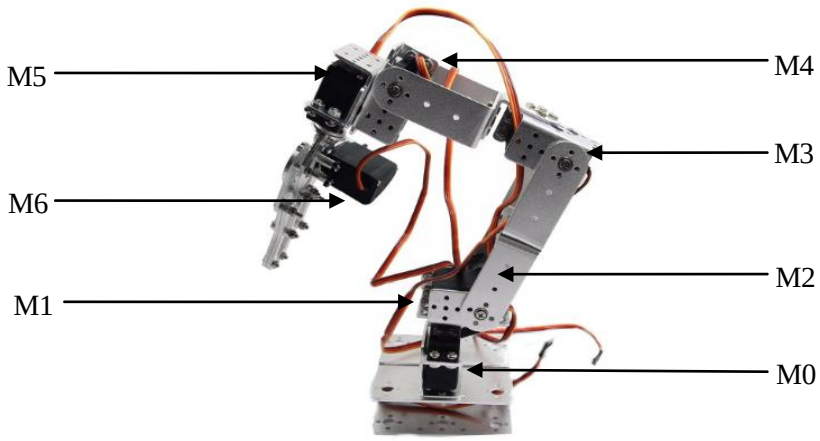

Ilustración 2: Imagen del brazo robótico construido

Tabla 1: Especificaciones servomotor MG996R

Voltaje de operación	4.8 V - 7.2 V
Velocidad de operación	0.17s/60º (4.8V), 0.14s/60º (6V)
Torque en reposo	9.4 kg·cm (4.8V), 12 kg·cm (6V)
Ángulo de giro	120º-180º máx.

Debido a la limitación de que la placa de *Arduino* no puede alimentar más de dos servomotores de esta potencia, y ya que el objetivo de este trabajo es mover/no mover el brazo y no requiere de alcanzar una posición determinada, se ha optado por simplemente mover los servos M1 y M2. Estos se conectarán para que trabajen en paralelo en la placa de *Arduino*, que será utilizada como mera intermediaria entre el hardware y el software. El montaje que puede apreciarse en la *ilustración 3* consiste en conectar las señales de control de los respectivos motores a las salidas 9 y 10 de PWM del *Arduino* y alimentarlos a 5V.

El control de los servos se hace mediante Matlab, y el código puede consultarse comentado paso por paso en el capítulo 5 y completo en el anexo D.

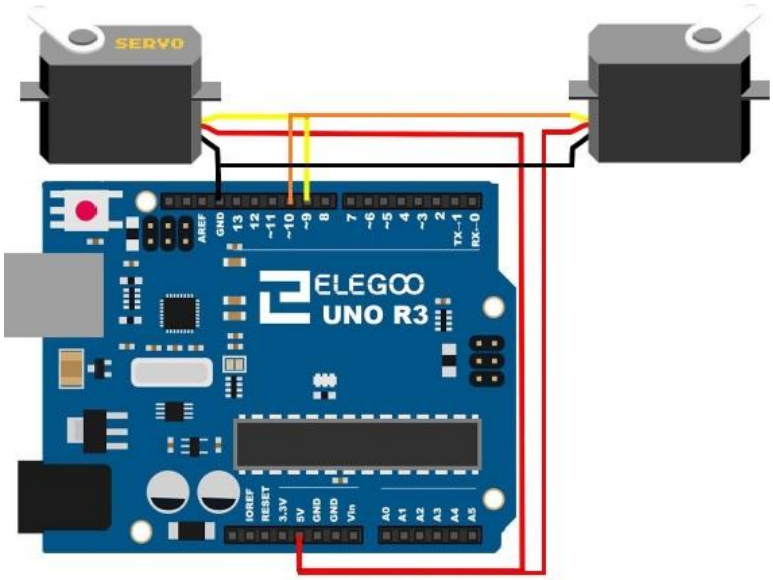

Ilustración 3: Montaje dos servomotores en paralelo en placa de Arduino

1.5.2 Software

En cuanto a software se ha utilizado Matlab, herramienta básica de investigación en ingeniería, tanto para la programación de los servomotores como para la creación de la red neuronal. Para la programación de los servos ha sido necesario instalar la librería *Matlab Support Package for Arduino Hardware* y para la creación de la RNA se ha empleado el *Neural Network Toolbox*, que proporciona funciones y aplicaciones para poder crear, simular y entrenar RNAs.

Además, ha sido necesario instalar la librería BioSig para poder procesar los archivos biomédicos (archivos de entrada a la red neuronal) cuyo formato gdf (General Data Format) no es compatible directamente con Matlab.

2 EL CEREBRO

El cerebro, situado al final de la médula espinal, forma parte del Sistema Nervioso Central. Junto a él, formando el encéfalo se encuentran otros órganos sumamente importantes para el funcionamiento del cuerpo humano como el tálamo, el hipotálamo, el bulbo raquídeo, los colículos, el puente o el cerebelo [6].

El cerebro se divide en dos hemisferios (izquierdo y derecho) unidos por el conocido como cuerpo calloso y, cada uno de ellos, controla principalmente a las partes contrarias del cuerpo (es decir, el hemisferio izquierdo del cerebro controla la parte derecha del cuerpo). Además, en cada hemisferio se desarrollan en mayor medida ciertas conductas; por ejemplo, en el hemisferio izquierdo se desarrolla el lenguaje, la lógica y las matemáticas mientras que en el hemisferio derecho se desarrolla la visión espacial, el arte y la música.

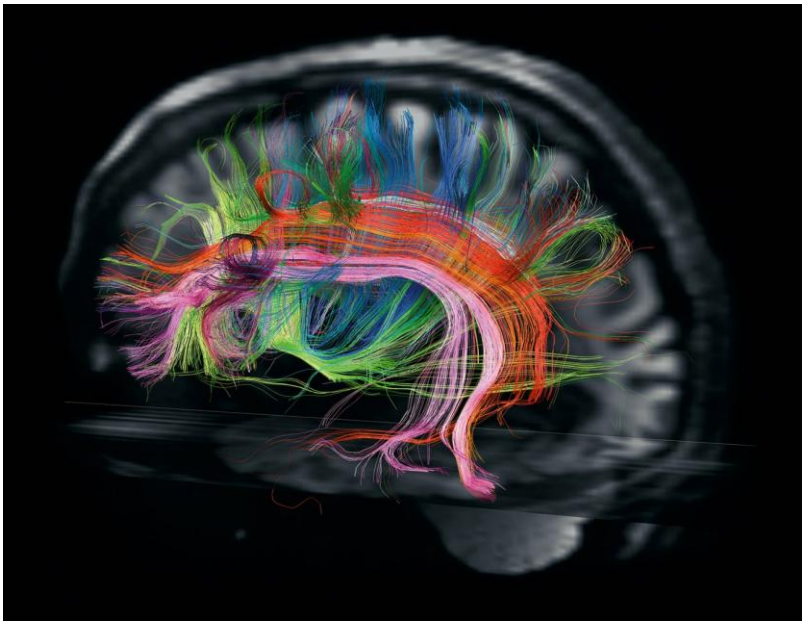

Ilustración 4: Rutas específicas de determinadas funciones cognitivas en un cerebro humano [6]

Las células que forman el cerebro son conocidas como neuronas y se encargan de recibir, procesar y transmitir información a través de impulsos eléctricos y reacciones químicas. En el sistema nervioso existen una gran diversidad de neuronas que se adaptan y se especializan según las funciones a llevar a cabo para que la comunicación entre estas sea lo más rápida y eficiente posible. La comunicación entre las más de 80 millones de neuronas de un cerebro humano se lleva a cabo en un área determinada de estas conocida como sinapsis, donde se produce la estimulación eléctrica de una neurona a otra.

Cada neurona, al conectarse con miles de neuronas a su alrededor y comunicarse, forma un patrón rítmico y repetitivo conocido como onda cerebral, que puede relacionar neuronas individualmente o grupos de estas. Las ondas cerebrales son, pues, las variaciones locales de potencial eléctrico en puntos dados de la corteza cerebral.

2.1 Ondas cerebrales

Mediante un EEG pueden estudiarse estas ondas y establecerse una clasificación básica según su frecuencia. Esta clasificación se realiza en base a lo denominado un EEG normal, en el que no se encuentran trazados extraños ni alteraciones de ninguna enfermedad. A pesar de que el EEG es utilizado para diagnosticar diversas enfermedades, no todas pueden detectarse con este método.

Las **ondas Delta** ($0.2 - 4Hz$) son las más lentas y de mayor amplitud, llegando a tener un voltaje del doble o el cuádruple que el resto de ondas. Son características del sueño profundo y reparador, del trance, o de la época de lactancia y juventud, desapareciendo conforme crecemos. Su aparición fuera de estos casos suele implicar enfermedades cerebrales.

Ilustración 5: ondas Delta ($0.2 - 4Hz$*)*

Las **ondas Theta** ($4 - 8Hz$) aparecen con el procesamiento de información interna, la desconexión del mundo exterior, y con la experimentación de emociones muy profundas principalmente miedos y traumas. Está también relacionada con el aprendizaje y la memoria y con la transición entre la vigilia y el sueño. En niños aparece solo en las regiones parietal y temporal del cerebro, mientras que en adultos es muy frecuente apareciendo en situaciones de estrés emocional (frustración, desánimo) y en trastornos nerviosos.

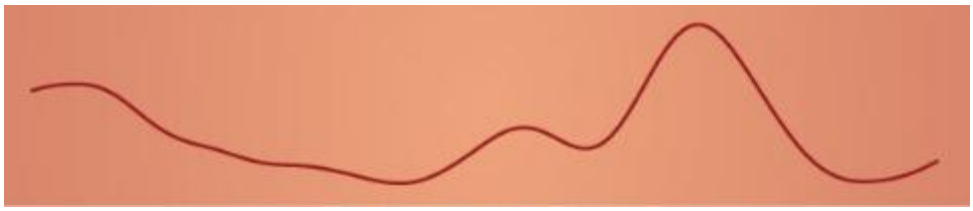

Ilustración 6: ondas Theta ($4 - 8Hz$*)*

Las **ondas Alpha** ($8 - 12Hz$) son visibles cuando el sistema nervioso central se encuentra en reposo, pero despierto y atento. Son características de la coordinación mental y de la integración cuerpo-mente, y desaparecen con el sueño profundo. Tienen un voltaje aproximado de $50\mu V$ en adultos, llegando a tener una amplitud de $100\mu V$ o $200\mu V$ en niños y decreciendo con la madurez. No se conoce ninguna relación clara entre la cantidad de ritmo Alpha y determinados parámetros psicológicos como la personalidad o la inteligencia.

Ilustración 7: ondas Alpha ($8 - 12Hz$*)*

Las **ondas Beta** ($12 - 30Hz$) aparecen con la atención dirigida a tareas cognitivas externas, con la atención y resolución de tareas o problemas cotidianos, con la toma de decisiones y la concentración. Su frecuencia es alta y su voltaje bajo. Pueden subdividirse a su vez en ondas Beta 1 ($12 - 15Hz$), ondas Beta 2 ($15 - 22Hz$) (que aparecen en el SNC cuando está comprometido en una tarea) y las ondas Beta 3 ($22 - 30Hz$) (características del SNC concentrado en cogniciones complejas o nuevas experiencias, o con ansiedad o excitación). Tiene una amplitud típica de $5 - 10\mu V$, aunque puede superar los $30\mu V$.

Ilustración 8: ondas Beta ($12 - 30Hz$*)*

Por último, las **ondas Gamma** ($30 - 90Hz$) son las de mayor frecuencia, y aparecen en ráfagas cortas cuando se da información simultánea en varias áreas del SNC. Se corresponden con un estado de alta resolución y modulan las percepciones y la consciencia. También se relaciona con estados de espiritualidad y meditación.

Ilustración 9: ondas Gamma ($30 - 90Hz$*)*

Podemos encontrar además el **ritmo Mu** ($8 - 10$Hz) que aparece con los estímulos sensoriales y se suprime o atenúa con los movimientos de extremidades y otras ondas como la V, que aparece durante el sueño, y la Lambda, que se relacionan con el movimiento y la búsqueda de elementos con los ojos.

Para el caso de estudio las ondas más interesantes son las Mu, que tal y como ya se ha comentado, están relacionadas con el movimiento de las extremidades. Aunque tienen una frecuencia prácticamente igual que las ondas Alpha, se diferencian de estas en que aparecen en trenes de pocos segundos de duración y en que tienen un aspecto más agudo y menos sinusoidal que las Alpha. En cuanto a la distribución en el cerebro, se encuentran en las regiones centrales, unilateral o bilateral, denominándose también actividad alpha precentral. [7]

2.2 Brain Computer Interface (BCI)

La interfaz cerebro-ordenador se trata de un sistema de comunicación que monitoriza la actividad cerebral para traducirla a comandos y órdenes para controlar un dispositivo.

Para registrar la actividad cerebral y poder monitorizarla existen varios métodos como el electroencefalograma (EEG), la electrocorticografía (ECoG), la magnetoencefalografía (MEG), la tomografía por emisión de positrones (PET) o las imágenes de resonancia magnética funcional (fMRI).

De entre las mencionadas, la más simple, barata, y la que, por tanto, suele usarse es el EEG; la ECoG es una técnica invasiva que requiere de una intervención para colocar los electrodos en la corteza cerebral, y las técnicas MEG, PET y FMIR requieren de instalaciones de alto coste. Sin embargo, el EEG es simple, barato, no invasivo, portátil y de bajo coste.

Según la naturaleza de la señal de entrada se pueden clasificar los sistemas BCI en dos grupos: endógenos y exógenos.

2.2.1 BCI endógenos

Estos sistemas dependen de la capacidad de controlar la actividad electrofisiológica del usuario y requieren un entrenamiento intensivo. Algunos ejemplos de sistemas BCI endóginos son:

- **BCI basados en potenciales corticales lentos (Slow Cortical Potentials, SCP):** los SCP son cambios de voltaje lentos generados en la corteza cerebral, con una duración de entre 0.5 y 10 segundos. Si el voltaje es negativo, el SCP suele asociarse con el movimiento.

  Se ha demostrado que se puede aprender a controlar estos potenciales.

- **BCI basados en imágenes motoras o ritmos sensoriomotores:** se basa en un modelo de dos o más clases de imágenes motoras (es decir, movimiento de manos, pies o lengua) u otras tareas mentales (cálculos aritméticos) que producen cambios en la amplitud de las ondas cerebrales Mu y Beta. Se aprecian variaciones en estos ritmos tanto en las ejecuciones de movimientos del cuerpo como para la preparación de estos o para movimientos imaginados.

2.2.2 BCI exógenos

Estos sistemas dependen de la actividad electrofisiológica evocada por estímulos externos y no requieren de una etapa de entrenamiento. Algunos ejemplos de sistemas BCI exógenos son:

- **BCI basados en potenciales evocados:** los potenciales evocados son cambios producidos en las ondas cerebrales al realizar una tarea y pueden ser previos o posteriores a esta. Algunos de los potenciales evocados más conocidos son:
  - LPR: potencial negativo producido al realizar un movimiento con una mano.
  - CNV: onda negativa lenta que se produce cuando se presentan dos estímulos separados por un intervalo de tiempo.
  - N100: onda negativa que aparece 100 ms después de un estímulo.
  - P300: pico positivo que aparece 300 ms después de un estímulo relevante para el usuario.

  Cada potencial evocado es característico de una zona del cerebro por lo que puede identificarse y medirse con moderada facilidad.

- **BCI basados en potenciales evocados visuales de estado estable:** estos potenciales se detectan en el EGG sobre la zona visual de la corteza tras la aplicación de un estímulo visual. Cuando el sujeto mira una imagen que parpardea a más de 6 Hz es posible detectarlo.

En un BCI es fundamental el procesado de la señal y puede dividirse en cuatro etapas: **pre-procesado**, donde se eliminan posibles artefactos que distorsionan la señal de interés (por ejemplo, el parpadeo), **extracción de características de la señal del EEG**, **métodos de selección de características significativas del usuario** y **la clasificación del conjunto de características**, que acaban traduciéndose en un comando. [8]

2.3 Posible análisis de ondas cerebrales

Como se comenta en el apartado 1.2, este trabajo comienza con la construcción de un EEG en el proyecto *Neurotronics*. Con este proyecto se ha conseguido extraer señales cerebrales de un sujeto mediante un casco de electrodos, posteriormente filtradas y recogidas en LabView. Sin embargo, para poder hacer un estudio profundo de las características de estas señales y posteriormente poder utilizarlas en una red neuronal, es necesario hacer un análisis previo con un programa como *Brainstorm*.

2.3.1 Brainstorm

Brainstorm es un recurso abierto y gratuito dedicado al análisis de grabaciones cerebrales como pueden ser el MEG, EEG, PET, fMRI o el ECoG. Esta incluye una interfaz gráfica relativamente simple dedicada al análisis de resultados, que no requiere de un conocimiento profundo de medicina ni de programación para ser utilizado y que puede añadirse como toolbox a Matlab [9]. Esta aplicación está realizada por la Centre National de la Recherche Scientifique (CNRS, France) para el *Cognitive Neuroscience & Brain Imaging Laboratory* (La Salpetriere Hospital and Pierre & Marie Curie University, Paris, France) y por el Montreal Neurological Institute para el *MEG Program* en la McGill University, con el apoyo de otras entidades como la French National Research Agency (ANR) y el Epilepsy Center in the Cleveland Clinic Neurological Institute.

Ilustración 10: Logo de Braistorm

2.3.2 Qué permite Brainstorm

Este toolbox tiene numerosas aplicaciones. Entre las más interesantes se encuentran:

- Digitalizar la posición de los electrodos utilizados en un EEG.
- Digitalizar la cabeza del sujeto y dar un modelo en 3D de esta.
- Importar datos a Matlab.
- Convertir y permitir leer datos en los formatos más conocidos (.gdf, por ejemplo).
- Corrección de artefactos.

Es decir, esta aplicación permite convertir las señales obtenidas del EEG en un archivo .gdf legible para nuestro código en Matlab de extracción de características (apartado 3.4) y posteriormente, por nuestra red neuronal.

2.3.3 Conversión a archivo .gdf

En primer lugar, será necesario crear un protocolo en Brainstorm, lo que es, en resumidas cuentas, un directorio donde se encontrarán todos los experimentos de un mismo sujeto. Como vemos en la *ilustración 13*, Brainstorm requiere cargar la anatomía del sujeto; es decir, un escáner cerebral. Aquí está el cuello de botella que se ha encontrado en esta aplicación: en este proyecto no se cuenta con los medios para obtener un escáner cerebral de un sujeto. Sin embargo, se explicará brevemente qué podría hacerse de contar con ello.

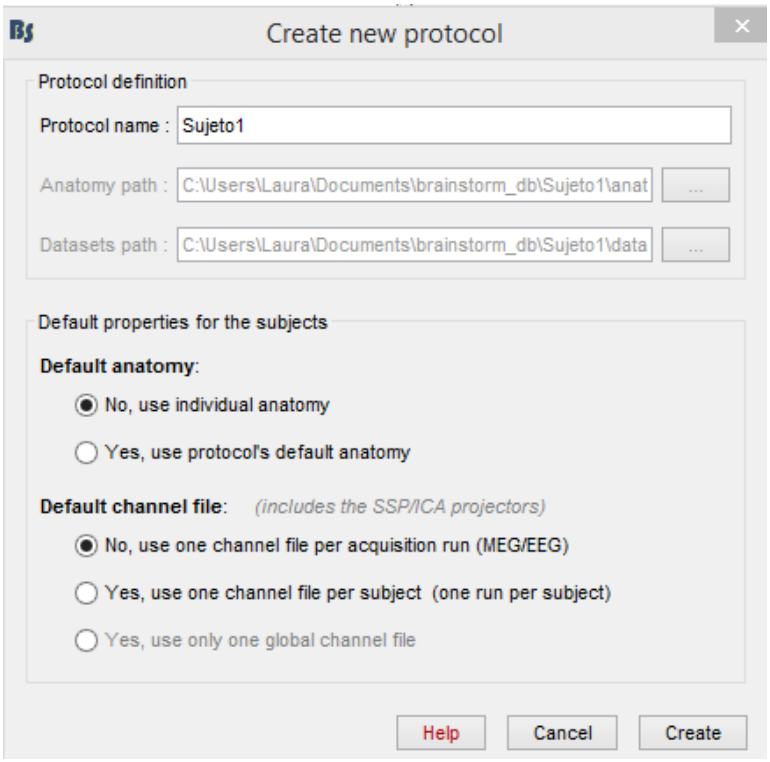

Ilustración 11: Creación de un nuevo protocolo en Brainstorm

A continuación será necesario crear un «Subject» dentro del protocolo del sujeto e importar la anatomía. En este caso de demostración se está utilizando una anatomía aportada por el curso de tutoriales de Brainstorm; de estar en un caso real, sería necesario tomar los resultados de la resonancia magnética y procesarlos con FreeSurfer, BrainSuite, BrainVISA o CIVET, que son los programas soportados por Brainstorm. Acto seguido, aparecerán las vistas coronal, sagital y axial del sujeto:

Ilustración 12: Vistas coronal, axial y sagital del sujeto

Con los botones que aparecen en el cuadrante inferior derecho, se podrá establecer los puntos fiduciales, es decir, el nasión (NAS), la oreja izquierda (LPA), la oreja derecha (RPA), la comisura anterior (AC), la comisura posterior (PC) y el punto interhemisférico (IH), que son utilizados para registrar los sensores del EEG y para alinear los puntos del sujeto con los puntos anatómicos típicos.

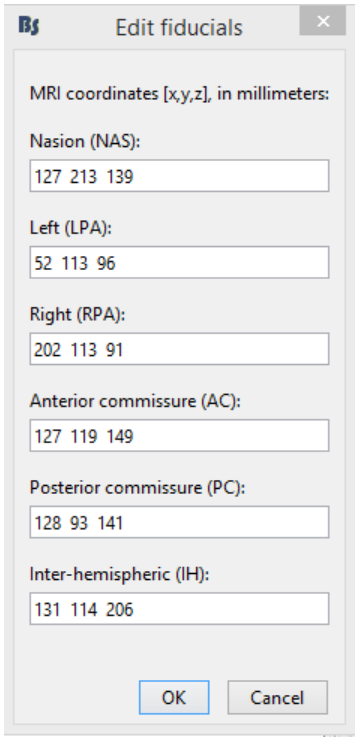

Ilustración 13: Coordenadas de los puntos fiduciales del sujeto de demostración

Gracias a Brainstorm y al escáner cerebral del sujeto, puede obtenerse una vista 3D de la cabeza, la calavera exterior, la calavera interior y del cerebro, tanto por separado como en una misma imagen.

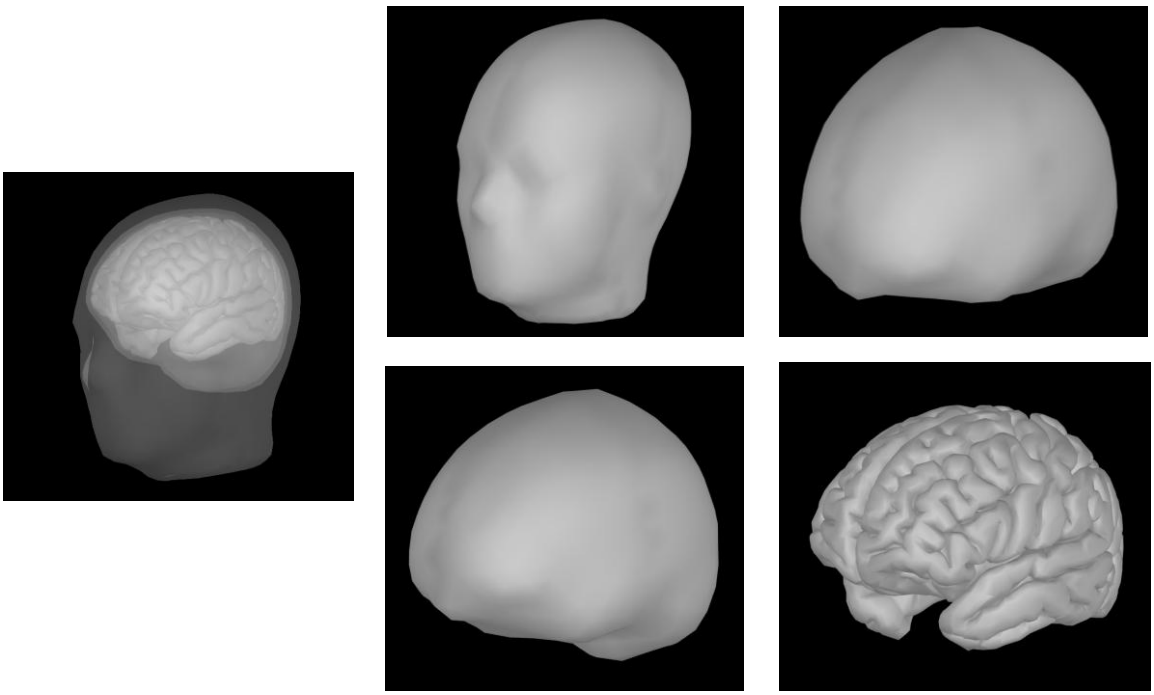

Ilustración 14: vistas 3D de cabeza, calavera exterior, calavera interior y cerebro (en orden de izquierda a derecha y de arriba a bajo). A la izquierda superpuestos en una misma imagen, a la derecha por separado.

Todos esta información pueden exportarse a Matlab en archivos .mat.

A continuación de los pasos comentados ya podrían importarse las grabaciones de los experimentos realizados al sujeto. En primer lugar, nos aparecerán los resultados de todos los canales y se podrán seleccionar fácilmente los canales que se quieran analizar:

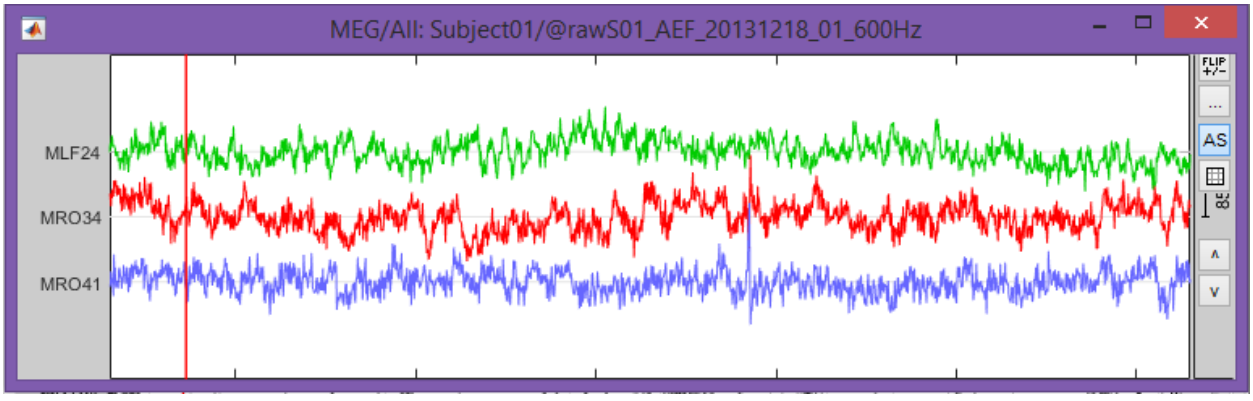

Ilustración 15: Selección de tres canales del EEG

Lo más interesante que puede hacerse con Brainstorm es la colocación de los eventos que se requieran. Es decir, una vez obtenidas las señales del cerebro, pueden marcarse momentos y/o segmentos sobre estas señales: instantes de pestañeo, puntos de testeo, momentos en los que el sujeto movió un brazo, etc., según la clasificación que se quiera, pudiendo así eliminar los segmentos que contengan errores y clasificar los que tengan utilidad para el proyecto.

Una vez hecho esto, se obtendría el fichero en la forma necesaria para comenzar con la extracción de características y el entrenamiento de la red neuronal.

2.4 Caso de estudio

Para el caso de estudio de este proyecto podría haberse creado tanto un BCI basado en imágenes motoras como en potenciales evocados. No obstante, se parte de un proyecto previo que comenzó con una extracción de características para un BCI basado en imágenes motoras y se ha seguido esta metodología.

Recapitulando desde el punto 3.1, concluimos que a la hora de mover una extremidad se suprimirá la onda mu del lado contrario del cuerpo. Es decir, si se mueve el brazo izquierdo, se suprimirá la onda mu del hemisferio derecho del cerebro, [10] siendo esto importante para la extracción de características previa al diseño de la red neuronal.

Sabiendo esto se puede pasar a la extracción de características, que se toma del proyecto realizado por Nagore Peirotén [10].

2.4.1 Extracción de características

Dado que la onda Mu está contenida en la misma frecuencia que la Alpha, se analizarán las ondas cerebrales Alpha y Beta pues son los ritmos que varían con el movimiento. Se tomarán las ondas obtenidas en los electrodos C3 (hemisferio izquierdo) y C4 (hemisferio derecho) durante los instantes de referencia y de movimiento. Las características a calcular y analizar serán:

- **El valor medio**: media aritmética de todos los valores que toma la señal en un intervalo de tiempo.
- **La desviación típica**: medida que proporciona la dispersión o variación de la señal con respecto a su valor medio.
- **La varianza**: medida que proporciona la dispersión o variación de la señal con respecto a su esperanza. Puede calcularse como el cuadrado de la desviación típica.
- **El valor mínimo**: el valor más bajo que toma la señal.
- **El valor máximo**: el valor más alto que toma la señal.
- **La PSD**: función que proporciona información sobre la distribución de la potencia de la señal sobre las distintas frecuencias que la forman.

Estas seis características tomadas en dos canales distintos de Alpha y Beta dan un total de 48 datos de entrada para la red neuronal.

En el proyecto *Diseño de una red neuronal en Matlab para análisis de señales de electroencefalograma* Nagore Peirotén [10] se encuentra una explicación más extensa de la extracción de características.

3 EXPERIMENTO DE PARTIDA

Como se ha comentado en el capítulo anterior, a pesar de que se han podido obtener resultados con el EEG realizado con el equipo de *Neurotronics*, para poder utilizar estos datos como entrada para la red neuronal es necesario que pasen por un tratamiento con un programa como *Brainstorm* que exige además un escáner cerebral. Como esto sale de las posibilidades de este proyecto, se ha optado por utilizar unos datos disponibles en internet proporcionados por la Graz University of Technology, Austria, donde comparten los datos obtenidos de realizar un EEG a 9 sujetos distintos. A cada sujeto se le realizó el electroencefalograma a la vez que ejecutaban cuatro tipos de tareas motoras: movimiento de la mano izquierda (clase 1), movimiento de la mano derecha (clase 2), movimiento de ambos pies (clase 3) y movimiento de la lengua (clase 4). Para cada sujeto se realizaron dos sesiones en diferentes días, y en cada sesión se hicieron 6 ejecuciones separadas en cortos periodos de tiempo, constando cada ejecución de 48 intentos (12 para cada clase posible) obteniendo un total de 288 ejecuciones por sesión.

El experimento realizado se basa en los sujetos sentados frente a una pantalla de ordenador en la que se le van dando las instrucciones a seguir. Se comienza ($t = 0s$) con una cruz en la pantalla acompañada de un tono de alerta. Dos segundos más tarde ($t = 2s$), una flecha apuntando a izquierda, derecha, abajo o arriba (clases de movimiento 1, 2, 3 o 4 respectivamente) aparece y permanece en pantalla durante $1.25s$. Se pide a los sujetos que mantengan el movimiento hasta que desaparezca una cruz similar a la primera de la pantalla en $t = 6s$.

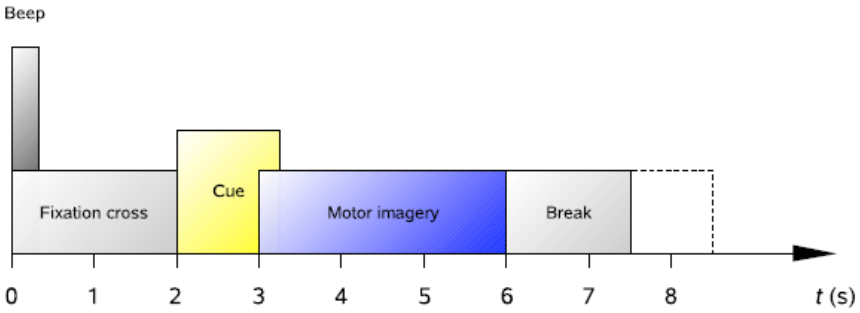

Ilustración 16: Esquema del experimento realizado

Para la grabación de los datos se han utilizado 22 electrodos con el montaje internacional 10-20 mostrado en la *ilustración 5*. Este montaje sitúa los electrodos entre un 10% y 20% de la distancia total entre puntos reconocibles del cráneo (nasión, inión y punto preauricular).

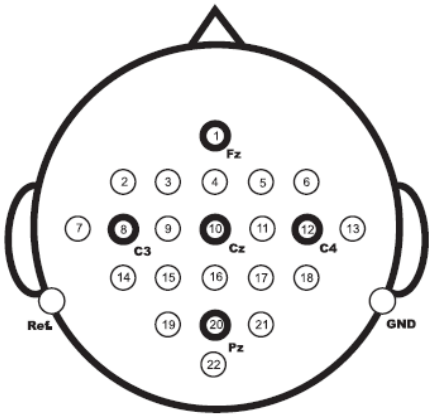

Ilustración 17: Montaje correspondiente al sistema internacional 10-20

A estos 22 electrodos se le añaden 3 canales EOG monopolares, es decir, en lugar de hacer una comparación de voltaje entre dos puntos con actividad cerebral se compara el voltaje de un punto con actividad cerebral y un punto inactivo o de reposo. Estos 3 canales están muestreados a 250 Hz, filtrados con un paso banda entre 0.5 Hz y 100 Hz (con el filtro Notch a 50Hz habilitado, eliminando esta banda de frecuencia) y con la sensibilidad del amplificador fijada a 1 mV, consiguiendo el máximo nivel de salida con este voltaje.

Como se ha comentado anteriormente, los datos vienen dados en formato gdf para señales biomédicas. Se realiza un archivo por sesión, pero solo una de las sesiones contiene las clases de movimiento dadas en cada instante, mientras que los otros archivos no proporcionan información adicional para este proyecto.

4 REDES NEURONALES ARTIFICIALES (RNA)

Las redes neuronales artificiales son actualmente una de las formas más fáciles de crear Inteligencia Artificial (IA). Se basan en una simplificación de la sinapsis del cerebro humano para funcionar. Como se ha visto al comienzo del punto anterior, las neuronas del cerebro humano se comunican con el resto de neuronas mediante la sinapsis y las redes neuronales artificiales crean un modelo simplificado de esta.

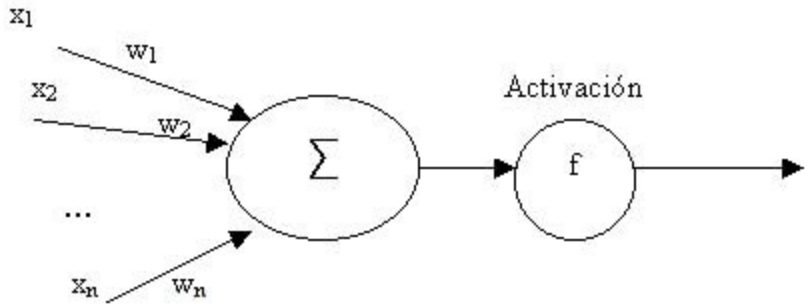

Ilustración 18: Modelo de neurona artificial simplificada

En la *ilustración 18* puede verse como la neurona artificial está formada por un vector de entradas *x* de *n* componentes y un conjunto de pesos sinápticos *w* que pasan a la función de activación definiendo así la salida de la neurona.

Las RNA se organizan por nodos de neuronas de entrada, nodos de neuronas ocultos y nodos de neuronas de salida.

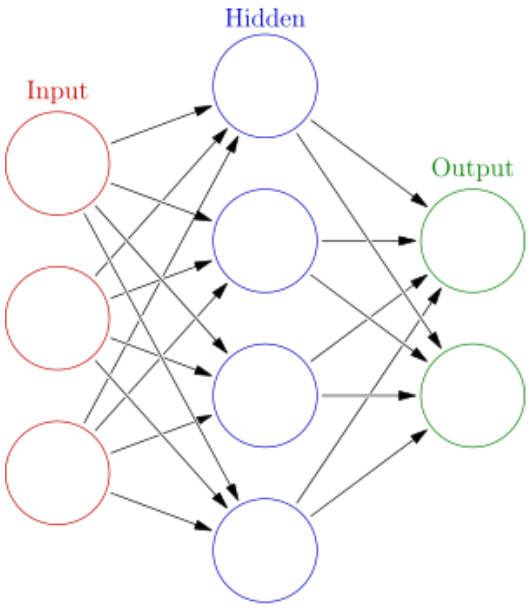

Ilustración 19: Nodos de una RNA

Los nodos de entrada reciben información del exterior mientras que los nodos de salida transmiten la información hacia el exterior. Sin embargo, los nodos ocultos no tienen intercambio de información con el exterior, sino que intercambian información con otros nodos de la red con los que están fuertemente conectados.

Los nodos ocultos se organizan en capas, pudiendo tener un sistema con topología monocapa, con una sola columna de nodos ocultos entre la entrada y la salida (*Figura 10*), o topología multicapa, con varias columnas de capas ocultas entre la columna de nodos de entrada y la de nodos de salida.

Para crear una RNA pasaremos por varias fases, que veremos a continuación.

4.1 Topología de la red

En la fase de creación de la RNA habrá que diseñar la arquitectura de la misma, es decir, habrá que determinar el número de neuronas que la formen, su disposición en capas y la conectividad entre estas, además de la cantidad de entradas y salidas que tendrá. La topología de la red determinará la cantidad de conocimiento que puede albergar y estará relacionada con el algoritmo de entrenamiento de la red, que se explicará en el apartado 4.3.

Podemos encontrar varios tipos de clasificación según la topología, entre ellos:

- Según el número de capas de la red:
    - **Red monocapa:** la red cuenta con un vector de entrada que, sin procesar la información, la envía a una capa de neuronas que se encarga de modificar los datos y darles salida. En este caso solo se dispone de una capa de procesamiento.
    - **Red multicapa**: la red tiene una capa de entrada con *n* neuronas, una capa de salida con *m* neuronas y cierto número de capas ocultas formadas por *h* neuronas.
- Según el flujo de información:
    - **Redes feedforward:** en este tipo de redes la información fluye en un único sentido, desde la entrada hasta la capa de procesamiento y de ahí hasta la salida.
    - **Redes recurrentes:** en estas redes la información no fluye en un único sentido, sino que realimenta capas anteriores. [11]

Es necesario determinar las funciones de activación y transferencia que se usarán. La función de activación es la encargada de relacionar la información de entrada de la neurona con el siguiente estado de activación que tenga esa neurona. Existen dos modelos:

- Modelos acotados: el valor de la activación de la neurona puede ser cualquiera dentro de su rango continuo de valores.
- Modelos no acotados: no existe ningún límite para los valores de activación.

Normalmente se utiliza la misma función de activación para todas las neuronas.

Con respecto a las funciones de activación y de transferencia hay que tener en cuenta que existen dos tipos de neuronas:

- **Neuronas lineales:** cuando su salida es linealmente dependiente de sus entradas, es decir, proporcional a las funciones de transferencia y de activación. Esto puede producir problemas por falta de persistencia en las respuestas, ya que pequeños cambios en la entrada pueden producir grandes fluctuaciones a la salida.
- **Neuronas no lineales**: la función de activación, la de transferencia o ambas son funciones no lineales, de manera que la respuesta de la neurona no será función lineal de sus entradas. Esto producirá respuestas acotadas, desapareciendo problemas de fluctuación y falta de adecuación.

4.2 Regla de aprendizaje

El tipo de aprendizaje en una red neuronal artificial está determinado por cómo cambian los parámetros de esta durante el proceso.

- **Aprendizaje supervisado:** se le presentan a la red patrones de estímulos de entrada de forma repetitiva. Estos patrones junto con su respectiva salida correcta forman un juego de ensayo.

  En el juego de ensayo deberá estar representada de forma equilibrada toda la información que la red debe aprender.

  Al realizar el entrenamiento, la respuesta que dé la red se comparará con la respuesta correcta, reajustando los pesos sinápticos conforme sea necesario. Una vez conseguido un patrón correcto, se pasa al siguiente; cuando se terminen todos se vuelve al primer patrón y se reajusta hasta conseguir todos los resultados correctos. Cuando esto ocurre, el algoritmo de aprendizaje converge y finaliza la fase de aprendizaje.

- **Aprendizaje no supervisado o autoorganizativo:** a la red no se le especifica cuál debe ser la respuesta correcta ni se le informa de si los resultados son o no correctos. Simplemente se le suministran grandes cantidades de datos con las que construye sus propias asociaciones, exigiéndosele que capte por sí misma las características de los datos de entrada.

  Las neuronas se auto-organizan aprendiendo a captar las regularidades de los datos de entrada sin ningún criterio ni ayuda.

- **Aprendizaje híbrido o por reforzamiento:** no se dan patrones de salida correctos, solo se le dice a la red si la salida es o no correcta según patrones de entrada dados.

4.3 Algoritmo de entrenamiento

Una vez determinado el tipo de aprendizaje a llevar a cabo se elegirá el algoritmo de entrenamiento que será, en términos principales, una función de minimización de error.

Esta función está formada por dos términos: el primero, el *error*, será el término que evalúe cómo se ajusta la salida de la red a la salida denominada como «correcta»*;* el segundo será el término de *regularización*, que controla la complejidad de la red neuronal evitando así el sobreaprendizaje.

El valor de error que comete la red depende de los pesos asociados a sus neuronas.

Algunos de los algoritmos de entrenamiento más conocidos y utilizados son:

- **Descenso del gradiente:** solo hace uso del vector gradiente, por lo que se le considera un método de primer orden. Es el algoritmo que se recomienda utilizar en redes neuronales muy grandes con miles de parámetros; de lo contrario, puede obtenerse una disminución rápida del error, pero no producir la convergencia rápidamente.

- **Método de Newton:** hace uso de la Hessiana, por lo que se conoce como un algoritmo de segundo orden. Este método obtiene primero la dirección de entrenamiento y después la velocidad de entrenamiento adecuada. Por lo general requiere de menos iteraciones que el descenso del Gradiente para minimizar el error, pero el cálculo de la Hessiana y su inversa (también necesaria) son muy costosos computacionalmente.

- **Gradiente Conjugado:** se considera intermedio a los dos anteriores, intentando acelerar la convergencia del Descenso del Gradiente sin requerir el coste computacional del método de Newton. También se recomienda para redes neuronales muy grandes.

- **Método Cuasi-Newton:** para intentar disminir el coste computacional del método de Newton se propone construir una aproximación de la inversa de la Hessiana en cada iteración utilizando información sobre las primeras derivadas de la función de error. Es más rápido que los anteriores

algoritmos, por lo que es bastante habitual su uso.

- **Algoritmo de Levenberg-Marqueardt o método de mínimos cuadrados amortiguado:** utiliza las funciones de error que se expresan como suma de errores cuadráticos. Hace uso del vector del gradiente y de la matrz Jacobiana, sin necesidad de la Hessiana. No se recomienda para redes grandes. [12]

4.3.1 Problemas comunes en el entrenamiento

Es habitual encontrar problemas durante la fase de entrenamiento. Los más frecuentes son:

- Que la red no se entrene con precisión suficiente, dando lugar a un alto porcentaje de fallos imposibles de reducir o a un entrenamiento muy lento. Para solucionarlo habría que analizar la red y el juego de ensayo, estudiando si es el más óptimo para la aplicación que buscamos.
- Sobreentrenamiento: es un problema muy común en el que la red aprende los patrones de entrada, pero no es capaz de generalizar ni encontrar relaciones entre ellos. Puede darse por dos motivos:
    - Que la topología de la red sea muy compleja. Es decir, que el número de capas sea muy grande para el problema que se intenta resolver.
    - Que la red se haya entrenado durante demasiados ciclos.
- Entrenar a la red con un juego de ensayo que tiene ruido. De esta forma, se falsea la entrada de la red neuronal y la esta pierde la capacidad de generalizar.

4.4 Fase de validación

Una vez terminado el aprendizaje, la red puede generalizar y dar respuestas correctas ante nuevos patrones. Para que el rendimiento de la red sea el mayor posible, los datos usados en el entrenamiento deben cubrir un rango amplio y no podrán utilizarse para la fase de validación.

5 REALIZACIÓN Y RESULTADOS

5.1 Extracción de características y entrada de la RNA

Para el caso de este proyecto se ha utilizado el Neural Network Toolbox de Matlab para crear la RNA, el cual utiliza un tipo de aprendizaje supervisado mediante un algoritmo de entrenamiento de Levenberg-Marquardt.

Al final del capítulo 2 se comentaba la extracción de características de las señales tomadas de los electrodos C3 y C4 correspondientes a las ondas Alpha y Beta llevada a cabo. En este punto, esas características pasan a ser los datos de entrada de la red neuronal.

Con el objetivo de aumentar la precisión de la RNA se han variado los datos de entrada de la red neuronal con respecto al proyecto de partida ya comentado [10]. Para conseguir esto se han utilizado datos más generales y diferentes; es decir, en este trabajo se ha intentado diferenciar el movimiento del brazo izquierdo de otros movimientos como el del brazo derecho, el de la lengua y el de los pies. En el Anexo A puede verse el código que construye las matrices de entrada y salida de entrenamiento de la red. Además, puede verse en el extracto a continuación que se toman los eventos relacionados con el brazo izquierdo como correctos *(Salida=1)* y el resto como incorrecto *(Salida=0)*.

```
if h.EVENT.TYP(i)==770 % Movimiento mano izquierda
   Salida(evento)=1;
end
if (h.EVENT.TYP(i)==769 || h.EVENT.TYP(i)==771 ||
    h.EVENT.TYP(i)==772)%Si es cualquier otro tipo de movimiento
   Salida(evento)=0;
end
```

5.2 RNA

Una vez construidas las matrices de entrada y salida necesarias para entrenar a la red, se pasa a la topología de la red. Gracias a las facilidades que proporciona el Toolbox de Matlab, la topología se ha elegido mediante prueba y error llegando a la conclusión de que los mejores resultados se obtienen con 25 neuronas en la capa oculta y con la RNA entrenada 3 veces. En el Anexo B puede consultarse el código generado por Matlab de la RNA.

A continuación se muestra una tabla con los porcentajes de error obtenidos para distintas combinaciones de número de neuronas y reentrenamientos.

Tabla 2: Experimentos RNA

Nº neuronas en capa oculta	Nº de entrenamientos	%Error
10	1	Training: 23.63% Validation: 36.11% Testing: 19.44%
10	2	Training: 24.24% Validation: 25% Testing: 30.55%
20	1	Training: 23,63% Validation: 27.77% Testing: 16.66%
20	2	Training: 25.45% Validation: 19.44% Testing: 27.77%
50	1	Training: 28.48% Validation: 27.77% Testing: 30.55%
50	2	Training: 35.15% Validation: 13.88% Testing: 25%
100	1	Training: 23.03% Validation: 25% Testing: 33.33%
100	2	Training: 26.66% Validation: 13.88% Testing: 16.66%
1000	1	Training: 27.27% Validation: 33.33% Testing: 13.88%
25	3	Training: 17.57% Validation: 16.66% Testing: 25%

Estos mismos datos pueden verse de forma más clara en una matriz de confusión, donde se reflejan varios parámetros que se detallan a continuación. Para el caso utilizado se obtiene la siguiente matriz de confusión (proporcionada por el toolbox de Matlab).

Ilustración 20: Matriz de confusión para 25 neuronas y 3 re-entrenamientos

En esta ilustración podemos ver las matrices de confusión de las fases de entrenamiento, de validación, de test y la total. Observamos lo siguiente:

- **Tasa de verdaderos positivos**, es decir, porcentaje de casos en los que la red declara un positivo y efectivamente lo es. Se calcula como la suma de verdaderos positivos (posición 5 de la matriz) dividido entre la suma de casos positivos (posiciones 2 y 5 de la matriz). El resultado puede verse en la primera fila de la posición 8 de la matriz. Para más exactitud, el resultado es del 42.86%.
- **Tasa de verdaderos negativos**, es decir, el porcentaje de casos en los que la red declara un negativo y efectivamente lo es. Se calcula como la suma de verdaderos negativos (posición 1 de la matriz) divido entre el total de casos negativos (posiciones 1 y 4 de la matriz). El resultado puede verse en la primera fila de la posición 7 de la matriz. Para más exactitud, el resultado es del 95.98%.
- **Tasa de falsos positivos**, es decir, el porcentaje de casos en los que la red declara un positivo y no lo es. Se calcula como el número total de falsos positivos (posición 4 de la matriz) dividido entre el número total de negativos (posiciones 1 y 4 de la matriz). El resultado puede verse en la segunda fila de la posición 7 de la matriz. Para más exactitud, el resultado es del 4.02%.

- **Tasa de falsos negativos**, es decir, el porcentaje de casos en los que la red declara un negativo y no lo es. Se calcula como el número total de falsos negativos (posición 2 de la matriz) dividido entre el número total de casos positivos (posiciones 2 y 5 de la matriz). El resultado puede verse en la segunda fila de la posición 8 de la matriz. Para más exactitud, el resultado es del 57.14%
- **Precisión**, es decir, el total de casos definidos correctamente (posiciones 1 y 5) entre el total de casos tratados (posiciones 1, 2, 4 y 5). El resultado puede verse en la primera fila de la posición 9 de la matriz. Para más exactitud, el resultado es del 81.86%.
- **Exactitud**, es decir, el total de casos positivos verdaderos frente a los casos positivos falsos. Se obtiene como el número de positivos verdaderos (posición 5) entre el total de casos positivos registrados ya sean verdaderos o falsos (posiciones 4 y 5). El resultado puede verse en la primera fila de la posición 6 de la matriz. Para más exactitud, el resultado es del 79.41%.

5.3 Estudio post-RNA

Como salida de la RNA se obtienen valores entre 0 y 1 que, a priori, se clasificaron como:

- Valores entre 0 y 0.5 se denominan iguales a 0.
- Valores entre 0.5 y 1 se denominan iguales a 1.

Haciendo pruebas se pudo comprobar que la mayoría de los resultados erróneos se localizaban entre 0.3 y 0.7. Por lo tanto se pasó a la siguiente clasificación:

- Valores entre 0 y 0.3 se denominan iguales a 0.
- Valores entre 0.7 y 1 se denominan iguales a 1.
- Valores entre 0.3 y 0.7 se denominan "caso de estudio".

5.3.1 Casos de estudio

Para analizar estos casos se toma una muestra aleatoria de 36 pruebas (el 15% de las pruebas totales, el mismo porcentaje utilizado por Matlab) del mismo juego de datos puesto que no se disponen de más.

A estos datos se les aplica la RNA y, observando la salida, se hace un estudio de las características extraídas de las ondas cerebrales, comprobándose que:

- En los casos en los que la salida se encuentra entre 0.3 y 0.5 y eso es erróneo (es decir, la salida debería ser un 1 o un número próximo a este) se observa que la media de la PSD de la onda beta del canal C3 de la señal de referencia es un orden más alta que en los casos en los que la salida es un 0.
- En los casos en los que la salida se encuentra entre 0.5 y 0.7 y eso es erróneo (es decir, la salida debería ser un 0 o un número próximo a este) no se encuentra una característica destacable con la que poder hacer un estudio similar.

Por lo tanto, se hace una nueva clasificación:

- Valores entre 0 y 0.3 se denominan iguales a 0.
- Valores entre 0.7 y 1 se denominan iguales a 1.
- Valores entre 0.3 y 0.7 se denominan "caso de estudio".
    - Valores entre 0.3 y 0.5 cuya $PSD > 1 \cdot 10^4$ se denominan iguales a 1.

El código en el que se realiza este estudio puede consultarse en el anexo C.

5.4 Movimiento del brazo robótico

Como se comentó en el primer capítulo, es necesario instalar la librería *MATLAB Support Package for Arduino Hardware*. De esta forma, Matlab podrá comunicarse con una placa de Arduino por puerto serie y esta podrá programarse con un .mat.

En primer lugar, será necesario realizar las conexiones tanto de la placa (*a*) como de los servomotores (*s1*, *s2*) e inicializar los vectores que contendrán los ángulos de giro de los servomotores (*v1*, *v2*).

```
a=arduino('COM3', 'uno');
s1=servo(a,'D9');
s2=servo(a, 'D10');
v1=zeros(1,i);
v2=zeros(1,i);
```

De esta forma indicamos que los servomotores *s1* y *s2* de la placa *a* se encuentran conectados a las entradas digitales (PWM) 9 y 10.

A continuación se determina la posición del robot en función de la salida revisada de la RNA; es decir, la salida del estudio post-RNA. Como puede verse en el Anexo D, el tratamiento post-RNA se hace en este mismo archivo .mat.
En este caso, si el valor de la salida es 1 se moverá el servomotor *s1* a 150º y el servomotor *s2* a 120º. Para introducir estos valores en el vector que posteriormente pasará a los servos es necesario que las posiciones se encuentren en radianes.

```
for j=1:i
    if OutputMotor_rev(j)==1
        v1(j)=150/180;
        v2(j)=120/180;
```

En caso de que la salida sea 0, los servos se moverán a 90º; de nuevo, expresados en radianes. Esta posición será la que determinaremos como *reposo*.

```
 else
        v1(j)=90/180;
        v2(j)=90/180;
    end
end
```

Una vez construidos los vectores con las posiciones de los servomotores, se les pasará esta información. Las posiciones a la que los servos se moverán en cada instante se denominarán *ang1* y *ang2*, para *s1* y *s2* respectivamente. Con la orden *writePosition* denominaremos el servo y la posición a la que moverse.

```
M=length(v1);
for i=1:M

    ang1=v1(i);
    ang2=v2(i);
    writePosition(s1,ang1);
    writePosition(s2,ang2);
```

Para asegurar el buen funcionamiento del robot, leeremos la posición de los servos y las guardaremos en *pos1* y *pos2* para ambos servos de forma respectiva. Esto servirá como realimentación para el usuario, pudiendo utilizarse para disminuir el error.

```
    pos1=readPosition(s1);
    pos2=readPosition(s2);
```

A continuación, se imprimirá por pantalla la posición de los servos, que se mantendrá durante dos segundos.

```
    fprintf('Posición actual del Servo1 es de %d grados\n',pos1*180);
    fprintf('Posición actual del Servo2 es de %d grados\n',pos2*180);

    pause(2);
end
```

6 CONCLUSIONES

Estudiando los resultados conseguidos tras realizar el apartado 5.2 se puede comprobar que la precisión es del 81.86% lo cual, es un valor aceptable, sobre todo partiendo de la base de una red neuronal con un máximo del 67% de precisión. Sin embargo, el resto de resultados son muy dispares. Se muestra un pequeño estudio a continuación:

- La tasa de verdaderos positivos es relativamente baja, menor a un 50%. Traduciéndolo al funcionamiento del robot, habrá un número significativo de casos en los que el usuario querrá mover el brazo izquierdo pero el brazo robótico no ejercerá ningún movimiento.
- La tasa de verdaderos negativos es muy alta, de un 95.98%. Es decir, excepto en contadas ocasiones, si el usuario no quiere mover el brazo izquierdo, el robot no se moverá.

Aunque pueda parecer que la tasa de verdaderos positivos es desesperanzadora, se trata en realidad del parámetro cuyo valor es aceptable que sea más bajo, pues no supone un peligro para el usuario. Sin embargo, si el robot se moviera solo sin intención por parte del sujeto podría llegar a conllevarle riesgos.

En vista de que no se conseguían mejorar los parámetros de error de la RNA con cambios en las neuronas de la capa oculta ni con el número de entrenamientos, se pasó al estudio post-RNA que cruza los datos de las características extraídas de las ondas cerebrales con la salida de la RNA para conseguir una mayor precisión y que queda explicado en el apartado 5.3.

6.1 Estudio post-RNA de los datos

Como salida de la RNA se obtienen valores entre 0 y 1 que, a priori, se clasificaron como:

- Valores entre 0 y 0.5 se denominan iguales a 0.
- Valores entre 0.5 y 1 se denominan iguales a 1.

Con esto, comprobamos una precisión del 81.43%, es decir, un valor muy similar al que Matlab nos proporciona en la matriz de confusión.

Si se disminuye el valor de corte desde 0.5 hasta 0.3 se obtiene una precisión del 71.31%, es decir, se empeora.

Si se aumenta el valor de corte hasta 0.7 se obtiene una precisión del 56.54%, aún peor que la anterior.

Sin embargo, haciendo estas pruebas se pudo comprobar que la mayoría de los resultados erróneos se localizaban entre 0.3 y 0.7. Por lo tanto se pasó a la siguiente clasificación, ya comentada anteriormente:

- Valores entre 0 y 0.3 se denominan iguales a 0.
- Valores entre 0.7 y 1 se denominan iguales a 1.
- Valores entre 0.3 y 0.7 se denominan "caso de estudio".

6.1.1 Casos de estudio

Tal y como se ha explicado anteriormente, la clasificación queda de la siguiente forma:

- Valores entre 0 y 0.3 se denominan iguales a 0.
- Valores entre 0.7 y 1 se denominan iguales a 1.
- Valores entre 0.3 y 0.7:

- Valores entre 0.3 y 0.5 cuya $PSD > 1 \cdot 10^4$ se denominan iguales a 1.

  Realizando una nueva prueba aleatoria con 36 casos distintos, se consigue una precisión de un **91.56%.** Este estudio puede consultarse en el anexo C.

- Valores entre 0.5 y 0.7: no se obtiene ninguna característica significativa con la que mejorar los resultados.

Por lo tanto se concluye que, con un mayor estudio de las características extraídas de las ondas cerebrales se consiguen mejores resultados, superando incluso el 90% de precisión. Es decir, con un mayor conocimiento de las ondas cerebrales que pueda ser brindado por un especialista, tomando señales de más de dos electrodos y creando una red neuronal más compleja podría incrementarse con relativa facilidad este porcentaje hasta acercarse a un 100% de precisión.

Si además, pudiera dependerse de un paciente que proporcionase mayor número de ejemplos y que entrenara junto con la red neuronal, podría hacerse este mismo estudio diferenciando entre varios movimientos de un mismo brazo en lugar de diferenciando entre movimientos de varios miembros.

7 ANEXO A

```
%% SEPARACIÓN DE CANALES Y EXTRACCIÓN DE CARACTERÍSTICAS
clc; clear all;
% evento -> Variable para contar el numero de eventos de mano izquierda,
% derecha, lengua y pies
% eventos_totales -> Variable para contar el numero de eventos totales
% Salida -> Donde se almacenará si es movimiento de mano izquierda o no
% Si es mov. mano izquierda Salida=1, sino, Salida=0
evento=1;
malos=0;
eventos_totales=1;
[s,h]=sload('A09T.gdf'); % Se carga el archivo
for i=1:length(h.EVENT.TYP) % Se recorren los distintos eventos
    if (h.EVENT.TYP(i)==770 || h.EVENT.TYP(i)==771 || h.EVENT.TYP(i)==772 ||
h.EVENT.TYP(i)==769) % Si el evento es del tipo movimiento manos, pies o
lengua
        if h.ArtifactSelection(eventos_totales)==0 % Si es un evento libre de
artefactos
            muestra=h.EVENT.POS(i-1);
            if h.EVENT.TYP(i)==770 % Movimiento mano izquierda
               Salida(evento)=1; % Se le asigna un 1 a la salida
            end
            if (h.EVENT.TYP(i)==769 || h.EVENT.TYP(i)==771 ||
h.EVENT.TYP(i)==772)%Si es cualquier otro tipo de movimiento
               Salida(evento)=0;
            end
            if (h.EVENT.TYP(i)==770 || h.EVENT.TYP(i)==769  ||
h.EVENT.TYP(i)==771 || h.EVENT.TYP(i)==772)% Se realiza la separación de las
señales y la extracción de características
                cont=0;
                % Se separa cada uno de los eventos en dos señales: la
                % referencia y el movimiento.

                %Se obtiene la señal durante el movimiento y un poco antes
                %de que comience.
                for i=(muestra+700):(muestra+700+511)
                    cont=cont+1;
                    senal_mov(cont,:)=s(i,:);
                end
                cont=0;
                % Después se obtiene la señal en el periodo de referencia
                for i=(muestra):(muestra+511)
                    cont=cont+1;
                    senal_ref(cont,:)=s(i,:);
                end


                % SEPARACIÓN DE CANALES
                % Se separa cada una de las señales obtenidas previamente en
                  dos señales,cada una de un canal distinto
                   canal8_mov = senal_mov(:,8); %C3
                   canal12_mov = senal_mov(:,12); %C4

                   canal8_ref = senal_ref(:,8); %C3
                   canal12_ref = senal_ref(:,12); %C4
```

```
C3_mov = canal8_mov; %C3
C4_mov = canal12_mov; %C4

C3_ref = canal8_ref; %C3
C4_ref = canal12_ref; %C4

% Se cargan los filtros de paso banda previamente
diseñados
load('filtro_alfa_10.mat')
load('filtro_beta_4.mat')

% SE REALIZA EL FILTRADO DE LAS SEÑALES
% Ondas alfa
filtrada_C3_alfa_mov=filter(filtro_alfa_10,C3_mov);
filtrada_C4_alfa_mov=filter(filtro_alfa_10,C4_mov);
filtrada_C3_alfa_ref=filter(filtro_alfa_10,C3_ref);
filtrada_C4_alfa_ref=filter(filtro_alfa_10,C4_ref);
% Ondas beta
filtrada_C3_beta_mov=filter(filtro_beta_4,C3_mov);
filtrada_C4_beta_mov=filter(filtro_beta_4,C4_mov);
filtrada_C3_beta_ref=filter(filtro_beta_4,C3_ref);
filtrada_C4_beta_ref=filter(filtro_beta_4,C4_ref);


% PRIMERA CARACTERÍSTICA: VALOR MEDIO
% Se inicializan a 0 las sumas de cada una de las señales
suma_beta_C3_ref=0; suma_beta_C3_mov=0;
suma_beta_C4_ref=0; suma_beta_C4_mov=0;
suma_alfa_C3_ref=0; suma_alfa_C3_mov=0;
suma_alfa_C4_ref=0; suma_alfa_C4_mov=0;
L=length(filtrada_C3_beta_ref);
for i=1:L
    % Se realiza la suma de cada una de las componentes
    de cada señal
suma_beta_C3_ref = suma_beta_C3_ref + filtrada_C3_beta_ref(i);
suma_beta_C4_ref = suma_beta_C4_ref + filtrada_C4_beta_ref(i);
suma_beta_C3_mov = suma_beta_C3_mov + filtrada_C3_beta_mov(i);
suma_beta_C4_mov = suma_beta_C4_mov + filtrada_C4_beta_mov(i);
suma_alfa_C3_ref = suma_alfa_C3_ref + filtrada_C3_alfa_ref(i);
suma_alfa_C4_ref = suma_alfa_C4_ref + filtrada_C4_alfa_ref(i);
suma_alfa_C3_mov = suma_alfa_C3_mov + filtrada_C3_alfa_mov(i);
suma_alfa_C4_mov = suma_alfa_C4_mov + filtrada_C4_alfa_mov(i);
end
% Se realiza la media de cada una de las señales
media_beta_C3_ref = (1/L)*suma_beta_C3_ref;
media_beta_C3_mov = (1/L)*suma_beta_C3_mov;
media_beta_C4_ref = (1/L)*suma_beta_C4_ref;
media_beta_C4_mov = (1/L)*suma_beta_C4_mov;
media_alfa_C3_ref = (1/L)*suma_alfa_C3_ref;
media_alfa_C3_mov = (1/L)*suma_alfa_C3_mov;
media_alfa_C4_ref = (1/L)*suma_alfa_C4_ref;
media_alfa_C4_mov = (1/L)*suma_alfa_C4_mov;


% CARACTERÍSTICA 2: DESVIACIÓN TÍPICA
suma_2_beta_C3_ref=0; suma_2_beta_C3_mov=0;
suma_2_beta_C4_ref=0; suma_2_beta_C4_mov=0;
suma_2_alfa_C3_ref=0; suma_2_alfa_C3_mov=0;
suma_2_alfa_C4_ref=0; suma_2_alfa_C4_mov=0;
for i=1:L
```

```
suma_2_beta_C3_ref = suma_2_beta_C3_ref + (filtrada_C3_beta_ref(i)-
media_beta_C3_ref)^2;
suma_2_beta_C4_ref = suma_2_beta_C4_ref + (filtrada_C4_beta_ref(i)-
media_beta_C4_ref)^2;
suma_2_beta_C3_mov = suma_2_beta_C3_mov + (filtrada_C3_beta_mov(i)-
media_beta_C3_mov)^2;
suma_2_beta_C4_mov = suma_2_beta_C4_mov + (filtrada_C4_beta_mov(i)-
media_beta_C4_mov)^2;
suma_2_alfa_C3_ref = suma_2_alfa_C3_ref + (filtrada_C3_alfa_ref(i)-
media_alfa_C3_ref)^2;
suma_2_alfa_C4_ref = suma_2_alfa_C4_ref + (filtrada_C4_alfa_ref(i)-
media_alfa_C4_ref)^2;
suma_2_alfa_C3_mov = suma_2_alfa_C3_mov + (filtrada_C3_alfa_mov(i)-
media_alfa_C3_mov)^2;
suma_2_alfa_C4_mov = suma_2_alfa_C4_mov + (filtrada_C4_alfa_mov(i)-
media_alfa_C4_mov)^2;
                    end
            desviacion_beta_C3_ref = sqrt(double((suma_2_beta_C3_ref/(L-1))));
            desviacion_beta_C4_ref = sqrt(double((suma_2_beta_C4_ref/(L-1))));
            desviacion_beta_C3_mov = sqrt(double((suma_2_beta_C3_mov/(L-1))));
            desviacion_beta_C4_mov = sqrt(double((suma_2_beta_C4_mov/(L-1))));
            desviacion_alfa_C3_ref = sqrt(double((suma_2_alfa_C3_ref/(L-1))));
            desviacion_alfa_C4_ref = sqrt(double((suma_2_alfa_C4_ref/(L-1))));
            desviacion_alfa_C3_mov = sqrt(double((suma_2_alfa_C3_mov/(L-1))));
            desviacion_alfa_C4_mov = sqrt(double((suma_2_alfa_C4_mov/(L-1))));
                    % A partir de la desviación típica se puede calcular la
varianza
                    varianza_beta_C3_ref=(desviacion_beta_C3_ref)^2;
                    varianza_beta_C4_ref=(desviacion_beta_C4_ref)^2;
                    varianza_beta_C3_mov=(desviacion_beta_C3_mov)^2;
                    varianza_beta_C4_mov=(desviacion_beta_C4_mov)^2;
                    varianza_alfa_C3_ref=(desviacion_beta_C3_ref)^2;
                    varianza_alfa_C4_ref=(desviacion_beta_C4_ref)^2;
                    varianza_alfa_C3_mov=(desviacion_beta_C3_mov)^2;
                    varianza_alfa_C4_mov=(desviacion_beta_C4_mov)^2;

                    % CARACTERÍSTICA 3: VALOR MÁXIMO Y MÍNIMO
                    max_beta_C3_ref=0; max_beta_C3_mov=0;
                    max_beta_C4_ref=0; max_beta_C4_mov=0;
                    max_alfa_C3_ref=0; max_alfa_C3_mov=0;
                    max_alfa_C4_ref=0; max_alfa_C4_mov=0;
                    min_beta_C3_ref=inf; min_beta_C3_mov=inf;
                    min_beta_C4_ref=inf; min_beta_C4_mov=inf;
                    min_alfa_C3_ref=inf; min_alfa_C3_mov=inf;
                    min_alfa_C4_ref=inf; min_alfa_C4_mov=inf;
                    for i=1:L
                        % Primero se calculan los mínimos de las ondas beta
                        if (filtrada_C3_beta_ref(i)<min_beta_C3_ref)
                            min_beta_C3_ref=filtrada_C3_beta_ref(i);
                        end
                        if (filtrada_C4_beta_ref(i)<min_beta_C4_ref)
                            min_beta_C4_ref=filtrada_C4_beta_ref(i);
                        end
                        if (filtrada_C3_beta_mov(i)<min_beta_C3_mov)
                            min_beta_C3_mov=filtrada_C3_beta_mov(i);
                        end
                        if (filtrada_C4_beta_mov(i)<min_beta_C4_mov)
                            min_beta_C4_mov=filtrada_C4_beta_mov(i);
                        end
                        %Se calculan los mínimos de las ondas alfa
                        if (filtrada_C3_alfa_ref(i)<min_alfa_C3_ref)
                            min_alfa_C3_ref=filtrada_C3_alfa_ref(i);
```

```
                    end
                    if (filtrada_C4_alfa_ref(i)<min_alfa_C4_ref)
                        min_alfa_C4_ref=filtrada_C4_alfa_ref(i);
                    end
                    if (filtrada_C3_alfa_mov(i)<min_alfa_C3_mov)
                        min_alfa_C3_mov=filtrada_C3_alfa_mov(i);
                    end
                    if (filtrada_C4_alfa_mov(i)<min_alfa_C4_mov)
                        min_alfa_C4_mov=filtrada_C4_alfa_mov(i);
                    end
                    % Se calculan los máximos de las ondas beta
                    if (filtrada_C3_beta_ref(i)>max_beta_C3_ref)
                        max_beta_C3_ref=filtrada_C3_beta_ref(i);
                    end
                    if (filtrada_C4_beta_ref(i)>max_beta_C4_ref)
                        max_beta_C4_ref=filtrada_C4_beta_ref(i);
                    end
                    if (filtrada_C3_beta_mov(i)>max_beta_C3_mov)
                        max_beta_C3_mov=filtrada_C3_beta_mov(i);
                    end
                    if (filtrada_C4_beta_mov(i)>max_beta_C4_mov)
                        max_beta_C4_mov=filtrada_C4_beta_mov(i);
                    end
                    % Se calculan los máximos de las ondas alfa
                    if (filtrada_C3_alfa_ref(i)>max_alfa_C3_ref)
                        max_alfa_C3_ref=filtrada_C3_alfa_ref(i);
                    end
                    if (filtrada_C4_alfa_ref(i)>max_alfa_C4_ref)
                        max_alfa_C4_ref=filtrada_C4_alfa_ref(i);
                    end
                    if (filtrada_C3_alfa_mov(i)>max_alfa_C3_mov)
                        max_alfa_C3_mov=filtrada_C3_alfa_mov(i);
                    end
                    if (filtrada_C4_alfa_mov(i)>max_alfa_C4_mov)
                        max_alfa_C4_mov=filtrada_C4_alfa_mov(i);
                    end
                end

                % CARACTERÍSTICA 4: PSD
                Fs = 250; % Frecuencia de muestreo (en Hz)
                T = 1/Fs; % Periodo de muestreo
                t = (0:511)*T; % Vector de tiempo
                nfft=1024; % El numero de puntos de la fft
                x_filtrada_C3_alfa_mov = filtrada_C3_alfa_mov;
                Y_filtrada_C3_alfa_mov = fft(x_filtrada_C3_alfa_mov,nfft);
% Tomar la FFT, y llenando con ceros, de manera que el largo de la FFT sea
nfft
                Y_filtrada_C3_alfa_mov = Y_filtrada_C3_alfa_mov(1:nfft/2);
% La FFT es simétrica, así que se tira la mitad
                my_filtrada_C3_alfa_mov = abs(Y_filtrada_C3_alfa_mov).^2; %
Tomar la potencia espectral,módulo al cuadrado de la FFT
                x_filtrada_C4_alfa_mov = filtrada_C4_alfa_mov;
                Y_filtrada_C4_alfa_mov = fft(x_filtrada_C4_alfa_mov,nfft);
                Y_filtrada_C4_alfa_mov = Y_filtrada_C4_alfa_mov(1:nfft/2);
                my_filtrada_C4_alfa_mov = abs(Y_filtrada_C4_alfa_mov).^2;
                x_filtrada_C3_alfa_ref = filtrada_C3_alfa_ref;
                Y_filtrada_C3_alfa_ref = fft(x_filtrada_C3_alfa_ref,nfft);
                Y_filtrada_C3_alfa_ref = Y_filtrada_C3_alfa_ref(1:nfft/2);
                my_filtrada_C3_alfa_ref = abs(Y_filtrada_C3_alfa_ref).^2;
                x_filtrada_C4_alfa_ref = filtrada_C4_alfa_ref;
                Y_filtrada_C4_alfa_ref = fft(x_filtrada_C4_alfa_ref,nfft);
                Y_filtrada_C4_alfa_ref = Y_filtrada_C4_alfa_ref(1:nfft/2);
```

```
                    my_filtrada_C4_alfa_ref = abs(Y_filtrada_C4_alfa_ref).^2;
                    x_filtrada_C3_beta_mov = filtrada_C3_beta_mov;
                    Y_filtrada_C3_beta_mov = fft(x_filtrada_C3_beta_mov,nfft);
                    Y_filtrada_C3_beta_mov = Y_filtrada_C3_beta_mov(1:nfft/2);
                    my_filtrada_C3_beta_mov = abs(Y_filtrada_C3_beta_mov).^2;
                    x_filtrada_C4_beta_mov = filtrada_C4_beta_mov;
                    Y_filtrada_C4_beta_mov = fft(x_filtrada_C4_beta_mov,nfft);
                    Y_filtrada_C4_beta_mov = Y_filtrada_C4_beta_mov(1:nfft/2);
                    my_filtrada_C4_beta_mov = abs(Y_filtrada_C4_beta_mov).^2;
                    x_filtrada_C3_beta_ref = filtrada_C3_beta_ref;
                    Y_filtrada_C3_beta_ref = fft(x_filtrada_C3_beta_ref,nfft);
                    Y_filtrada_C3_beta_ref = Y_filtrada_C3_beta_ref(1:nfft/2);
                    my_filtrada_C3_beta_ref = abs(Y_filtrada_C3_beta_ref).^2;
                    x_filtrada_C4_beta_ref = filtrada_C4_beta_ref;
                    Y_filtrada_C4_beta_ref = fft(x_filtrada_C4_beta_ref,nfft);
                    Y_filtrada_C4_beta_ref = Y_filtrada_C4_beta_ref(1:nfft/2);
                    my_filtrada_C4_beta_ref = abs(Y_filtrada_C4_beta_ref).^2;
                    f= (0:nfft/2-1)*Fs/nfft; % Construccion del vector de
frecuencias
                      % Interesa la media de la PSD
                      sumaPSD_alfa_C3_ref=0;
                      sumaPSD_alfa_C3_mov=0;
                      sumaPSD_alfa_C4_ref=0;
                      sumaPSD_alfa_C4_mov=0;
                      sumaPSD_beta_C3_ref=0;
                      sumaPSD_beta_C3_mov=0;
                      sumaPSD_beta_C4_ref=0;
                      sumaPSD_beta_C4_mov=0;
                      for i=1:512

sumaPSD_alfa_C3_ref=sumaPSD_alfa_C3_ref+my_filtrada_C3_alfa_mov(i);

sumaPSD_alfa_C3_mov=sumaPSD_alfa_C3_mov+my_filtrada_C3_alfa_ref(i);

sumaPSD_alfa_C4_ref=sumaPSD_alfa_C4_ref+my_filtrada_C4_alfa_mov(i);

sumaPSD_alfa_C4_mov=sumaPSD_alfa_C4_mov+my_filtrada_C4_alfa_ref(i);

sumaPSD_beta_C3_ref=sumaPSD_beta_C3_ref+my_filtrada_C3_beta_mov(i);

sumaPSD_beta_C3_mov=sumaPSD_beta_C3_mov+my_filtrada_C3_beta_ref(i);

sumaPSD_beta_C4_ref=sumaPSD_beta_C4_ref+my_filtrada_C4_beta_mov(i);

sumaPSD_beta_C4_mov=sumaPSD_beta_C4_mov+my_filtrada_C4_beta_ref(i);
                      end
                      Media_PDS_alfa_C3_ref=(1/512)*sumaPSD_alfa_C3_ref;
                      Media_PDS_alfa_C3_mov=(1/512)*sumaPSD_alfa_C3_mov;
                      Media_PDS_alfa_C4_ref=(1/512)*sumaPSD_alfa_C4_ref;
                      Media_PDS_alfa_C4_mov=(1/512)*sumaPSD_alfa_C4_mov;
                      Media_PDS_beta_C3_ref=(1/512)*sumaPSD_beta_C3_ref;
                      Media_PDS_beta_C3_mov=(1/512)*sumaPSD_beta_C3_mov;
                      Media_PDS_beta_C4_ref=(1/512)*sumaPSD_beta_C4_ref;
                      Media_PDS_beta_C4_mov=(1/512)*sumaPSD_beta_C4_mov;


% SE GENERA UNA MATRIZ CON LAS CARACTERÍSTICAS OBTENIDAS
% Las columnas serán las muestras
% Las filas serán las siguientes características:
% 1 -> Valor medio beta C3 ref
% 2 -> Valor medio beta C3 mov
% 3 -> Valor medio beta C4 ref
```

```
% 4 -> Valor medio beta C4 mov
% 5 -> Valor medio alfa C3 ref
% 6 -> Valor medio alfa C3 mov
% 7 -> Valor medio alfa C4 ref
% 8 -> Valor medio alfa C4 mov
% 9 -> Desviación típica beta C3 ref
% 10 -> Desviación típica beta C3 mov
% 11 -> Desviación típica beta C4 ref
% 12 -> Desviación típica beta C4 mov
% 13 -> Desviación típica alfa C3 ref
% 14 -> Desviación típica alfa C3 mov
% 15 -> Desviación típica alfa C4 ref
% 16 -> Desviación típica alfa C4 mov
% 17 -> Varianza beta C3 ref
% 18 -> Varianza beta C3 mov
% 19 -> Varianza beta C4 ref
% 20 -> Varianza beta C4 mov
% 21 -> Varianza alfa C3 ref
% 22 -> Varianza alfa C3 mov
% 23 -> Varianza alfa C4 ref
% 24 -> Varianza alfa C4 mov
% 25 -> Valor maximo beta C3 ref
% 26 -> Valor maximo beta C3 mov
% 27 -> Valor maximo beta C4 ref
% 28 -> Valor maximo beta C4 mov
% 29 -> Valor maximo alfa C3 ref
% 30 -> Valor maximo alfa C3 mov
% 31 -> Valor maximo alfa C4 ref
% 32 -> Valor maximo alfa C4 mov
% 33 -> Valor minimo beta C3 ref
% 34 -> Valor minimo beta C3 mov
% 35 -> Valor minimo beta C4 ref
% 36 -> Valor minimo beta C4 mov
% 37 -> Valor minimo alfa C3 ref
% 38 -> Valor minimo alfa C3 mov
% 39 -> Valor minimo alfa C4 ref
% 40 -> Valor minimo alfa C4 mov
% 41 -> Media PSD beta C3 ref
% 42 -> Media PSD beta C3 mov
% 43 -> Media PSD beta C4 ref
% 44 -> Media PSD beta C4 mov
% 45 -> Media PSD alfa C3 ref
% 46 -> Media PSD alfa C3 mov
% 47 -> Media PSD alfa C4 ref
% 48 -> Media PSD alfa C4 mov
                    Entrada(1,evento) = media_beta_C3_ref;
                    Entrada(2,evento) = media_beta_C3_mov;
                    Entrada(3,evento) = media_beta_C4_ref;
                    Entrada(4,evento) = media_beta_C4_mov;
                    Entrada(5,evento) = media_alfa_C3_ref;
                    Entrada(6,evento) = media_alfa_C3_mov;
                    Entrada(7,evento) = media_alfa_C4_ref;
                    Entrada(8,evento) = media_alfa_C4_mov;
                    Entrada(9,evento) = desviacion_beta_C3_ref;
                    Entrada(10,evento) = desviacion_beta_C3_mov;
                    Entrada(11,evento) = desviacion_beta_C4_ref;
                    Entrada(12,evento) = desviacion_beta_C4_mov;
                    Entrada(13,evento) = desviacion_alfa_C3_ref;
                    Entrada(14,evento) = desviacion_alfa_C3_mov;
                    Entrada(15,evento) = desviacion_alfa_C4_ref;
                    Entrada(16,evento) = desviacion_alfa_C4_mov;
                    Entrada(17,evento) = varianza_beta_C3_ref;
```

```
                    Entrada(18,evento) = varianza_beta_C3_mov;
                    Entrada(19,evento) = varianza_beta_C4_ref;
                    Entrada(20,evento) = varianza_beta_C4_mov;
                    Entrada(21,evento) = varianza_alfa_C3_ref;
                    Entrada(22,evento) = varianza_alfa_C3_mov;
                    Entrada(23,evento) = varianza_alfa_C4_ref;
                    Entrada(24,evento) = varianza_alfa_C4_mov;
                    Entrada(25,evento) = max_beta_C3_ref;
                    Entrada(26,evento) = max_beta_C3_mov;
                    Entrada(27,evento) = max_beta_C4_ref;
                    Entrada(28,evento) = max_beta_C4_mov;
                    Entrada(29,evento) = max_alfa_C3_ref;
                    Entrada(30,evento) = max_alfa_C3_mov;
                    Entrada(31,evento) = max_alfa_C4_ref;
                    Entrada(32,evento) = max_alfa_C4_mov;
                    Entrada(33,evento) = min_beta_C3_ref;
                    Entrada(34,evento) = min_beta_C3_mov;
                    Entrada(35,evento) = min_beta_C4_ref;
                    Entrada(36,evento) = min_beta_C4_mov;
                    Entrada(37,evento) = min_alfa_C3_ref;
                    Entrada(38,evento) = min_alfa_C3_mov;
                    Entrada(39,evento) = min_alfa_C4_ref;
                    Entrada(40,evento) = min_alfa_C4_mov;
                    Entrada(41,evento) = Media_PDS_alfa_C3_ref;
                    Entrada(42,evento) = Media_PDS_alfa_C3_mov;
                    Entrada(43,evento) = Media_PDS_alfa_C4_ref;
                    Entrada(44,evento) = Media_PDS_alfa_C4_mov;
                    Entrada(45,evento) = Media_PDS_beta_C3_ref;
                    Entrada(46,evento) = Media_PDS_beta_C3_mov;
                    Entrada(47,evento) = Media_PDS_beta_C4_ref;
                    Entrada(48,evento) = Media_PDS_beta_C4_mov;
                    evento=evento+1;
                end

                end

            if h.ArtifactSelection(eventos_totales)==1
                if (h.EVENT.TYP(i)==770 || h.EVENT.TYP(i)==769)
                malos=malos+1;
                end
            end

            eventos_totales=eventos_totales+1;
        end
    end

%Código original de Nagore Peirotén López de Arbina
%Modificado por Laura Escobar Moralejo
```

8 ANEXO B

```
%Función en Matlab para la RNA creada con el Neural Network Toolbox

function [Y] = RNA_25neuronas_20190325(Entrada)
%MYNEURALNETWORKFUNCTION neural network simulation function.
%
% Generated by Neural Network Toolbox function genFunction, 25-Mar-2019
17:54:43.
%
% [Y] = myNeuralNetworkFunction(X,~,~) takes these arguments:
%
%   X = 1xTS cell, 1 inputs over TS timesteps
%   Each X{1,ts} = 48xQ matrix, input #1 at timestep ts.
%
% and returns:
%   Y = 1xTS cell of 1 outputs over TS timesteps.
%   Each Y{1,ts} = 1xQ matrix, output #1 at timestep ts.
%
% where Q is number of samples (or series) and TS is the number of timesteps.

%#ok<*RPMT0>

% ===== NEURAL NETWORK CONSTANTS =====

% Input 1
x1_step1.xoffset = [-0.0818175695530037;-0.0552415956268897;-
0.0942734498764987;-0.061598767874602;-0.312650339772504;-0.198218443292961;-
0.233278892326382;-
0.232531624383889;2.26811133072582;2.25813681141491;2.26652021552636;2.549406
77812089;2.72997904807761;2.20593579875071;3.04626516749872;3.36810648301038;
5.14432900856686;5.0991818590671;5.13711388738966;6.49947492032874;5.14432900
856686;5.0991818590671;5.13711388738966;6.49947492032874;6.9923351479952;5.60
559208398222;6.92870250633097;7.10445147725676;8.34032297614891;5.92039454223
531;9.10952563529897;9.51082972561431;-25.3978747499242;-24.1196624639093;-
22.8619901929341;-25.4433200083209;-41.3269787610635;-47.0034951437282;-
42.9846096086719;-
41.5250336814673;2486.65296790038;3808.47205825032;5811.28823578817;4743.8299
2067912;2605.69052968893;2633.21742605111;3321.25442842225;2627.41250522143];
x1_step1.gain =
[12.0896904626107;10.2339793580181;10.9801437503688;10.3135279402683;3.536538
37608145;5.08031093822466;4.45343874429849;4.4315967341155;0.498284744205615;
0.538767117592119;0.562973664779512;0.462192521898721;0.221151365621027;0.145
574072250095;0.180132953585558;0.184591544749699;0.0582789723661796;0.0654761
155997129;0.0696266638343951;0.0490337154784624;0.0582789723661796;0.06547611
55997129;0.0696266638343951;0.0490337154784624;0.0832081301108956;0.107088973
107561;0.101741145313982;0.128218594996116;0.0616568413680404;0.0463581141228
539;0.059716390645516;0.0596896309942948;0.105918453299289;0.108823863207798;
0.130093961014958;0.109672689259035;0.0589968816324119;0.0496388841436342;0.0
601574563814692;0.061546350497494 6;1.56954144077558e-05;2.98396346228232e-
05;2.05555876606113e-05;2.04980530990414e-
05;0.000128073654062032;0.000114067406736267;9.59164099036424e-
05;0.000136266870310085];
x1_step1.ymin = -1;

% Layer 1
b1 = [1.5074660112114797;1.431291565442218;-
1.1878511764471855;1.1881398026834054;-1.0550970930018038;-
0.95509586309968075;-0.87882329694453276;-0.6743281548920762;-
```

```
0.60238723520637505;0.33045617338916222;0.29707364030764161;-
0.083108661633796113;-0.026573583257370803;-
0.18773223433868666;0.27768515566017066;-0.51145821395656299;-
0.49412766593797069;0.66947745214385679;-0.74993928257004638;-
0.81749789504562942;1.0640926520477563;1.1502524877333686;-
1.0774484389981764;1.4198262912615349;-1.3393481220744647];
IW1_1 = [-0.085441327720594401 0.00858752598437546 -0.21975596171421569
0.013959963082728462 0.3131571083894274 0.14149523426579744
0.11273086510401124 0.023750554169384184 0.31356838618669969 -
0.24576845252996865 0.32591647827630243 -0.10628192554642948 -
0.22606078144471628 0.2454146723259614 0.16202337892602492 -
0.091269778073441998 -0.25047355257623294 0.020305033761134496
0.11155674373173417 0.30761471140605695 0.19355984850680583
0.022855735132683075 0.31588202007408256 0.22210432857571633 -
0.20449563857536443 -0.22919664671623066 -0.08931415793676703
0.20896292000659128 -0.2873154238145465 0.28036223222559725 -
0.014036301613973935 -0.14467346509492549 -0.061898183774802683 -
0.34550898228991639 -0.30809290606894035 0.35536168998114087
0.12206583970755755 0.24792531872438009 0.23449571964024785 0.329780138494075
-0.23578806831960145 0.15759679994472808 -0.2088388417614544 -
0.25189124039537669 -0.23595236069470857 0.26677096739330675 -
0.15212085772777217 -0.049606983773411127;-0.25022814304748581 -
0.19835693836493751 -0.059864685582224644 0.10786453711439878
0.28058170973241742 0.16911729959139693 -0.19092675437095413 -
0.16635654724389862 0.015715847426168147 0.067134627120138776
0.42889916478057183 0.2788940543939844 0.15570157613397617
0.09974099293143808 0.36827143704705767 -0.0415705511490191
0.34103833569930087 -0.34905429625717793 -0.077427746561967645 -
0.30890421042322874 0.18557783305074829 -0.28556923562418113 -
0.015149007541267991 0.23987837346114971 0.073042147374097227
0.10795759763705991 -0.055709353935865576 0.041788003515255752
0.069238856467546323 -0.33166321469498916 -0.097932161420528796
0.37396921611060885 0.27566851701152789 -0.1863344174732455
0.16785793047119557 -0.058563843646406197 -0.013900357084036464 -
0.055697030641403529 -0.12435126889872117 -0.1013608560147444
0.24201739311244447 0.29740579237177545 0.085883487804227984
0.33644100377713732 0.40720159325471694 -0.18759182582333256 -
0.26909764594174324 0.099109520246954297;0.23241094567267545 -
0.047485443650276273 0.088455054084256274 -0.17067851356586569 -
0.32819298253731555 0.13579554966822818 -0.2187311454866154
0.16435098125856798 0.027470330265401917 -0.0041178334671954711 -
0.3794115205999809 -0.0019655973715862612 0.076999155324450105 -
0.33902291842627091 0.3657438653910029 0.17714310387777488 -
0.29586415376268144 -0.038340315370871 0.19022392120815404
0.19766894059552703 -0.088149583050998015 0.13515151783196525
0.089660173138068344 0.08038948249898499 -0.082358644154244071 -
0.027312429822600633 0.13711405507336299 -0.16174132861940799 -
0.42630748004681773 0.036638060338657789 0.31485416736869776 -
0.24857665389773725 0.48216537272996435 0.040548100467483722
0.10807402058022532 -0.39664654682353506 0.38481029611851619 -
0.27594423177944494 0.32485604674034452 0.13157772666201276
0.26586342135380703 -0.26268327650431345 -0.12642403659207224
0.32922907160001347 0.11076108078550161 -0.098143565074375466 -
0.30504088983314348 -0.12144620435921301;-0.030035158900043536 -
0.14178361716838969 0.35586939787085625 0.05027676247318795
0.15332396110199153 -0.23681295706002889 -0.07002063159706727
0.02528785317314881 0.23809157471642328 -0.13573193347220536
0.35128453748626787 0.039548152027019133 0.099142652403815604
0.054352612368032772 -0.022220568603235742 -0.067464712364790805
0.29862193312958168 0.047204685134388705 -0.028931731244491471 -
0.024324990205964625 -0.083967357931838016 -0.37624566087876632
0.22042296394085450 0.28309726235809424 -0.22160060210248250 -
```

```
0.24711487022964876 0.34518853339354311 0.21617742713398899
0.34888045138867024 0.28877814730004708 -0.013425235880859428
0.30373950086313395 0.30662246334850901 -0.191754423516866 0.2937860398114483
-0.09069623264159829 0.40753658950006688 -0.0027850729764871173 -
0.095471297031221564 -0.21914840732818047 -0.23376553196111829 -
0.0037296504277213633 -0.30131979206201004 -0.26443037227697236 -
0.17981393870878726 -0.14907364269538378 -0.17849597144837434
0.004386215629568575;0.030485968793019104 0.31589818282372828 -
0.33869525474674522 -0.29091810807794799 -0.28167192232827226 -
0.043280659767809461 -0.088286037549013444 -0.04567473234594141
0.18044626760110774 -0.14921323436340128 -0.34152629377126636
0.22692492147618343 0.1959715262303556 -0.30311443184888343
0.058563066372222243 0.29896436678215044 -0.07175939195552565
0.01204345621947002 -0.28258088353755101 0.21505125223951785
0.13204093703227604 0.026130770190540612 -0.15556933217242602 -
0.31522344651189727 0.21324052524526965 0.26921807824026639 -
0.18906342737525572 0.13282214451065949 -0.29881531396923589
0.094695368391669255 -0.21281792594059909 0.20458463768484744 -
0.12126976145064142 0.1278648955861372 0.05436576681271689
0.31423675003606666 0.13210706465242716 -0.2607405374318586 -
0.063294844100364789 -0.17982109925088666 -0.14915267363961088 -
0.26169972246330137 0.071300218087443273 -0.30752040644269635 -
0.1886060422702848 0.024535343751719459 -0.11549909589443488
0.31513615028981545;0.40401264734582981 -0.12601265650855695
0.33798800998834583 -0.1332975347198683 0.21714349663931848 -
0.2505447739450602 0.1990301252866544 -0.13405266371982127 -
0.24879132796237965 0.16770276991542132 -0.12869916451221644
0.4000741116763083 -0.17620842239869419 -0.32253951869805264
0.13643116876189418 0.085544973808063932 0.31061715985267069 -
0.33578051290335392 0.24765215791877387 0.1461933867559983
0.15904550443213228 0.00083660293851661044 0.041773128643754681 -
0.20178690501212668 0.16311405714812877 0.055969202479893294 -
0.30347281474994592 -0.038050098256175481 0.18377756638027554
0.3304576400896872 0.26357138772929939 -0.13450502475839776
0.43869026866502792 0.41503052755994407 -0.10079464923311972 -
0.17787178595919009 0.14620836150104652 -0.0475013869534072 -
0.035552235481066688 0.12425427588736802 -0.18624474464866683 -
0.22886104457267295 0.17091305965886383 -0.00075069508565460755
0.10570591168844638 0.46629054504553452 0.37933384306636941
0.14692972507283503;0.27746204757015569 -0.10727018807511679
0.25610411052546705 -0.091783974374783306 0.13727833396109523
0.36239650466137707 -0.03437312203347704 -0.24365348291155725
0.029095608385537944 -0.27392632862116628 -0.29128834763229267
0.40700621692633582 -0.34708486550567763 -0.009779723158934283
0.14339224223412475 0.28047316469225619 -0.29396651181489208 -
0.022291546362593232 -0.048235655676759623 0.40872822467801234
0.3066407162865048 0.20313163624306582 0.29956830508703486 -
0.20503614014405933 -0.18195143052438179 -0.22186723274020359
0.21205399965132585 0.23437343088767773 -0.080244905750911089
0.20133617504986773 0.12598042745845744 0.17206937285439672
0.24560844919638308 0.097214694262166335 0.14206315712168899
0.12047674784954387 0.3385935129813688 0.36112679294954786 0.3276940299382568
0.047527519305834987 -0.12440874436822735 -0.28521859269122263 -
0.26644345217458759 -0.088986450945540904 -0.17728387858921307
0.26765058071769765 0.087292103580485947 -
0.23317767954397978;0.060385194480233592 -0.13776010136420916
0.34915236857829829 0.30651336442713328 0.28779936629648545
0.12626875202938678 -0.25308450748345057 -0.15941262925454494
0.0094568037077747656 -0.014498458962829799 -0.22479375294013892 -
0.1720566445425652 0.026483694072069019 0.29037526692512039
0.055055052486447174 0.1434996532306512 -0.31575236257643319 -
0.026457069904475567 0.40302687735968473 0.17699517982180507 -
```

```
0.05433497520770126 -0.32199206625866639 0.087914080974783385 -
0.24695139468641866 -0.34379482314786008 -0.10030243123449642
0.23251886930634683 0.21153942047523044 -0.022912202548806319
0.097950857224250037 -0.19144970928536678 0.086447684593910709 -
0.16690900969720432 0.3744430325670236 -0.27832218886435861 -
0.055850408504047246 0.12423592535934445 0.31998584230441912 -
0.0021936034639171795 -0.21289394987581081 0.23828154091562584 -
0.21215011368932762 -0.090073571262646548 0.16958819527082702 -
0.37368010145483033 -0.30329164725379981 0.10931853200045868 -
0.16987373472354303;-0.048362473355460595 -0.23909591156664542 -
0.033638283014274406 0.10440832316265786 0.18958320674156726
0.059379357231508723 -0.3094961948617142 0.063994493369922772 -
0.30186951074943447 -0.29062247655700157 -0.16961289848805136 -
0.35294895288228451 0.090905091023814111 -0.30027171764025973 -
0.12475371475883626 -0.40189047056369598 0.20606827527741955 -
0.24160254505369327 0.049060619705715632 0.29892015132175648 -
0.20533918198486245 -0.029218789285713663 0.20808822271166891 -
0.27791591107136365 0.1864988356935196 0.18278276169024471 0.4163958995411981
-0.0053510041989818583 -0.26641014468886864 0.43549838660828716
0.1727919010421044 0.28661975685881008 0.033369203304437225 -
0.20028502296367073 -0.24134450286036646 0.32279264808004887
0.10555354856566224 0.22711403646114284 -0.033206281208701977
0.25647234032603022 -0.02309445346923189 -0.17616513937530623 -
0.19105671248368167 -0.21277066602696862 0.25756743059547743
0.1109700067175648 -0.21914283197050946 -0.11572283081443631;-
0.2810778953762384 0.049524359776409357 -0.15920896781430524
0.085728880447606626 -0.26831504970888281 0.13851172078346757
0.2000624797135373 -0.076282892158910404 0.18355875571826175 -
0.047458678357261544 -0.17720534450550238 -0.078914335934697225
0.30167732292402055 -0.11527412869527319 0.18943006641625706 -
0.093335718771372589 0.35990126215948981 0.07748803192070601
0.18993558561979584 0.14182790356064662 -0.07470959220789701 -
0.18892195700972039 -0.26179203538475404 0.31987685289770529
0.23244360706216702 0.31448111371259735 0.052646115870709789
0.23116843799943615 -0.19985198151842995 0.12841619191703843
0.19885661888425563 0.35085314193965234 0.38971872940343116
0.21574793556225999 0.17994470981801755 -0.25330599672410808 -
0.26332229050446954 -0.11406967959978027 0.28288332839948171 -
0.31552193701903414 0.057246869484583865 0.12995929960847072 -
0.26828372468140627 -0.29676082148165267 0.17043904837230833 -
0.070016627811327725 -0.26412098093359893 -0.23533450736076766;-
0.29378304530352095 0.1991428174826485 -0.28937122293223677 -
0.24449356046126344 0.11341251443311712 0.27171419116426887 -
0.028387965101181217 0.1133922223975477 0.15021909400377739
0.12861160959240037 0.077688144182611274 -0.22133017600544005 -
0.31983991957221525 0.18364447127366063 -0.041213099585280667
0.14609727655156643 -0.14532665309248585 -0.077990182695975505 -
0.046692244954242304 0.28683818110554637 -0.21372195868524355
0.10752214086884754 -0.20877279494131215 -0.36026274173035483
0.017583489997143497 0.16496666174311475 -0.20812952170327734
0.39308400697203771 -0.0509370959041947 -0.36473036531696612 -
0.1770262950673249 0.27602558501342556 0.19770573867039132
0.30200601840596225 -0.22073468540553026 -0.042126010876154106
0.22967131410830555 -0.063336805198493967 -0.2044927127929379
0.33308338786217673 -0.16581550266456771 -0.018725520371332433
0.31837820741206879 -0.25182754573100075 0.23682158418744964
0.078762087136708153 0.35878342337784336 0.352911144068844;-
0.20829636524498418 0.20456344140873239 -0.35022921134738483
0.12308830664456386 -0.0024853261737888092 0.038068066105220622 -
0.25794883140830788 0.24630900015117174 0.15383732038024314 -
0.16477701500157699 0.080712632415696234 -0.080806683140743601
0.004662677961049101 0.32592271048607513 -0.34360510281403178
```

```
0.14079215682676621 0.31863584920433102 -0.23734921612215304
0.34891106660262589 0.01680026483647672 -0.096977997818994965 -
0.37800595711217505 0.43732725801643396 0.21525817152618484
0.24122308628238937 0.28232864122241641 0.21547189972131478 -
0.0047253809368082746 0.1586014359001536 0.27104539427006108
0.34646093922596721 0.3454082966333939 0.11969741957305935 -
0.34185085906315504 -0.13955929725562113 -0.10971153922552941 -
0.29838248634700493 0.16865030050231813 0.0026670940862152451 -
0.068082675636247481 0.17848064400663397 0.1179079289559488 -
0.13380395706988929 -0.31198848561846798 -0.041622100702359284 -
0.10008420803841463 0.16597489734116957
0.0093303873234646291;0.075286015620348229 -0.28953324267682656 -
0.27188655350706475 0.031832676914073835 0.15757928477738622 -
0.054719079438435576 -0.0013759967003210611 0.35719300838436341
0.37862096295123215 -0.2430473252502855 -0.10346521668771065 -
0.021963109130211812 -0.2428671109810536 -0.037587308449626496 -
0.38053349325071456 -0.13414981659478059 0.40485622477477251
0.11273424347106395 0.19705214636053286 -0.22741685101245618
0.15021966205161449 -0.070224206959048602 0.022060837896598642
0.18430386751706423 0.11487395689014371 0.31635488613607132 -
0.041568048355359627 -0.011695750021566028 -0.3039854761647342 -
0.017327501396219136 0.021352413018449215 0.23834277802435769
0.17654195548307666 -0.14085988475306485 0.17332775742592371 -
0.28460775885744011 0.38117668625524798 0.12981902063169581 -
0.091657744950087383 -0.12063005067692925 0.022079194612709727 -
0.062616751791900752 -0.31920973429014793 -0.38120112976336956 -
0.40417746940401933 0.36598946321932896 -0.095545770539220212 -
0.26304162356120636;-0.23239813451081398 0.24894833785821913
0.0096430638725053773 -0.39893428128642205 0.11973418090380299
0.36704986480012169 0.31797644083489196 0.085083239532184204 -
0.27841257798625701 -0.31299232797331605 -0.17255217870996639 -
0.34387073250455552 0.44611957388823964 -0.20989312951291583 -
0.13599551632466547 -0.022743061404315409 -0.27388629423667676
0.20688867310658196 0.18030824477335819 0.061805194233523611 -
0.087913336577485388 -0.035566284547958911 0.502302262478259 38 -
0.16974966575331024 -0.43542100114338445 0.032855009140809 07 -
0.284056226635232 0.028328600456914595 -0.130825248176110 55 -
0.093874468291259339 0.15706287541649172 -0.404777473184178 17
0.29121431807154174 0.016752646248537714 -0.171480690668683 43
0.045974071255937429 0.21584659745877818 0.432203390334119 09
0.090957433241443647 0.43410835344392784 -0.232778373726988 56
0.11413307967410531 -0.40252787888660335 0.12291450542091 29
0.083261494361953389 0.38091979850881769 -0.0066818427129826481
0.26807437890654184;0.39781310090692967 0.31571058727641854
0.43512104069262386 -0.27877759603704239 -0.045646021787189781 -
0.075871526929079669 0.11638271071271972 0.15467215151885727 -
0.015558598788443291 0.33467543403014088 -0.13508027058170011
0.4303723746529024 0.15238869505312883 -0.053000857918982319 -
0.18292762579014418 -0.068256034645403835 -0.199044361932364 94
0.26611019193728191 0.017612584732812381 -0.080522912175170061
0.3412623730739886 -0.092310550161643634 0.453533221859420 47
0.0094607043145713554 0.25333795639886703 0.038958750359674235 -
0.069158387162016874 -0.2543923846396776 7 0.038798891927706317
0.046635716332358759 -0.13280792152236509 0.115582701182723 64 -
0.108729261118512 -0.095238754252635491 -0.335626255679058 68 -
0.45177094187662553 0.20872814477249968 0.348684464752617 55
0.26054000317420634 0.20408053473441273 -0.2043038220820757 -
0.038831627807056839 -0.20842645745199698 0.2106795292379875
0.098507273475959931 0.13716437896304878 0.355988791761528 42
0.33548502330805519;-0.43060602543339815 0.061329287785714635
0.11124934124961189 0.16706042228357884 -0.18189548165445338
0.011558611732247766 -0.090147241940233755 0.192898219494975 49 -
```

```
0.33886801945889117 -0.405148925287588 -0.045126331869274713
0.2818816978140774 0.14722283211191145 0.27513705702726093 -
0.23679789958578792 -0.039224652096782081 -0.019366161560277824 -
0.39659729712080444 0.44488675326680849 -0.25123380080161289 -
0.19597407387689783 -0.093435670300470272 0.20329243806160252
0.37872896938483458 -0.13354984977325046 -0.22644682260677326 -
0.20729997005800307 0.19210701484704665 -0.22697895688182446 -
0.019205614731813848 -0.26011870272719995 0.094616974677268359 -
0.12108002592171045 0.18596037824259071 -0.062479833419137668 -
0.40342105705933368 0.11284496811542336 -0.052372148867126593
0.32187408148156627 -0.34127005308471514 0.33259447293064531 -
0.00039336787296842101 0.085442199024453658 0.034178794662922007 -
0.31925124275700623 0.45061318615152529 -0.11067293251974349
0.12101336808173935;-0.090778934622351801 0.043375290317838128 -
0.14202123357303501 0.25080150389651812 0.30201721588937397 -
0.15436221681392356 -0.072601974353656729 -0.21584028285427814 -
0.38977632794664857 -0.28432729346900204 -0.010419887453165223
0.0021906673527028885 0.16523310570973515 0.21751874213426325
0.34650265609330855 0.2664210436701393 0.20781110035635786
0.31924783978063187 -0.2413826757271329 0.053541331326283 47 -
0.084368646059636121 0.055761942300258216 0.062589998490416479 -
0.38796655243163169 0.25978735030508077 -0.059149322864293065 -
0.36138410678148081 0.30544164810565611 0.32200285353942304 -
0.11136880136423845 0.087327510578600873 0.05817140503430622
0.34430768800508976 -0.034588069708191498 0.15168350532856653
0.014403697113884098 -0.28397594827582867 -0.078145183132309642
0.1269716464480343 0.1125482176131829 -0.34411100677791484
0.062856356358798396 0.10630466498849767 -0.029689670579865254 -
0.24321291577229265 -0.34605628850684828 -0.19274887447467587
0.27086702239568261;0.15675880435123837 0.10935696132918758 -
0.29990677152626566 -0.33080335215658219 0.069640224231796957
0.038565134093712185 0.22985481697964025 -0.24353119169763085 -
0.018276335984165226 0.30907861179365598 0.058836102257776676
0.19872405169064536 -0.19208572463674095 0.076640346056205969
0.13940995341640972 -0.19027260726073522 0.25699822245114573
0.26487642365717734 -0.25652614084055059 -0.33029296435482675 -
0.17726150518039926 -0.22839389096074431 -0.33639303659496778 -
0.37056210151219909 -0.26043235450167818 -0.040038199066786614 -
0.1418901166397131 -0.090649553569220678 0.21582120418036962
0.16465793883690735 -0.18427734858097597 -0.33297035819769721 -
0.17768252790370156 -0.2681790512936657 0.013464745848879231 -
0.05891942775457789 -0.30007304870550378 0.095585576866263539
0.35899631615731514 -0.28676645930765199 0.31979280698371532
0.10219601707321281 -0.2598402888349724 -0.14603318496436177
0.14652498566621874 0.1019734914195202 -0.38589881760903971 -
0.28520249940521319;-0.32073934103172158 -0.12868842045494425
0.26564634852947711 -0.06444709616136611 0.37062914118499068
0.064792795442019602 0.16930856857549606 0.13517313797065786 -
0.23842625970928949 0.066243795873667966 0.066129328588108591
0.35577612306615258 0.27347154372178956 -0.21707013023639324
0.24766102373626891 -0.3363742977579845 0.21777242079593762
0.33091616927636236 -0.037843616196197187 0.111018315481620 08
0.11941954849155062 0.33547350580957697 0.1189970634012773
0.28391767670939283 -0.34298811234846915 0.14909337359449126
0.15140982999482733 -0.15204166364014832 -0.10216911908026453 -
0.25169418961454798 0.30663272385960494 -0.12141125924388493
0.012754554423104828 -0.23214270946184248 0.11822957789524421 -
0.22226512571902796 0.29266107550351161 0.15629620255369692 -
0.31942230487442891 -0.24968233062492237 0.170041440464714 95 -
0.048817963502927379 0.26435111817600049 -0.33961279700502106
0.2897516302013089 -0.020289946787306272 0.31221126094177726
0.27710236162835417;-0.12982295991603979 0.21757102618527691
```

```
0.37493738426355067 0.21233207470689205 0.19204420917655896
0.0006598444027737134 -0.21669775892798948 -0.41359747450259332 -
0.092615764012985907 -0.39622551988371107 0.11823142580605157 -
0.55470957932357745 -0.033997300554882745 0.13773885216553544
0.043061373220539066 -0.21131024994737843 0.18900697581879822 -
0.0085466641262865496 -0.15397977592251069 0.17783519191221386 -
0.10697422275931566 -0.1100739730285314 0.058137045762123384 -
0.30968428977140827 -0.29290050192589895 -0.35498937947384435 -
0.26793609672599844 0.1374764400759389 0.25773856567623604 -
0.16905440575840464 -0.10302807848343447 -0.069032441960111934
0.14126003197803025 -0.34345918881998505 -0.32916357670126783
0.26255581483850127 -0.10544917272398804 -0.35358095714298382 -
0.18637455223138932 -0.12208959191654048 0.33330925943081785 -
0.17812490962326685 -0.1768019611108243 -0.16675708822647267
0.22507841197560022 0.19614548435192297 -0.46752392172043417 -
0.25141369334909697;0.25575712859798116 -0.30034797467950924
0.042707272920589116 0.299528256906544 -0.055498929404672888 -
0.092777871012962004 -0.20185165626944748 -0.23505546964361873 -
0.13019177469844856 0.173078418858717 97 0.21862661915080783 -
0.29298636902952258 -0.15023888234782803 -0.22119504980620922
0.1622722249490306 0.20279521952137378 -0.30422808251147054 -
0.24772625920752447 -0.25362894168042849 0.035792175147819218 -
0.016784611097666088 0.25941069641868453 0.22355607694262955 -
0.17967738691417801 0.27807005396121603 0.090005460425262557
0.15999408315757785 0.36997138707663069 0.19056683775993263 -
0.16974539029814131 -0.20764280860236664 -0.1536272966644199 -
0.20363464095069292 0.32404794072931281 0.019172133260261952 -
0.30581134432723223 0.12936969825518208 0.25079523138959969 -
0.079366026010176824 0.24112665877779341 -0.26446005146571744
0.22747790634230672 0.29276310259057398 -0.33763695598274335
0.20525912255640086 0.082459415227556981 -0.070662357671462894 -
0.26625435270935949;0.25725701120153133 0.024150479219140243 -
0.29692209137423403 0.33527157133974583 0.054830041632631429
0.15436919105183855 -0.30329521219642891 0.18311337330328739 -
0.063080020706435647 0.27177960325175321 0.2533617793959837
0.2543351121956548 -0.11933487187009444 -0.28981364203282495 -
0.12111587744615715 -0.0059655462059392851 -0.31541710461364325 -
0.031516125042639863 -0.27996120580215889 -0.012641458844419133 -
0.1073039273066939 -0.1779386529452304 0.097135662244219795 -
0.3849853571998611 -0.28352658595168062 0.043790100990982496 -
0.056537653195373822 0.13775355962487654 0.10291410656559599 -
0.25913463337533305 -0.10497382041360427 -0.39596832954367062
0.38943610291305436 -0.20274232297430081 -0.0046855452496835292
0.24350596736657057 0.040879058506369734 -0.060656778397203362 -
0.16825300757905368 -0.14601011401202929 -0.32198042001585642 -
0.07940314823071562 -0.29213877250878434 -0.2570281138409875
0.13113206909871206 0.29497900249014869 0.28194079114422738
0.22300993015361181;0.069800086635462819 0.353614530517413 87
0.25212181746682127 0.0099209716700678838 -0.38178700725107462 -
0.28940511129909108 0.24864910671137055 0.09733691594109263
0.15424999831819125 0.21971971475128763 -0.16426639637981039
0.34470010243096411 -0.21608812956555271 -0.1786033385791041
0.083523563701467338 -0.17599802215978172 -0.18911752608391247 -
0.012815418887259636 -0.1567657286467658 -0.10569262712567046
0.058769841293957965 -0.27813799553230079 0.20486742966743371 -
0.10981847370564324 -0.44286846278380643 0.40791801308306858 -
0.2048109801118973 -0.12444763949535813 0.28627437087138963 -
0.32910813818347673 0.38450963282596368 -0.15729683373940034
0.2211707609901144 0.30925037036403163 -0.30738649827729758
0.09252664585366091 0.12718598937377507 0.49579665771110148 -
0.19276344845132096 0.43110167786061171 -0.47632930289137376 -
0.34506831895135437 -0.21270247007844578 -0.24320237150404128 -
```

```
0.28714366579012002 0.26954989499978887 -0.24543726949635697 -
0.18254880983319474;0.19790146505443768 -0.30366560057406128
0.28292133510888734 -0.05305526494138809 0.17031203006814205 -
0.23074140001217139 -0.21552267319059978 0.091436917186208541
0.033154115059779443 0.23772313802612693 0.2956528884551497 -
0.33826271272558633 -0.257714148574272 -0.064948557430508327
0.15204099628050194 -0.12691061775622084 0.21592805557534298
0.22725065805331701 0.27384611652592539 0.11579537658535369 -
0.22311747482744282 -0.19312254902675424 0.13248418505854601 -
0.18925299852467678 -0.14199824881215065 0.2582394544623271
0.18697504991146396 0.19737348416820386 -0.16681095674897728 -
0.0081038746467758643 -0.16117120837952276 0.14488138749324728 -
0.31913264514684647 0.30945527824095431 -0.0027933379016167816 -
0.1478647910869027 -0.25785922381679782 0.16065968281481935 -
0.3632570005387552 -0.28192557180667493 -0.37515035986243217 -
0.27435222407828552 0.041002610902169334 -0.22578087820368181 -
0.068553971381677528 -0.017280203647212902 0.25101398332649688 -
0.060852241984983416;-0.22072703001535302 -0.1962305856248675 -
0.19780747947665403 -0.11577299003746599 0.29902801810749879 -
0.12401806205232614 0.1249064649372663 0.39252802529542508
0.17986270092926529 -0.17251997326151577 0.29783297263238678 -
0.26772231704654853 0.27768251297201324 -0.23711202631317127
0.21691619503604664 0.083789952723786476 -0.10103930768519855
0.034046861243110828 -0.099673928441516471 -0.12844673444913929
0.24780610628775415 -0.092094533216355356 -0.29801103551826191
0.20892140510988894 -0.27595797033852815 0.17026332184795148 -
0.3567092676054136 -0.057432333428565845 0.11487257993056683 -
0.30046298713509506 0.27840242638085655 -0.31597090453208038 -
0.051633892433354256 0.26399113912539968 0.11845006073653339
0.26390196373990171 0.2686852919112126 0.34578004098576104
0.30662520986819153 0.2215311050302263 -0.27487952116420933 -
0.29318515960861097 -0.26889495250188566 0.26797232814489652
0.19145555245303217 -0.011613597137974031 -0.32720006221876502 -
0.20687780756420648];

% Layer 2
b2 = -0.026741102075264728;
LW2_1 = [-0.15664043151832904 -0.64061554121364073 0.53267584344069541 -
0.50269391534681296 0.21175773693295261 0.89270548165249697
0.62567376597077085 0.16623676965490364 -0.49926393123410234
0.1615799765954723 0.33656760668096808 0.54866799134186761
0.35839404850466972 1.0267043382528496 0.7751186157183243 0.78132923913283558
-0.004185069098883127 -0.46403033291189771 -0.54369599915925926 -
0.83994463208465109 0.5046468770289857 0.63784993403613943 1.0120070730834836
-0.28047247647538043 0.25174776287402756];

% ===== SIMULATION ========

% Format Input Arguments
isCellX = iscell(Entrada);
if ~isCellX
    Entrada = {Entrada};
end

% Dimensions
TS = size(Entrada,2); % timesteps
if ~isempty(Entrada)
    Q = size(Entrada{1},2); % samples/series
else
    Q = 0;
end
```

```
% Allocate Outputs
Y = cell(1,TS);

% Time loop
for ts=1:TS

    % Input 1
    Xp1 = mapminmax_apply(Entrada{1,ts},x1_step1);

    % Layer 1
    a1 = tansig_apply(repmat(b1,1,Q) + IW1_1*Xp1);

    % Layer 2
    a2 = logsig_apply(repmat(b2,1,Q) + LW2_1*a1);

    % Output 1
    Y{1,ts} = a2;
end

% Final Delay States
Xf = cell(1,0);
Af = cell(2,0);

% Format Output Arguments
if ~isCellX
    Y = cell2mat(Y);
end
end

% ===== MODULE FUNCTIONS ========

% Map Minimum and Maximum Input Processing Function
function y = mapminmax_apply(x,settings)
y = bsxfun(@minus,x,settings.xoffset);
y = bsxfun(@times,y,settings.gain);
y = bsxfun(@plus,y,settings.ymin);
end

% Sigmoid Positive Transfer Function
function a = logsig_apply(n,~)
a = 1 ./ (1 + exp(-n));
end

% Sigmoid Symmetric Transfer Function
function a = tansig_apply(n,~)
a = 2 ./ (1 + exp(-2*n)) - 1;
end
```

9 ANEXO C

```
%%Estudio post-RNA
%Con este código hacemos el estudio post-rna para incrementar la precisión
%del brazo robótico.
bien=0;
mal=0;

for i=1:length(OutputMotor) %Para cada salida de la red neuronal

    if(OutputMotor(i)<0.7 && Salida(i)==0) %Si la salida de la RNA<0.7 y el
valor real de salida=0
        bien=bien+1; %La RNA ha clasificado bien
        OutputMotor_rev(i)=0; %Ponemos la salida revisada del motor a 0.
    elseif(OutputMotor(i)>0.3 && Salida(i)==1) %Si la salida de la RNA>0.3 y
el valor real de salida=1
        bien=bien+1; %La RNA ha clasificado bien
        OutputMotor_rev(i)=1; %Ponemos la salida revisada del motor a 1.
    else
        if (OutputMotor(i)>0.3 && OutputMotor(i)<0.7) %%Si la salida se
encuentra entre 0.3 y 0.7 será motivo de estudio
            if (OutputMotor(i)>0.3 && OutputMotor(i)<0.5 && Salida(i)==1) %Si
la salida de la RNA se encuentra entre 0.3 y 0.5 y el valor real de salida=1
                if (Entrada(41,i)>1e4) %Si el valor de la PSD>1e4
                    bien=bien+1; %Nuestro criterio post-RNA es bueno
                    OutputMotor_rev(i)=1; %Ponemos la salida revisada del
motor a 1
                else
                    estudio_KO_b=estudio_KO_b+1; %Si el valor de la PSD<1e4
                    mal=mal+1; %Nuestro criterio post-RNA es malo
                    OutputMotor_rev(i)=0; %Ponemos la salida revisada del
motor a 0

                end
            elseif (OutputMotor(i)>0.5 && OutputMotor(i)<0.7) %Si la salida
de la RNA está entre 0.5 y 0.7
                 OutputMotor_rev(i)=1; %Ponemos la salida revisada del motor a
1
            end
        else %en cualquier otro caso
            mal=mal+1; %La red habrá clasificado mal
        end
    end
end
```

10 ANEXO D

```
%% Clasificación salida motor tras comprobación del estudio post-RNA

for i=1:length(OutputMotor) %Para cada salida de la red neuronal

    if(OutputMotor(i)<0.3) %Si la salida de la RNA<0.3
        OutputMotor_rev(i)=0; %Ponemos la salida revisada del motor a 0.
    elseif(OutputMotor(i)>0.7) %Si la salida de la RNA>0.7
        OutputMotor_rev(i)=1; %Ponemos la salida revisada del motor a 1.
    else
        if (OutputMotor(i)>0.3 && OutputMotor(i)<0.7) %%Si la salida se
encuentra entre 0.3 y 0.7 será motivo de estudio
            if (OutputMotor(i)>0.3 && OutputMotor(i)<0.5) %Si la salida de la
RNA se encuentra entre 0.3 y 0.5
                if (Entrada(41,i)>1e4) %Si el valor de la PSD>1e4
                    OutputMotor_rev(i)=1; %Ponemos la salida revisada del
motor a 1
                else
                    OutputMotor_rev(i)=0; %Ponemos la salida revisada del
motor a 0
                end
            elseif (OutputMotor(i)>0.5 && OutputMotor(i)<0.7) %Si la salida
de la RNA está entre 0.5 y 0.7
                 OutputMotor_rev(i)=1; %Ponemos la salida revisada del motor a
1
            end
        end
    end
end

%% Control de dos servomotores con movimientos independientes

%Conexión del arduino y de los servomotores
a=arduino('COM3', 'uno');
s1=servo(a,'D9');
s2=servo(a, 'D10');
v1=zeros(1,i);
v2=zeros(1,i);

for j=1:i
    if OutputMotor_rev(j)==1 %Si el valor de la salida=1 moveremos el robot a
la posición:
        v1(j)=150/180; %Servo1=150° en rad
        v2(j)=120/180; %Servo2=120° en rad
    else %Si el valor de la salida=0 moveremos el robot a una posición de
"reposo":
        v1(j)=90/180; %Servo1=90° en rad
        v2(j)=90/180; %Servo2=90° en rad
    end
end

M=length(v1);
for i=1:M

    ang1=v1(i); %Posición a la que se moverá el Servo1
```

```
    ang2=v2(i); %Posición a la que se moverá el Servo2
    %Escribimos los ángulos en los servos correspondientes
    writePosition(s1,ang1);
    writePosition(s2,ang2);
    %Leemos la posición a la que se han movido los servos como
    %"realimentación" al usuario
    pos1=readPosition(s1);
    pos2=readPosition(s2);
    %Imprimimos por pantalla las posiciones de los servos
    fprintf('Posición actual del Servo1 es de %d grados\n',pos1*180);
    fprintf('Posición actual del Servo2 es de %d grados\n',pos2*180);

    pause(2); %Se mantiene la posición durante 2 segundos

end
```

REFERENCIAS

[1] E. espectador. [En línea]. Available: https://www.elespectador.com/noticias/ciencia/tetraplejico-come-con-neuroprotesis-articulo-686904.

[2] ABC. [En línea]. Available: https://www.abc.es/salud/noticias/20150521/abci-science-paraplejia-robot-201505211707.html.

[3] D. University. [En línea]. Available: https://riunet.upv.es/bitstream/handle/10251/12540/tesina_con_publicaciones.pdf?sequence=1.

[4] Andade. [En línea]. Available: https://andade.es/.

[5] 2. minutos. [En línea]. Available: https://www.20minutos.es/noticia/2900121/0/avances-protesis-piernas-brazos/sensitivas-oseointegracion-max-ortiz-catalan/.

[6] N. Geographic. [En línea]. Available: https://www.nationalgeographic.com.es/ciencia/grandes-reportajes/secretos-del-cerebro-2_8039/3.

[7] Viguera. [En línea]. Available: https://www.viguera.com/es/index.php?controller=attachment&id_attachment=84.

[8] CSIC. [En línea]. Available: http://www.fgcsic.es/lychnos/es_es/articulos/Brain-Computer-Interface-aplicado-al-entrenamiento-cognitivo.

[9] Brainstorm. [En línea]. Available: https://neuroimage.usc.edu/brainstorm/Introduction.

[10] N. Peirotén López de Arbina, «Peirotén López de Arbina, Nagore,» de *TFG - Diseño de una red neuronal en Matlab para análisi de señales de electroencefalograma*, Sevilla, 2018, p. 18.

[11] U. d. Valle. [En línea]. Available: https://www.academia.edu/8172191/Redes_Neuronales_Artificiales_Arquitecturas_y_Aprendizaje.

[12] F. S. Caparrini. [En línea]. Available: http://www.cs.us.es/~fsancho/?e=165.