UNIVERSIDAD DE VALLADOLID

FACULTAD DE MEDICINA

ESCUELA DE INGENIERÍAS INDUSTRIALES

TRABAJO DE FIN DE GRADO

GRADO EN INGENIERÍA BIOMÉDICA

Simuladores Cerebrales: Revisión de Modelos Micro- y Macroescala

Autor:

D. Javier de Castro Poy

Tutora:

D.ª Belén Carro Martínez

Valladolid, 20 de septiembre de 2023

TÍTULO:	**Simuladores Cerebrales: Revisión de Modelos Micro- y Macroescala**
AUTOR/A:	**D. Javier de Castro Poy**
TUTOR/A:	**D.ª Belén Carro Martínez**
DEPARTAMENTO:	**Departamento de Teoría de la Señal y Comunicaciones e Ingeniería Telemática**

TRIBUNAL

PRESIDENTE:	**D.ª Belén Carro Martínez**
SECRETARIO:	**D.ª Rosa M. Menchón**
VOCAL:	**D. Miguel Ángel Martín**
SUPLENTE 1:	**D.ª María García**
SUPLENTE 2:	**D. Jesús Poza**

FECHA: 20 de septiembre de 2023

CALIFICACIÓN:

Resumen

Las simulaciones de redes cerebrales pretenden comprender las funciones del cerebro tanto en condiciones normales como patológicas. A tal fin, existen en la actualidad múltiples simuladores y paquetes software pertenecientes al ámbito de la neurociencia computacional.

El objetivo final de los modelos computacionales consiste en tratar de explicar la relación entre la estructura, la función y la dinámica cerebrales. Los modelos multiescala trabajan con datos biológicos de distinto tipo y granularidad, en un rango que va desde modelos de neuronas, sinapsis y microcircuitos -microescala- hasta modelos a macroescala con cerebros virtuales. En el presente trabajo se pretende realizar una revisión de los modelos microescala y macroescala, resaltando sus principales características y funcionalidades y su orientación para investigación. Finalmente, se trabajará experimentalmente con un modelo macroescala, para el cual se elaborará una guía de usuario.

Palabras clave

Neurociencia, cerebro, modelos, multiescala, simulación

Abstract

The simulations of brain networks aim to understand brain functions in both normal and pathological conditions. To this end, there are currently multiple simulators and software packages within the field of computational neuroscience.

The ultimate goal of computational models is to attempt to explain the relationship between brain structure, function, and dynamics. Multiscale models work with biological data of different types and granularity, ranging from models of neurons, synapses, and microcircuits - at the microscale - to macro-scale models with virtual brains. In this work, we aim to provide a review of microscale and macro-scale models, highlighting their key characteristics, functionalities, and their orientation for research. Finally, experimental work will be conducted with a macro-scale model, for which an user guide will be developed.

Keywords

Neuroscience, brain, models, multiscale, simulation

Agradecimientos

Me gustaría dedicar este trabajo a mis padres y a mi hermano por ser mi pilar fundamental y por no haber dudado de mí en ningún momento. Gracias por haber estado allí siempre que lo he necesitado.

“What I cannot create I do not understand.”

Richard Feynman

ÍNDICE

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

1. Introducción

El cerebro es uno de los sistemas más complejos que pueden ser encontrados en la naturaleza. La neurociencia es el campo que se ocupa de su estudio y tiene por objetivo principal comprender cómo a partir de los procesos que se dan en este órgano surge su complejo comportamiento [1]. Este problema puede ser abordado desde diferentes perspectivas, sin embargo una que resulta especialmente interesante es la simulación de sistemas neuronales y cerebrales. Los sistemas de simulación pueden ayudar a entender cómo ocurren los fenómenos biológicos que dan lugar a la cognición o qué mecanismos fallan en un cerebro con cierta patología, mediante la comprobación directa de hipótesis. Un sistema de simulación implementa un algoritmo que busca soluciones de ecuaciones matemáticas que describen la dinámica y los patrones de conectividad de un grupo de neuronas y sus sinapsis [2]. Estos simuladores deben ser capaces de predecir medidas que pueden ser recogidas en pacientes reales con el objetivo de ser comparadas.

Los simuladores deben caracterizarse por ser de propósito general, es decir, no deben diseñarse para comprobar una hipótesis en concreto, sino que deben de tener la flexibilidad suficiente como para admitir distintas propuestas. El objetivo principal de estas herramientas es comprobar distintas propuestas de modelos de redes para compararlas con datos reales y determinar qué modelo puede explicar mejor la realidad, de forma que, eventualmente, surja un modelo que finalmente recoja las características biológicas del sistema. Además, también son útiles para realizar una validación de los métodos estadísticos que se utilizan para estimar las conectividades funcionales entre las poblaciones y áreas neuronales a partir de las medidas experimentales, y comprobar su capacidad para recuperar la conectividad subyacente a partir de las señales medidas [3].

1.1. Motivación

Actualmente, en el campo de la neurociencia existe una necesidad imperante de encontrar distintos métodos que permitan conseguir un conocimiento más profundo sobre cómo funciona realmente el cerebro y cómo surge el comportamiento que este manifiesta. Una expansión de su entendimiento permitirá, eventualmente, ayudar en el diagnóstico y el tratamiento de enfermedades que a día de hoy se desconoce cómo actúan por completo. El uso de simuladores cerebrales puede ayudar comprender mejor cómo el cerebro opera y toma las complejas decisiones en ordenes temporales muy reducidos. Interesa conocer y evaluar las distintas opciones disponibles para su aplicación en investigación y clínica. Además, el estudio de simuladores cerebrales consiste en una integración perfecta de los conocimientos de medicina y biología con el uso de softwares informáticos y otros campos técnicos, lo que le convierte en un campo de estudio óptimo para la ingeniería biomédica.

1.2. Objetivos

El presente trabajo comprende de una serie de objetivos que se irán desarrollando durante el documento:

- El objetivo inicial de este trabajo consiste en realizar una revisión de los simuladores y paquetes software, tanto microescalares como macroescalares más importantes de uso común en investigación científica, destacando las características de cada uno de ellos y comparándolos.
- También se lleva a cabo una revisión del estado del arte de la aplicación tanto en investigación como en clínica de estos simuladores, planteando los distintos retos y dificultades que se originan a la hora de desarrollarlos.
- Por último, se tratará la integración multiescalar, donde se suman las características y funcionalidades de los sistemas microescala y macroescala en un único entorno de simulación.

1.3. Hipótesis

Se postula que los simuladores cerebrales evaluados, tanto microescalares como macroescalares, desempeñan un papel fundamental en la comprensión de los fenómenos neurofisiológicos que tienen lugar, desde los procesos que ocurren a nivel neuronal hasta el cerebro completo, pasando por todas las escalas intermedias. Estos tendrán aplicaciones sustantivas en la investigación y, sobre todo, en la clínica, ayudando a los pacientes que sufran patologías a ser diagnosticadas y tratadas de una forma más adecuada y efectiva.

1.4. Estructura

En cuanto a la estructura del trabajo, primero se explicarán los conceptos básicos sobre la simulación y las bases biológicas que rigen el comportamiento del cerebro. Posteriormente, se tratarán los simuladores de microescala NEST, NEURON y Brian, de los que se explicarán sus características y aplicaciones y se realizará una comparación de ellos. Posteriormente se explicarán las propiedades y aplicaciones de *The Virtual Brain*, un simulador macroescalar; y por último se mostrará el trabajo experimental desarrollado en esta plataforma.

2. Principios de la simulación

Antes de comenzar con el desarrollo de distintos softwares, es esencial entender una serie de conceptos previos sobre la simulación que se irán desarrollando a lo largo de todo el trabajo. Existen distintos tipos de simuladores, desde aquellos que emulan la actividad de neuronas concretas agrupadas en un circuito a nivel microscópico, hasta los que realizan simulaciones de todo el cerebro en su conjunto o a nivel macroscópico. Lo primero que se debatirá es la organización en diferentes escalas de este órgano, cómo interactúan entre ellas y cómo abordar este problema en la simulación. Además, se indicarán una serie de conceptos biológicos básicos necesarios para el posterior desarrollo del documento. También se mostrarán la existencia de diferentes modelos que reflejan el comportamiento de una o un grupo de neuronas. Por último, se expondrá como el cerebro puede ser entendido como una red.

2.1.El problema de la integración multiescalar.

a) La naturaleza multiescalar del cerebro

La multiescalaridad es una propiedad fundamental de muchos sistemas complejos, incluyendo el cerebro humano. Esta propiedad se refiere a cómo los sistemas operan en múltiples escalas espaciales y temporales, que abarcan el nivel molecular, el celular, los circuitos neuronales locales y regionales e incluso las interacciones sociales. Todas estas van a interactuar entre sí, surgiendo la actividad cerebral. Por esta razón, para entender el cerebro en su conjunto, es esencial comprender cómo actúa cada una de estas escalas y cómo se relacionan entre sí [4].

Desde la actividad neuronal individual hasta la dinámica colectiva de áreas relativamente grandes del cerebro, la organización espacial de este órgano se divide en diferentes niveles. A nivel microscópico se encuentra la unidad básica del sistema nervioso, la neurona. Es aquí donde ocurren fenómenos electroquímicos, como el flujo de iones o el potencial de membrana que permite la comunicación entre células y el procesamiento de información por estas. La siguiente escala es la mesoscópica, donde las poblaciones neuronales interactúan entre sí para formar redes funcionales que procesan información específica. Por último, aparece la escala macroscópica, donde diferentes regiones cerebrales interactúan entre sí para coordinar funciones cognitivas complejas y puede ser registradas mediante diversas técnicas.

También se debe tener en cuenta la escala temporal en la que suceden todos estos fenómenos. Por ejemplo, los impulsos neuronales que suceden en milisegundos afectan a los patrones de actividad cerebral que pueden ser recogidos por distintas pruebas como el electroencefalograma, obteniendo señales que pueden tener varios minutos de duración [5].

Integrar todos estos sucesos que surgen a diferentes escalas y que interactúan entre sí en un mismo marco teórico, no es sencillo y origina una serie de problemas que pueden ser abordados mediante el uso de modelos cerebrales multiescala.

b) Estrategias de simulación

El problema de la integración multiescalar puede ser abordado desde dos puntos de vista diferentes: el problema directo y el problema inverso. El problema directo consiste en tratar de predecir los efectos a partir del conocimiento de las causas. Es decir, cómo las causas elementales, por ejemplo un conjunto de señales de células aisladas, van a determinar mediciones colectivas observables como señales de resonancia magnética nuclear (RMN), electroencefalografía (EEG), magnetoencefalografía (MEG) y tomografía por emisión de positrones (PET). La resolución del problema directo se aborda mediante una estrategia de modelado "*bottom-up*" o "de abajo a arriba", puesto que se va progresando desde las escalas más pequeñas hacia las mayores [4].

La estrategia "de abajo a arriba" incluye modelos de microescala, biofísicamente detallados de neuronas y sinapsis que pueden ser ensamblados formando microcircuitos, de los que se puede reproducir su dinámica local. Estos modelos se pueden simplificar en redes de neuronas de impulsos (SNN) o en modelos de campo medio (MF) para permitir simulaciones más manejables a gran escala.

Por otro lado, el problema inverso consiste en, a partir de las mediciones observables a nivel sistémico, inferir las causas microscópicas desconocidas que las generan. Para su resolución se usa la estrategia de modelado "*top-down*" o de "arriba hacia abajo". Esta suele ser la estrategia más común y requiere de métodos numéricos para tratar los sistemas no lineales del cerebro.

Los modelos de macroescala o modelos de cerebro virtual se incluyen en esta última estrategia. Estos derivan de reconstrucciones anatomo-funcionales basadas en registros cerebrales de conjunto (como MRI, EEG, MEG o PET) y atlas. Los nodos contienen modelos simplificados de la actividad neuronal (como masas neurales o MF) que se conectan a través del conectoma. La dinámica cerebral generada por estos modelos se puede correlacionar con la observada experimentalmente para extraer parámetros ocultos mediante un proceso inferencial de inversión del modelo.

Los modelos que han sido mencionados son considerados como "*data-driven*" o basados en datos, puesto que utilizan datos biológicos a diferentes escalas para simular la actividad cerebral. Sin embargo, también se deben tener en cuenta los modelos "*task-driven*" o basados en tareas que son diseñados para simular funciones cerebrales específicas, como el aprendizaje, la cognición, el control sensoriomotor y el comportamiento; con el objetivo de entender los mecanismos que se llevan a cabo para implementar estas funciones.

El modelado y simulación del cerebro se ve claramente afectado por la evolución de las técnicas que permiten obtener datos experimentales del cerebro a diferentes escalas y de la capacidad de computación de los ordenadores para realizar simulaciones masivas.

2.2.Bases biológicas de la dinámica neuronal cerebral

a) Microescala

La neurona

Las neuronas son las células que actúan como unidades elementales de procesamiento del sistema nervioso central, las cuales están conectadas con otras en un patrón indicado. Otro tipo de células existentes en el sistema nervioso son las células de la glía que proporcionan energía y estabilidad estructural al tejido cerebral [6].

La neurona *spiking* o de impulsos ideal se divide en tres partes funcionales:

- Dendritas: recoge las señales de otras neuronas y las transmite al soma.
- Soma: unidad de procesamiento central que realiza un procesamiento no lineal, si el input total que llega al soma excede un cierto umbral, se genera una señal de salida.
- Axón: lleva la señal a otras neuronas. Estas se pueden encontrar cerca o en otras áreas del cerebro.

La unión entre dos células se llama sinapsis**.** La neurona que envía la señal se llama presináptica y la que la recibe se llama postsináptica.

La señal con la que se comunican las neuronas consiste en pulsos eléctricos cortos. Estos pulsos se llaman potenciales de acción o *spikes*, de ahí el nombre *spiking neurons*. Para poder comprender cómo se genera estos potenciales de acción y cómo se propagan a través de las neuronas, es importante comprender las bases fisiológicas que sustentan estos procesos, con la intención de poder modelarlos posteriormente.

Bases fisiológicas

La neurona, al igual que el resto de las células del cuerpo, está rodeada por una membrana. Dicha membrana está formada por una bicapa fosfolipídica en la que se distribuye una componente proteica dando lugar al modelo del mosaico fluido. La existencia de la membrana celular provoca la aparición de un medio interno celular, caracterizado por una gran concentración de potasio (K^+), y un medio extracelular, caracterizado por una alta concentración de sodio (Na^+).

Las distintas concentraciones de solutos van a generar un gradiente de concentración a ambos lados de la membrana celular. Los iones pueden moverse a través de la membrana y lo pueden hacer a favor de dicho gradiente, lo que denominamos transporte pasivo, puesto que no se necesita energía; o lo pueden hacer en contra de gradiente, lo que llamamos transporte activo, puesto que es necesario un aporte de energía que pueda vencer a dicho gradiente. Este último tipo es esencial para mantener la diferencia de concentraciones, siendo un ejemplo de este las bombas Na^+- K^+.

Las bombas Na^+- K^+ tienen una estequiometría característica: por cada tres iones de Na^+ llevados fuera de la célula, se introducen dos iones K^+. Podemos observar que el movimiento de estos iones se realiza en contra de gradiente y que se bombea más carga positiva hacia fuera de la célula que hacia dentro generándose una diferencia de potencial. A esta diferencia de potencial eléctrico entre el interior y el exterior de una célula se le

llama potencial de membrana. En las neuronas el potencial de membrana es electronegativo, es decir, el interior celular es más negativo que el medio extracelular. Por convenio se expresa el potencial intracelular respecto al extracelular.

En la membrana celular existen canales iónicos específicos para cada ion, que favorecen su movimiento a favor de gradiente, generándose un potencial de difusión. Debido a que para este ion también existe una diferencia de potencial, llegará un momento en el que la difusión neta de dicho ion será nula, llegando a un equilibrio electroquímico en el que las fuerzas eléctricas y químicas que mueven los iones son iguales y opuestas. Para el potencial de difusión que ocurre esto se llama potencial de equilibrio (**tabla 2.1)**.

El potencial de equilibrio de un ion a una diferencia de concentración dada a través de una membrana permeable a este ion se calcula con la ecuación de Nernst:

$$E_x = \frac{k \cdot T}{q} \cdot \ln\left(\frac{[C_i]}{[C_e]}\right) \qquad \text{Eq. 2.1.}$$

Donde

- E_x: potencial de equilibrio para el ion X
- k: constante de Boltzmann
- T: temperatura
- q: carga del ion
- C_i: concentración en el medio interno
- C_e: concentración en el medio externo

Tabla 2.1. Valores típicos de potencial de equilibrio para ciertos iones

Ion	**Potencial de Equilibrio**
Na^+	+65 mV
Ca^{2+}	+120 mV
K^+	-85 mV
Cl^-	-90 mV

El **potencial de reposo** en las células excitables es aquel potencial de membrana que tiene la célula en reposo (el periodo entre potenciales de acción). Si la membrana fuese permeable solo para un ion, el potencial de membrana coincidiría con el potencial de equilibrio de dicho ion. Sin embargo, la membrana celular es permeable para distintos iones, por lo que para calcular su potencial de reposo se utiliza la **ecuación de Goldman**:

$$Em = \frac{g_{K^+}}{g_T} \cdot E_{K^+} + \frac{g_{Na^+}}{g_T} \cdot E_{Na^+} + \frac{g_{Cl^-}}{g_T} \cdot E_{Cl^-} + \frac{g_{Ca^{2+}}}{g_T} \cdot E_{Ca^{2+}} \qquad \text{Eq. 2.2.}$$

Siendo:

- Em: Potencial de membrana
- g_X: Conductancia para ese ion
- g_T: Conductancia total
- E_X: Potencial de equilibrio para dicho ion

Según la ecuación, cada ion va a tratar de llevar el potencial de membrana hacia su potencial de equilibrio. Cuanto mayor sea la conductancia de los iones en el reposo, más cerca estará el potencial de reposo del potencial de equilibrio de dichos iones. En el reposo, la conductancia es mayor para el K^+ y el Cl^-, lo que explica que las neuronas excitables tengan un potencial de reposos electronegativo de entre -70 y -80 mV. La diferencia entre el potencial de membrana y el potencial de equilibrio de un ion es la fuerza impulsora neta. Si es negativa para un catión entrará en la célula, si fuese un anión saldría de esta y al revés si fuese positiva.

Como ya hemos mencionado anteriormente, las señales con las que las neuronas transmiten información son unos pulsos eléctricos denominados potenciales de acción. Estos consisten en una rápida despolarización (el potencial de membrana se hace menos negativo), seguida de una repolarización hasta alcanzar de nuevo el potencial de reposo.

Inicialmente el potencial de membrana es de -70 mV, debido a la alta permeabilidad del K^+ y el Cl^-, al contrario de la permeabilidad del Na^+ que es baja. Cuando hay una corriente de entrada a la célula, su membrana se despolariza y si llega al umbral de aproximadamente a -60 mV, se van a activar los canales de Na^+ dependientes de voltaje, abriéndose sus puertas de activación y aumentado la permeabilidad de este ion produciéndose una corriente de entrada, despolarizando el potencial de membrana hasta el potencial de equilibrio del sodio de -65 mV. Ante la despolarización inicial, las puertas de inactivación del sodio y los canales de potasio también van a responder, pero más lentamente. Por lo tanto, las puertas de inactivación cerrarán los canales de sodio poniendo fin a la fase de ascenso y se abrirán los canales de potasio, aumentando su conductancia llegando a ser mayor que en el reposo (postpotencial hiperpolarizante) y permitiendo su salida de la célula, repolarizando el potencial de membrana. Finalmente se vuelve a las condiciones normales de reposo.

Las características del potencial de acción pueden resumirse en:

- Tiene forma estereotípica, no cambia en su recorrido despolarizándose y repolarizándose al mismo potencial respectivamente.
- La propagación del potencial de acción en una fibra nerviosa es unidireccional, se transmite la corriente local a una región adyacente inactiva, pero la región anterior de donde proviene el potencial de acción primario se encuentra en periodo refractario, por lo que no se producirá otro potencial de acción.
- La respuesta es de todo o nada. Tan sólo se produce el potencial de acción si el potencial de membrana alcanza un cierto umbral que active los canales dependientes de voltaje. En caso contrario, no se producirá.

Si todos los potenciales de acción tienen una forma parecida, esta no será determinante a la hora de transmitir información, pero sí lo será el número de potenciales de acción que se produce en un cierto tiempo, llamado tren de impulsos o *spike trains.*

La sinapsis

La sinapsis es la comunicación entre dos neuronas donde se transmite la información. Esta es unidireccional, de la presináptica a la postsináptica. Se conocen dos tipos de sinapsis la eléctrica y la química.

- **Sinapsis eléctrica**: las células están conectadas entre sí mediante estructuras llamadas uniones de hendidura (*gap junctions*). Estas se caracterizan por ser muy estables y están atravesadas por unos canales que permite la corriente iónica entre células. Esta forma de comunicación es muy eficiente y se traduce en respuestas rápidas y coordinadas.

- **Sinapsis química**: en este caso, las neuronas no están en contacto, sino que existe una separación de 20-40 nm. Cuando el potencial de acción llega al botón presináptico, la despolarización activa canales de calcio provocando su entrada favoreciendo la exocitosis. Las vesículas sinápticas se van transportando hacia la zona activa de la sinapsis y son capaces de, por exocitosis, liberar su contenido dentro de la hendidura sináptica. Dentro de esta, los neurotransmisores se unen de forma específica a un receptor de la célula postsináptica, lo que provoca la conversión de una señal química en eléctrica. Esta conversión de señales es lo que permite amplificar y modular con gran precisión el efecto que queremos provocar.

Dependiendo de la interacción que surja entre los neurotransmisores y los receptores que los captan se pueden diferenciar dos tipos de sinapsis. Las sinapsis excitatoria incrementa la posibilidad de que se active un potencial de acción en la neurona postsináptica y genera un potencial postsináptico excitatorio (PPSE) que despolariza la célula acercándola al umbral, gracias a la apertura de canales de sodio y potasio. Entre los neurotransmisores excitadores se encuentran el glutamato o la serotonina.

Por otro lado, las sinapsis inhibitorias van a generar un potencial postsináptico inhibitorio que va a producir una hiperpolarización alejando a la neurona del umbral. Estos se producen por la apertura de canales Cl-. Algunos neurotransmisores inhibitorios son GABA y glicina.

b) Mesoescala

Estos elementos neuronales se van a comunicar entre sí formando redes neuronales que van a generar una actividad coordinada entre todas ellas, a partir de la cual van a surgir procesos cognitivos que involucran la participación de diferentes áreas del cerebro a gran escala. Por lo tanto, una característica importante de las neuronas va a ser su conectividad con otras neuronas, lo que va a determinar las funciones mentales en las que va a participar. Si se analiza la conectividad de un grupo de neuronas, se puede observar que estas se organizan en circuitos, surgiendo un nuevo nivel intermedio entre la actividad local de las neuronas, o los nodos de la red, y la dinámica extensa generada por la red en su totalidad, el llamado nivel mesoescalar. En la mesoescala, los nodos de una red se pueden agrupar con otros nodos con características similares, formando grupos o comunidades [7].

La detección de configuraciones a nivel mesoescala puede desvelar cómo el procesamiento paralelo que tiene lugar en diferentes grupos neuronales se integra para dar lugar a fenómenos de una mayor complejidad [8]. Tradicionalmente, este nivel se ha entendido cómo ciertas comunidades segregadas procesan información especializada, mientras que ciertos nodos con un alto perfil de conectividad integran esta información. Por el contrario, según [7] se han encontrado evidencias de estructuras no en las que muchos de sus nodos están conectados con otras comunidades, en vez de ser unos pocos los que lo hacen, Por lo que todavía se debe investigar cómo se produce este procesamiento entre los diferentes grupos.

c) Macroescala

Todas las conexiones físicas que se van a producir entre los distintos circuitos neuronales y que van a transportar información entre estos; van a dar lugar a tractos de materia blanca que contienen axones que conectan regiones de materia gris. Esta es la llamada conectividad estructural del cerebro, la cual se obtiene mediante la resonancia magnética ponderada por difusión (DWI). Esta consiste en capturar varias veces un voxel, variando cada una de las veces el eje espacial a los que va a ser sensible la difusión. El resultado es el tractograma que puede ser visualizado con muchos detalles. Toda esta información puede ser recogida en matrices de conectividad estructural, las cuales describen las relaciones entre distintas regiones cerebrales [9].

Sin embargo, la conectividad estructural no es suficiente para describir la dinámica de red cerebral que surge en el cerebro como consecuencia de las interacciones de un gran número de neuronas. Es necesario conocer otra característica fundamental de la actividad cerebral, la conectividad funcional. La conectividad funcional describe la dependencia estadística de la actividad de dos regiones remotas. Esta se obtiene mediante la imagen de resonancia magnética funcional (fMRI), de la que se obtiene la señal *blood oxygenation-level* dependent (BOLD), que relaciona un aumento en la actividad de las neuronas con un mayor consumo de oxígeno. Realizando una correlación de Pearson entre dos de las señales BOLD obtenidas de cada una de las regiones, se obtiene la matriz de conectividad funcional. Sin embargo, se debe tener en cuenta que la conectividad funcional no es constante, sino que puede variar con el tiempo. Por eso se define la conectividad dinámica funcional, que describe dicha variación temporal. La obtención de estas matrices no se va a realizar considerando la señal BOLD completa. En este caso esta será dividida en intervalos temporales, sobre los que se va a calcular la correlación de Pearson de cada uno. Así, se obtiene una matriz funcional por cada intervalo y se puede evaluar cómo estas son diferentes en cada uno de estos, observando una evolución en la conectividad funcional. [9] [10].

Además, existen otras técnicas que permiten el estudio del cerebro cómo son la electroencefalografía (EEG) y la magnetoencefalografía (MEG). El EEG refleja la actividad neuronal del cerebro y recoge los campos eléctricos que se producen como consecuencia del cambio del potencial de acción de las membranas celulares. La magentoencefalografía, por su parte captura los campos magnéticos que se producen en el cerebro. Ambas señales se caracterizan por su alta resolución temporal y pueden ser tratadas y analizadas para obtener información del comportamiento de grupos neuronales [11] [12].

2.3.Modelado de sistemas neuronales

El principal tipo de modelos usados para la simulación de sistemas neuronales y cerebrales son los llamados modelos generativos [13]. Estos consisten en ecuaciones que determinan la evolución de las señales en función de los parámetros del sistema, como por ejemplo el modelo Wilson-Cowan o el modelo Kuramoto. Estos modelos pueden ser divididos en:

- Modelos biofísicos: son aquellos que están basados en restricciones y suposiciones biológicas y tratan de reproducir sucesos de forma realista.
- Modelos fenomenológicos: reproducen comportamientos observables de un sistema, capturando sus características estadísticas o dinámicas de interés, sin entrar en detalles biológicos, consiguiendo así reducir la complejidad del problema.
- Modelos agnósticos: modelos basados en datos que son capaces de reproducir el comportamiento a partir de una escasa cantidad de datos.

a) Modelos microescala

Los modelos neuronales de células *spiking* o de impulso, simulan el comportamiento de una neurona biológica que emite impulsos eléctricos breves llamados *spikes*. Estos *spikes* se generan cuando el potencial de membrana de la neurona alcanza un umbral, y se transmiten a otras neuronas a través de las sinapsis. A continuación se van a explicar dos tipos de estos modelos: modelo Hodkin-Huxley e *integrate-and-fire neurons* [14].

Modelo Hodgkin–Huxley

El modelo de Hodgkin-Huxley es un modelo matemático que permite explicar cómo se genera los potenciales de acción en una neurona [15]. Este consiste en considerar a la neurona como un circuito eléctrico, en donde cada uno de sus elementos representan un elemento biofísico de la célula. En este modelo, la membrana celular semipermeable, que separa los medios interno y externo, se modela como un condensador capaz de almacenar carga eléctrica. También va a haber tres resistencias que representan los tres tipos de canales que moderan el flujo de iones: sodio, potasio y otro no específico que regula la corriente principalmente de iones de cloro. El potencial de Nernst para cada tipo de ion va a estar representado por las tres baterías distintas, E_{Na}, E_K, E_L.

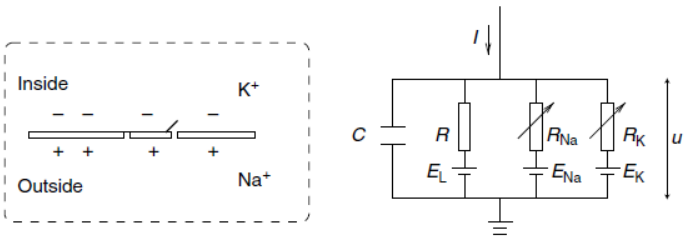


Figura 2.1. Modelo de Hodgkin-Huxley

Cuando se introduce una corriente en este circuito *I(t)*, parte de ella va a dirigirse al capacitador cargándolo $I_C(t)$ y el resto va a atravesar los canales de iones $I_k(t)$.

$$I(t) = I_C(t) + \sum_k I_k(t) \qquad \text{Eq. 2.3}$$

Sabiendo que la capacidad es $C=q/u$, siendo q una carga y u el voltaje a través del capacitador:

$$C\frac{du}{dt} = -\sum_k I_k(t) + I(t) \qquad \text{Eq. 2.4.}$$

Los canales van a estar caracterizados por su resistencia o su conductancia (la inversa de la resistencia, g). Esta va a ser constante para R, pero la de los canales de sodio y potasio van a variar dependiendo de si estos están abiertos o no. Por ello, se introducen tres variables (m,h,n) que modulan la conductancia de los canales y representan la probabilidad de que estén abiertos en un momento determinado. Cambios de estas variables en ciertas situaciones van a originar los potenciales de acción.

$$\sum_k I_k(t) = g_{Na}m^3h(u - E_{Na}) + g_K n^4(u - E_K) + g_L(u - E_L) \qquad \text{Eq. 2.5.}$$

Modelo integrate-and-fire

Los potenciales de acción de las neuronas, como se ha explicado en el anterior aparatado b), tienen una forma muy parecida. Por lo tanto, lo importante en realidad es si se produce dicho potencial, más allá de su forma. Por esta razón, puede ser más útil desarrollar un modelo fenomenológico más simple que modele la producción de potenciales de acción en un tiempo concreto sin incluir los cambios reales asociados con el voltaje de membrana y las conductancias, en vez de usar un modelo más complejo como el de Hodgkin-Huxley con el que se puede reproducir medidas electrofisiológicas detalladas.

Aquí es donde surgen los modelos *integrate-and-fire*. Estos describen el potencial de membrana de una neurona basándose en las entradas sinápticas y la corriente que se le inyecta. El potencial inicial de membrana es el potencial de reposo. Cuando el potencial de membrana supera un umbral, se produce un potencial de acción. Inmediatamente después el potencial vuelve a su valor de reposo. Además, se puede añadir un periodo refractario absoluto, en donde el valor umbral es infinito; y un periodo refractario relativo, donde el valor umbral es muy alto. Este modelo es de un solo compartimento o puntual, es decir, no se tiene en cuenta su estructura espacial. El modelo se describe como:

$$C\frac{du}{dt} = I(t) \qquad \text{Eq. 2.6.}$$

Además se puede incluir en el modelo la corriente de "fuga" o "*leak*" que representa la difusión de iones de la membrana, obteniendo el modelo *Leaky integrate-and-fire* (LIF), donde:

$$C\frac{du}{dt} = I(t) + I_{leak}(t) \qquad \text{Eq. 2.7.}$$

$$I_{leak}(t) = -\frac{C_m}{\tau_m}[v(t) - V_0 \qquad \text{Eq. 2.8.}$$

Siendo V_0 el potencial de reposo y τ_m la constante de tiempo de la membrana pasiva.

Los inputs sinápticos que recibe la neurona, a través de otras a las que está conectada, pueden ser excitatorios o inhibitorios y tienen asociadas un peso. Estas sinapsis pueden ser modeladas o bien con modelos sinápticos de corriente o bien con modelos sinápticos de conductancia. Los modelos de corriente provocan un cambio en el potencial de membrana independientemente de su valor y representan cada input como una corriente que se inyecta en la neurona. Si llegan varias sinapsis a la vez se suman directamente. Por otro lado, los modelos sinápticos de conductancia producen un cambio en la conductancia en la membrana dependiendo del su estado actual. Es decir, una sinapsis va a provocar una variación proporcional a la diferencia entre el potencial actual y el potencial de reversión de un canal iónico [16].

b) Masas neuronales y campos neuronales medios

A la hora de modelar el nivel mesoscópico, se podría definir extensas redes de modelos neuronales conectados entre sí, como serían las redes neuronales *spiking*. Esto supone reproducir las conexiones sinápticas de miles de neuronas, lo que puede resultar computacionalmente costoso y no muy práctico. Por ello, para describir este nivel de una forma más simplificada, se acude a los modelos de campo medio. Este enfoque trata de describir la actividad de un grupo neuronal como el promedio de todos los elementos que lo conforman, sin tener en cuenta las interacciones individuales. Además, son útiles para estudiar fenómenos a escala macroscópica, como las señales eléctricas y magnéticas del cerebro (EEG y MEG) [13].

Dentro de estos modelos de campo medio surgen los modelos de masa neuronal. Una masa neuronal tiene una variable de estado que puede indicar el potencial de membrana promedio o la tasa de disparo promedio de las neuronas que la componen. Esta masa va a estar conectada con otras y van a interaccionar entre ellas. Otro tipo de modelos son los modelos de campo neuronal, que van a determinar la evolución espacio-temporal de la actividad neuronal y pueden incluir diferentes tipos de neuronas, como excitatorias o inhibitorias, y o de sinapsis, como AMPA, NMDA o GABA. La principal diferencia entre las masas neuronales y los campos neuronales es que estos últimos definen la actividad neuronal tanto en el espacio como en el tiempo, pero los primeros tan solo lo van a definir en el tiempo [17].

Un ejemplo de modelo de masas neuronales es el modelo Wilson-Cowen. Este se compone de dos masas neuronales, una excitatoria y otra inhibitoria. Se basa en dos ecuaciones diferenciales no lineales acopladas describen la proporción de actividad o nivel medio de actividad de cada población, y su influencia mutua se describe mediante una función sigmoidea [18]. Las ecuaciones son:

$$\dot{E}_k = \frac{1}{\tau_e}(-E_k + (k_e - r_e E_k)S_e[\alpha_e(c_{ee}E_k - c_{ei}I_k + P_k - \theta_e + I(E_k, E_j, u_{kj}) + W_\varsigma \cdot E_j + W_\varsigma \cdot I_j)])$$

Eq. 2.9.

$$\dot{I}_k = \frac{1}{\tau_i}(-I_k + (k_i - r_i I_k) S_I[\alpha_i(c_{ie} E_k - c_{ii} I_k + Q_k - \theta_i + \mathcal{I}(E_k, E_j, u_{kj}) + W_\varsigma \cdot E_j + W_\varsigma \cdot I_j)])$$

Eq. 2.10.

Donde $\mathcal{I}(E_k, E_j, u_{kj})$ es el término de acoplamiento de largo alcance, $W_\varsigma \cdot E_j$ y $W_\varsigma \cdot I_j$ es la actividad excitatoria e inhibitoria de las unidades vecinas. Para una descripción completa de los parámetros consultar la tabla 10 de [18].

c) Modelos fenomenológicos

Los modelos fenomenológicos se basan en su capacidad para reproducir comportamientos observables reduciendo la dimensionalidad del problema al deshacerse de distintas consideraciones y restricciones biológicas. La dinámica neuronal que generan estos sistemas puede ser estudiada mediante el uso de herramientas provenientes de la teoría de sistemas dinámicos.

El modelo de Kuramoto es un ejemplo de un modelo fenomenológico que describe la sincronización de osciladores acoplados. El modelo asume que cada oscilador tiene una frecuencia natural y una fase que cambia según la influencia de los demás osciladores. Se puede expresar mediante una ecuación diferencial que relaciona la velocidad de cambio de la fase con la frecuencia natural y el acoplamiento medio entre los osciladores. Se consideran las poblaciones neuronales como osciladores similares débilmente acoplados entre sí. Ha sido utilizado para estudiar transiciones de fase no deseadas en enfermedades como epilepsia y el Parkinson [13].

d) Modelos agnósticos

Los modelos agnósticos tratan de reproducir el comportamiento de un sistema a partir de cierta cantidad de dato previos. Esta información previa puede resultar el principal limitante en este tipo de problemas, puesto que normalmente es necesaria una gran cantidad de datos.

Un ejemplo de este tipo de modelos es el modelo Hopfield. El modelo de Hopfield es un tipo de red neuronal artificial. El modelo consiste en un conjunto de unidades binarias que pueden estar en dos estados: activado o desactivado. Cada unidad está conectada con todas las demás mediante pesos sinápticos que determinan la influencia entre ellas. El modelo se puede expresar mediante una función de energía que depende de los estados y los pesos de las unidades. El modelo tiene la propiedad de que la energía siempre disminuye cuando se actualiza el estado de una unidad, lo que implica que la red tiende a alcanzar un mínimo de energía. Estos mínimos de energía representan los patrones almacenados en la memoria de la red, los cuales van a ser los atractores hacia los que evoluciona el sistema. El modelo de Hopfield se ha utilizado para estudiar fenómenos como la memoria asociativa [13].

2.4.El cerebro como una red

El cerebro es un sistema complejo conformado por diversos elementos conectados entre sí a lo largo de distintas escalas, como ya se ha visto. Por esta razón, el cerebro puede ser interpretado como una compleja red de nodos que interactúan. Es a partir de esta interpretación donde surge la neurociencia de redes. En esta disciplina se usan herramientas matemáticas y computacionales con el objetivo de analizar la información que se obtiene del cerebro, la cual es cada vez mayor y de difícil comprensión. La construcción de estas redes se basa en recoger una serie de datos relacionales (asociaciones estadísticas, redes anatómicas), con los que posteriormente construir una red constituida por una serie de nodos (elementos que la forman) y aristas (relaciones entre cada uno de estos elementos).

Una parte fundamental de la neurociencia de redes es el análisis mediante la teoría de grafos. Esta es una rama de las matemáticas que permite evaluar diferentes propiedades de las redes analizando la estructura y la dinámica de las redes neuronales a diferentes escalas. La teoría de grafos permite definir conceptos como el grado, la distancia, el camino, entre otros. Estos conceptos ayudan a caracterizar las propiedades topológicas y funcionales de las redes cerebrales, como la modularidad, la centralidad, la eficiencia, la robustez y la sincronización [5].

3. Revisión de modelos microescala

A continuación se va a hacer una revisión de los simuladores que permiten la construcción de modelos a microescala. Estos se caracterizan por poder construir con ellos redes neuronales neurona a neurona, definiendo las características del sistema a nivel micro. Se explicará las características principales de cada software, así como el planteamiento que proponen a la hora de resolver los problemas Por, último se realizará una comparación entre cada uno de ellos y se destacarán las propiedades sobresalientes de unos sobre otros Los simuladores que se analizarán son: NEST, NEURON y Brian

Uno de los principales motivos de este trabajo, como se ha mencionado en numerosas ocasiones, es el análisis de la multiescalaridad, es decir, la capacidad de cada simulador de operar a distintas escalas cerebrales, desde la neurona hasta el cerebro en su totalidad. Uno de los objetivos de este punto es revisar en la literatura el tamaño de las redes que han sido diseñadas, su significado biológico y discutir la versatilidad de los modelos.

3.1.NEST

NEural Simulation Tool (NEST) es un simulador de modelos de redes neuronales *spiking*. Esta herramienta se centra en modelar la dinámica, estructura y tamaño de los sistemas neuronales, en vez de tratar de describir la morfología y las propiedades biofísicas exactas de las neuronas individualmente. Posee más de 50 modelos neuronales diferentes y más de 10 modelos sinápticos, además cada usuario puede diseñar sus propios modelos. Está implementado en C++, pero ofrece una interfaz en Python llamada PyNest. NEST está diseñado para usar de manera óptima el hardware informático del equipo de trabajo, pudiendo ser usado tanto en un portátil para los modelos más básicos, hasta en supercomputadoras para modelos que cuentan con miles de millones de neuronas y todas las sinapsis que se producen entre estas [19] [20].

a) Redes neuronales en NEST

Una red neuronal en NEST está definida por un conjunto de nodos y sus conexiones. Los nodos pueden ser modelos neuronales, subredes o *devices* (dispositivos). Por su parte, las conexiones vienen determinadas por un nodo emisor, un nodo receptor, un peso (cómo de fuerte es la conexión) y un retraso temporal (tiempo que tarda en enviarse la señal de un nodo a otro). Los modelos sinápticos van a establecer cómo se va a ver afectada el nodo postsináptico. Los diferentes tipos existentes van a reflejar la diversidad de las redes neuronales biológicas. La información que se transmite a través de una conexión de un nodo a otro se llama evento. Estos eventos contienen el peso de la conexión y una marca temporal que determina cuando debe ser entregado al nodo receptor. Los eventos son manejados por las funciones de cada modelo neuronal. [21]

Como ya se ha mencionado, tanto las neuronas como las sinapsis van a ser definidos en NEST a través de modelos. Los modelos son implementaciones de C++ de ecuaciones matemáticas que describen características y el comportamiento de estas neuronas o sinapsis. El usuario puede elegir de entre una lista de modelos ya elaborados ofrecidos

por NEST o, si el usuario tiene conocimientos de programación, diseñar un modelo más conveniente para su estudio a través del lenguaje de modelado NESTML. NEST incluye algunos modelos neuronales son Hodgkin-Huxley o *integrate-and-fire* y modelos sinápticos como *spike-timing-dependent plasticity* (STPD). [22]

Un tipo de nodo son los llamados *devices* o dispositivos. Estos se dividen en dispositivos de registro y dispositivos de estimulación. Los dispositivos de registro son empleados para muestrear o recolectar cantidades observables como potenciales, conductancias o picos de neuronas y sinapsis (**Fig. 3.1.)**. A su vez se dividen en *collectors* (colectores) que recogen los eventos producidos por los nodos emisores y *samplers* (muestreadores). Tras haber llegado a este tipo de nodos, la información es llevada a un *record back-end* que determina si es guardada en la memoria, mostrada por pantalla o escrita en archivos. El segundo tipo de dispositivo es el de estimulación que se encargan de introducir una a la red. Existen varios dispositivos que pueden introducir una señal analógica o un tren de impulsos. Los datos que necesitan los dispositivos pueden ser guardados por ellos mismos o por una fuente de información externa.

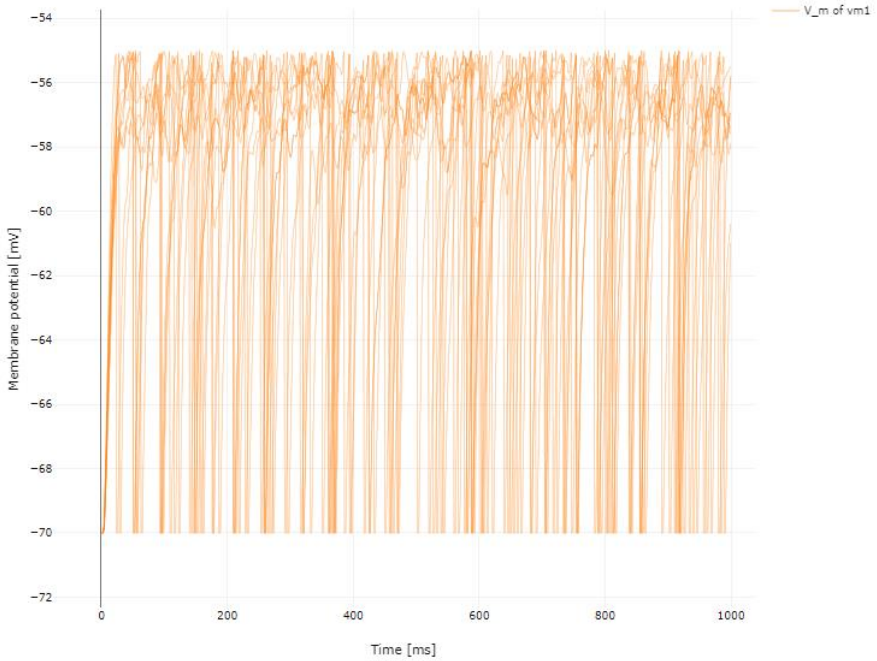


Figura 3.1. Registro del potencial de un grupo de neuronas.

Las conexiones se establecen con la función Connect() a la que hay que especificar como argumentos las poblaciones presinápticas, las poblaciones postsinápticas, la norma (*rule*) y el modelo sináptico. La norma indica de qué forma se conectan los nodos de las distintas poblaciones. Algunas de estas normas son *all-to-all* en el que cada uno de los nodos presinápticos están conectados con todos los nodos postsinápticos; o *fixed indegree* en el que se forman conexiones aleatorias, de manera que a cada nodo postsináptico le llegue un número determinado de conexiones.

NEST tiene la capacidad de generar redes neuronales espacialmente distribuidas, en donde los nodos tienen una localización concreta en la capa o *layer* y las neuronas se pueden conectar con propiedades y probabilidades dependiendo de la situación relativa

de las neuronas. Los nodos pueden ser emplazados sobre un plano o sobre un espacio tridimensional.

b) Interfaces de usuario

El lenguaje nativo de NEST es SLI (*Simulation Language Interpreter*) que interpreta los comandos de alto nivel. Este se encarga de crear y conectar redes, crear y manipular vectores y matrices, definir funciones, o del manejo de errores que pueden surgir a diferentes niveles de software. [21]

PyNEST es la interfaz de Python que sirve para interactuar con el simulador. El interpretador de Python importa NEST como un módulo que va actualizando dinámicamente el kernel del simulador. Desde un script de Python se usan funciones de alto nivel que van a generar lenguaje SLI. PyNEST permite diseñar y programar redes cerebrales y estudiarlas de una forma más sencilla que si se hiciera desde el propio lenguaje nativo, puesto que se aprovecha de la simplicidad de Python. Con el fin de evitar la existencia simultanea de dos interfaces, se optó por un enfoque que consiste en utilizar SLI como interfaz, Python envía datos y comandos SLI a NEST y NEST responde con estructuras de datos de Python (**Fig. 3.2.**). [23].

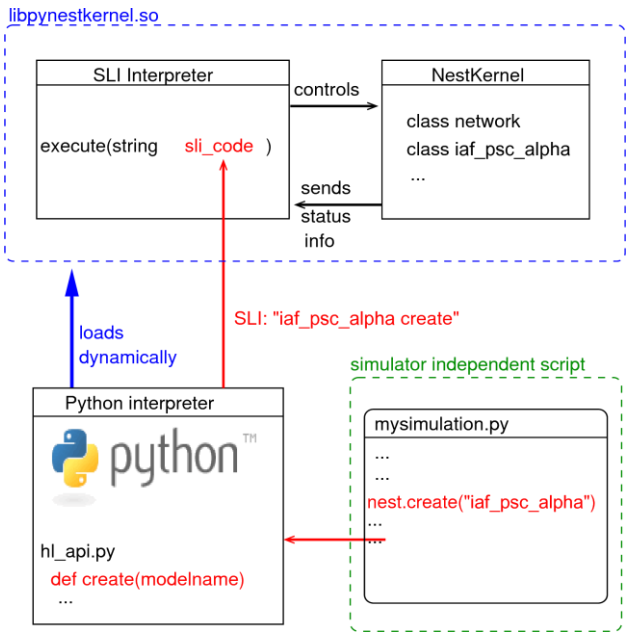


Figura 3.2. Integración Python-NEST para PyNest.

PyNN es un lenguaje independiente de simulador especializado en la elaboración y manejo de redes neuronales. Es posible elaborar un código único en PyNN y que pueda ser utilizado en los distintos simuladores que soportan el código (NEURON, NEST, Brian2) sin necesidad de realizar ninguna modificación. Esta cualidad resulta útil para poder comparar los resultados que arrojan distintos simuladores ante exactamente el mismo modelo, como se hace en [24]. PyNN permite desde la elaboración de redes de

poblaciones neuronales, capas o columnas y la comunicación entre estas, manteniendo la posibilidad de caracterizar neuronas y sinapsis individualmente con cierto nivel de detalle. [25]

NEST Desktop es una interfaz gráfica de usuario (GUI, *Graphical User Interface*) basada en la web que permite la rápida construcción, parametrización e instrumentación de modelos de redes neuronales y simular experimentos sin necesidad de tener conocimientos en programación (**Fig. 3.3.**). Emplea herramientas interactivas y elementos gráficos para un uso sencillo por parte del usuario. También tiene métodos que permiten el análisis de los resultados de la simulación de una forma visual. [26] Otra interfaz gráfica es *NEST Instrumentation App* que es usada para conectar los dispositivos de grabación y estimulación (anteriormente explicados) a las redes. Permite una adecuada visualización entre estos dispositivos y las neuronas. [27]

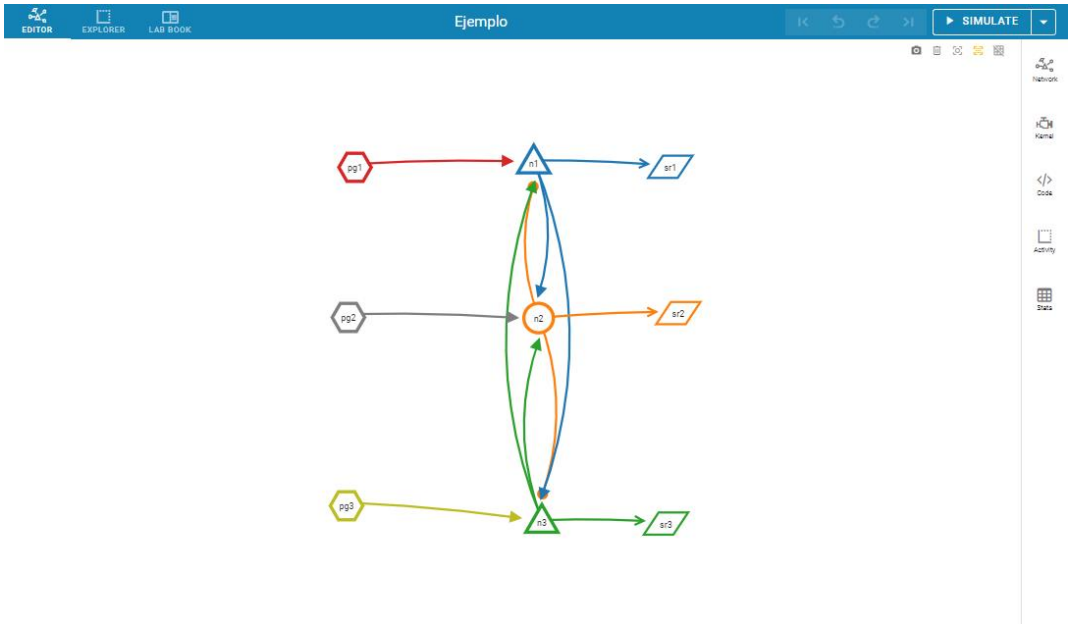


Figura 3.3. Red neuronal diseñada en NEST Desktop.

La programación de nuevos modelos neuronales o sinápticos, como se mencionó en el apartado anterior, se puede realizar a través del lenguaje de programación de NESTML. Se caracteriza por una sintaxis sencilla, una cadena de herramientas (*toolchain*) escrita en Python (PyNESTML) y buen rendimiento de la simulación mediante la generación de código en C++. Este lenguaje surge debido a la necesidad mantener un número cada vez más grande de modelos de neurociencia computacional cuya complejidad va en aumento. Aunque originalmente fue desarrollado para NEST, actualmente puede ser usado para simulaciones en distintas plataformas, tan solo cambiando un parámetro de la línea de comandos. [28]

c) Multiescalaridad en NEST

Observando la literatura, encontramos en [29] la simulación a través de NEST de una red cortical local. Este modelo representa 4 capas de la corteza cerebral, cada una compuesta por una población de células inhibitoria y otra de células excitatorias:L2/3, L4, L5 y L6. Como modelo neuronal se utiliza *spiking leaky integrate-and-fire neuron* con sinapsis estáticas. En esta investigación se trata de relacionar la conectividad estructural de la red local cortical con la actividad que surge en ella. Se concluye que, efectivamente la estructura delimita la actividad y se sugiere que esta no depende tanto de las propiedades de los distintos tipos de células. En el propio artículo se destaca la necesidad de unir la escala micro de esta red con la conectividad macroscópica. Para ello sería necesario elaborar modelos de cerebro completo para entender la interacción de la red local con otras regiones del cerebro.

En [30] se da un paso más allá y se elabora un modelo que es capaz de reproducir la dinámica en estado de reposo de la corteza cerebral del macaco encargada de la visión. Este modelo se quiere desarrollar para estudiar cómo la estructura de la conectividad cortical influye en la dinámica del sistema en múltiples escalas, desde las neuronas individuales, microcircuitos y áreas corticales; y para unificar las descripciones locales y globales de la corteza, explicando las posibles relaciones entre ellas. El modelo se puede considerar una extensión del desarrollado en el artículo anterior. También se considera un microcircuito conformado por una población inhibitoria y otra excitatoria por cada capa de la corteza cerebral, pero esta vez se esta vez se establecen 32 áreas con sus respectivas conexiones cortico-corticales, donde cada una de ellas estaría formada por dicho microcircuito de 1 mm^2.

Uno de los trabajos que resulta muy interesante es el que se lleva a cabo en [31] donde se desarrolla una arquitectura cerebro-cuerpo-entorno, que sirve para investigar cómo la percepción la actividad motora y las interacciones con el entorno constriñen la actividad cerebral y viceversa. Para ello se combina un modelo de redes de gran escala de neuronas *spiking* desarrollada en NEST que se conecta con un sistema de músculo esquelético elaborado en *Neurobiotics Platform*. La intención de este trabajo es crear una plataforma donde se pueda realizar experimentos in silico con el fin de testar las capacidades funcionales de sus modelos. El modelo de interés de este apartado es el realizado en NEST. Se hace hincapié en la necesidad de saber cómo interactúan distintas regiones del cerebro. Por ello, la red se compone de modelos de 5 regiones distintas interconectadas entre sí: corteza motora primaria, corteza sensoriomotora primaria, ganglios basales, cerebelo y tálamo. El modelo en su conjunto se compone de 1.005.905 neuronas y 1.588.469.795 sinapsis. Se muestra la capacidad de NEST de simular una red de gran escala en la que están involucradas varias regiones cerebrales que participan en la generación de movimientos.

3.2.NEURON

NEURON es un entorno de simulación para modelar neuronas individuales y redes de neuronas. Las simulaciones se caracterizan por ser eficientes en términos computacionales y por su precisión. Permite describir modelos neuronales de forma muy detallada, incluyendo propiedades anatómicas y biofísicas complejas. Entre sus cualidades, destacan [32]:

- Ofrece una sintaxis natural, empleando conceptos neurocientíficos de uso común para los investigadores, como secciones, canales iónicos, sinapsis, etc. Estos pueden ser usados para desarrollar modelos, de forma que el usuario pueda trabajar con términos que le resulten cercanos.

- Permite al usuario elegir entre diferentes métodos de integración numérica, pudiendo cambiar estos sin tener que reescribir el modelo.

- Proporciona herramientas gráficas e interactivas para crear y modificar modelos, controlar simulaciones y presentar los resultados de forma visual e intuitiva.

- Soporta la programación en hoc o Python, lo que ofrece una gran flexibilidad para definir la apariencia de la interfaz gráfica, las propiedades anatómicas y biofísicas de los modelos, y el flujo de control de las simulaciones.

- Permite al usuario definir mecanismos biológicos personalizados mediante el lenguaje NMODL o la GUI *ChannelBuilder*, que se compilan para lograr una mayor eficiencia computacional.

- Se puede aplicar a modelos de red que contienen un gran número de células y conexiones. Además, se puede aprovechar la arquitectura paralela de multiprocesadores o *clusters* para acelerar las simulaciones.

- Tiene una amplia documentación, soporte y comunidad de usuarios, así como cursos y seminarios periódicos para aprender a usarlo. También tiene una gran cantidad de modelos publicados que se pueden consultar y descargar.

a) Redes neuronales

NEURON permite crear modelos a diferentes escalas, desde una neurona individual hasta redes de estas ensambladas, bien mediante programación o mediante el uso de las distintas GUI que ofrece, o con la combinación de ambas. La creación de todos los modelos neuronales y las redes neuronales que se van a comentar, se encuentran explicados con detalle en la documentación [33].

En los modelos neuronales se puede indicar la morfología de la neurona a través de la definición de parámetros muy específicos (longitud, diámetro), y otras consideraciones biofísicas, por ejemplo, cómo afecta la actividad de los canales iónicos a las

concentraciones citosólicas. Estos modelos de neuronas individuales pueden ser creados a través de la GUI *Cell Builder*.

Una herramienta disponible es Import3D, que permite convertir datos morfométricos celulares en las propiedades que usará *CellBuilder* para crear el modelo. Además, puede analizar la información, corregir errores comunes y avisar de problemas complejos que requieren el criterio del usuario. Finalmente, puede arrojar una figura interactiva que muestra la forma de la célula y muestra distintas características de las secciones seleccionadas.

La construcción de redes puede realizarse directamente en código hoc o mediante *Network Builder*, que es capaz crear un archivo en hoc. *Network Builder* puede ser usado para microcircuitos que sean usados como plantillas a la hora de construir redes de mayor escala. Independientemente de la opción escogida, la creación de la red se comprende de tres pasos: definir el tipo de células, crear cada célula en la red y conectar estas células.

Los tipos celulares pueden ser o biofísicas realistas que requieren integración numérica para su solución, o artificiales que son más simples, se pueden resolver analíticamente y son más eficientes computacionalmente. Ambos tipos pueden estar presentes en una misma red. Dentro de estos tipos se pueden crear distintos modelos con diferentes características. Posteriormente, se colocan las células en la red y se establecen las conexiones entre estas, especificando el valor de distintos parámetros como los pesos o los retrasos temporales. Dichas interacciones pueden ser del tipo: sináptico, *gap junctions* o efápticas (conducción eléctrica a través del espacio extracelular).

b) NETPyNE

NetPyNE (*Networks using Python and NEURON*) es una herramienta de software, un paquete de Python, que facilita la construcción, simulación, optimización y análisis de modelos de redes neuronales multiescala en NEURON. NetPyNE permite al usuario definir las propiedades del modelo a diferentes escalas, desde molecular a circuito, mediante un lenguaje declarativo estandarizado y fácil de usar. NetPyNE también proporciona una interfaz gráfica de usuario (**Fig. 3.4.**), formatos de entrada/salida estandarizados, visualización y análisis de datos integrados, y ejecución paralela en computadoras o *clusters* [34].

NetPyNE permite a usuarios sin experiencia en programación poder construir complejos modelos en NEURON, gracias a la GUI de uso intuitivo o al uso del lenguaje declarativo. Mediante esta se puede especificar los parámetros del modelo que queda completamente separado de la implementación del código, explorar la red y analizar la simulación.

El flujo de trabajo en NETPyNE consiste en:

1. Especificación de alto nivel: definición de parámetros de la red, input introducido por el usuario.

2. Instanciación de la red: generación de un modelo de NEURON que contiene todos los requisitos indicados en el paso anterior y se representa como una estructura jerárquica de Python.

3. Simulación: se ejecuta la simulación en paralelo en NEURON, quedando ocultas las complejidades técnicas para el usuario.

4. Análisis: se dispone de varias funciones que muestran gráficas y figuras que reflejan las características de la red y los resultados de la simulación.

Una de las funciones que NETPyNE puede realizar es la exploración y optimización de parámetros que producen un comportamiento deseado en un modelo de red neuronal. El usuario puede definir un rango de valores para cada parámetro y usar algoritmos de optimización o exploración para encontrar los mejores ajustes. NetPyNE permite ejecutar múltiples simulaciones en paralelo usando supercomputadoras y analizar los resultados usando diferentes medidas de error o métricas de información.

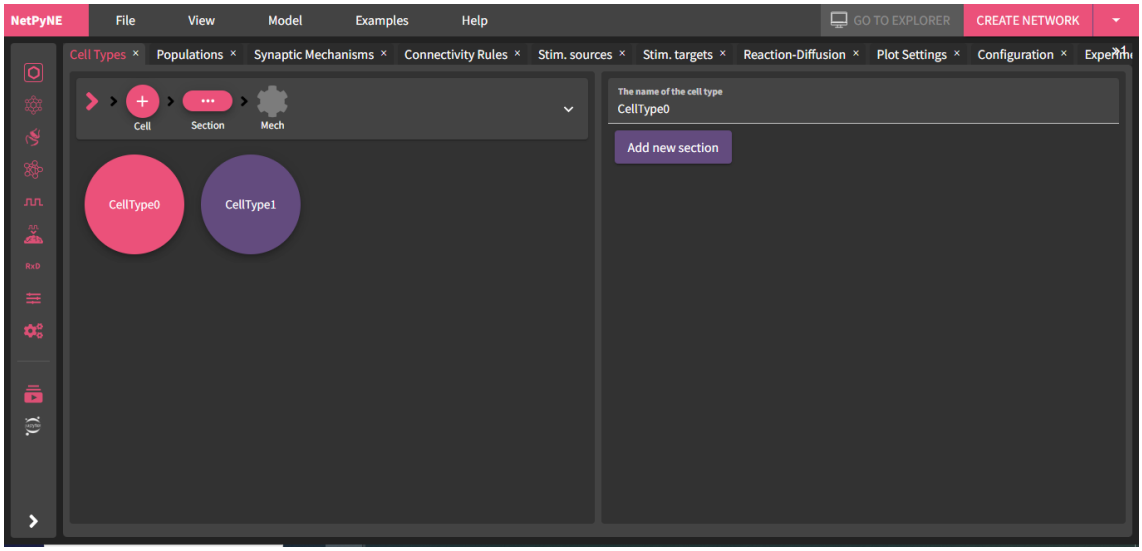


Figura 3.4. NetPyNe GUI.

c) Multiescalaridad

Anteriormente se ha visto cómo, para la construcción de las redes neuronales, primero se especifica las características de las neuronas y, posteriormente se va subiendo el nivel de las escalas añadiendo los detalles de cada una de estas, hasta crear una red neuronal de la complejidad requerida. Además, se ha incorporado a NEURON el módulo de Python Reacción-Difusión (RxD) [35] que modela cómo el transporte y las reacciones de especies químicas dentro y fuera de las células pueden afectar a la electrofisiología celular, interactuando con los canales iónicos y los potenciales eléctricos de las células. Mediante este módulo se añade una nuevo nivel escalar subcelular.

Se debe destacar la facilidad que otorga NetPyNE a la hora de incorporar los datos de las distintas escalas de una forma precisa por medio de su lenguaje declarativo, a la vez que

oculta los complejos procesos técnicos que tienen lugar para su implementación en NEURON [34]. La flexibilidad que presenta permite crear poblaciones celulares heterogéneas y con modelos de distinta complejidad, que puede resultar útil a la hora de simplificar redes multiescalares de gran tamaño computacional. Además, incorpora distintas funciones de conectividad y soporta el módulo RxD.

En [36] se hace hincapié en la importancia de la simulación de redes neuronales de gran tamaño a diferentes escalas y evalúa la capacidad del procesamiento en paralelo de NEURON, realizando las simulaciones en *clusters* de computadoras de alto rendimiento de tamaño medio. Se realizaron simulaciones de redes de diferentes tamaños (entre 500 y 100.000 células) de hasta tres tipos diferentes: compuestas por modelos neuronales *Izhikevich integrate-and-fire neurons*, modelos Hodgkin-Huxley (HH) y ambos modelos en una misma red. Se resalta la importancia de diseñar modelos en los que queden claramente separados las diferentes escalas, junto con los parámetros que caracterizan cada una de ellas. Concluye que NEURON tiene la capacidad de abordar este tipo de problemas multiescalares.

3.3.Brian

Brian es un simulador gratuito de código abierto escrito en Python que permite emular modelos de redes neuronales formadas por neuronas *spiking* de forma sencilla. Uno de los objetivos de Brian es tratar de resolver la dicotomía entre flexibilidad y rendimiento, es decir, entre la capacidad de definir fácilmente modelos y experimentos computacionales novedosos y la velocidad y eficiencia con la que se ejecutan las simulaciones. Para lograr este balance, Brian usa la técnica de generación de código, que consiste en transformar automáticamente una descripción de alto nivel del modelo, escrita en Python y en forma matemática, en código ejecutable en un lenguaje de bajo nivel, como C++, que se compila y se ejecuta sin requerir ninguna acción del usuario. Este código generado se inserta dentro del flujo del script de simulación, lo que lo hace compatible con el enfoque procedural y aumenta su eficiencia computacional. Esta técnica es el modo fundamental de operación de Brian 2, la última versión de este simulador que será explicada en este texto, frente al poco uso que se le daba en Brian 1. La generación de código no solo se usa para ejecutar los modelos, sino también para construirlos, y por lo tanto también acelera etapas como la creación de sinapsis. El marco de generación de código ha sido diseñado para ser extensible en varios niveles [37].

Brian se caracteriza por dos principios básicos: está basado en ecuaciones y resalta la necesidad de realizar experimentos computacionales.

La simulación basada en ecuaciones se refiere a que los modelos neuronales y sinápticos están descritos por ecuaciones matemáticas que se incorporan al propio script. Brian requiere la escritura de las ecuaciones explícitamente, en vez de utilizar modelos predefinidos como se hace en otros simuladores. Esto permite que el investigador conozca adecuadamente qué es lo que hace su código, con lo que se pretende evitar posibles discrepancias entre modelos publicados en artículos e implementaciones de estos realizadas por otros usuarios.

Una característica distintiva de Brian es su capacidad para incorporar fácilmente las peculiaridades más relevantes de los modelos en el código. Esto significa que los detalles específicos de los modelos neuronales y sinápticos pueden ser fácilmente expresados, lo que facilita la personalización y adaptación a las necesidades de investigación. Esto aporta una gran flexibilidad a Brian, algo que no se conseguiría con modelos preestablecidos. Además, Brian ofrece la misma expresividad tanto para los modelos sinápticos como para los neuronales. Esto significa que los investigadores pueden describir tanto las propiedades de las conexiones sinápticas como el comportamiento de las neuronas utilizando la misma sintaxis y estructura en el código. Otra propiedad de Brian es que en los propios modelos se puede incluir elementos no neuronales, como puede ser la actividad de unos músculos activados por motoneuronas.

El segundo aspecto fundamental de Brian es la posibilidad de incorporar en el mismo script la definición de modelos con la descripción de experimentos computacionales, dando la posibilidad de implementar protocolos de simulación. Por otro lado, Brian también permite escribir el protocolo experimental en Python, usando estructuras de control como bucles o condicionales para especificar la secuencia y las condiciones de las simulaciones. Esto permite implementar una gran variedad de protocolos que no podrían ser capturados por un formato puramente descriptivo. Además, Brian ofrece la posibilidad de almacenar y restaurar el estado del modelo entre simulaciones, lo que facilita la realización de experimentos que requieren modificar o comparar algún aspecto del modelo.

El uso de Python como lenguaje de programación por parte Brian, le permite adquirir una serie de ventajas. Python es un lenguaje de alto nivel que suele ser descrito como "pseudocódigo ejecutable". Esto es importante porque, por norma general, la comunidad científica que trabaja en este ámbito no suele tener unos conocimientos muy extensos de programación, por lo que es necesario la utilización de un lenguaje fácil de aprender. El código que se genera en Brian se caracteriza por ser legible para humanos, tanto para los modelos, permitiendo escribir las ecuaciones diferenciales que definen el modelo directamente en su forma matemática; como para el experimento computacional que se realiza. Esto facilita la comprensión, la reproducción y la modificación de los modelos y experimentos, así como la comunicación entre los investigadores.

Sin embargo, a pesar de todas sus ventajas, Brian presenta ciertas limitaciones. La primera es la incapacidad de ejecutar redes de grandes dimensiones en supercomputadoras, aunque este tipo de experimentos no son tan comunes como ejecutar varias veces una red más pequeña variando sus parámetros. La segunda desventaja es que Brian se centra en modelos neuronales puntuales o de un solo compartimento, en los que una neurona se ve representada sin ningún tipo de división interna en dendritas, soma y axón. A pesar de que también se pueden utilizar modelos multicompartimentales basado en ecuaciones, esta característica no está tan bien especializada como en otros simuladores.

a) Redes neuronales

A continuación, se va a exponer cómo se construirían las redes neuronales en Brian, a partir de los visto en la documentación [38]. Lo primero que se debe tener en cuenta es que los modelos que se describen en Brian son sistemas físicos, por lo que las variables que los conforman poseen unidades físicas que se debe tener en cuenta para evitar incoherencias en dicho modelo. La creación de redes neuronales en este simulador se basa en implementar grupos neuronales a través del comando "*NeuronGroup()*" que posee varios argumentos de entrada, siendo los dos primeros: el número de neuronas que conforman dicho grupo y las ecuaciones que definen el modelo. Estas ecuaciones deben ser establecidas previamente como "*string*". Para ejecutar el programa se utiliza el comando run(), al que se le debe indicar el tiempo de simulación.

Si se pretende elaborar modelos más complejos, "*NeuronGroup()*" tiene más argumentos de entrada, como *threshold* y *reset* que sirven para crear poblaciones de neuronas *spiking*, si la variable supera el valor umbral, su valor retorna al punto dado por *reset*. También, se puede especificar la refractariedad, tiempo en el que la neurona no puede enviar un impulso; y el método de integración numérica. En el caso que en un grupo neuronal se establezcan varias neuronas, todas ellas van a estar determinadas por el mismo conjunto de ecuaciones, pero cada una de ellas se puede inicializar y tener parámetros con distintos valores.

El siguiente paso sería establecer la sinapsis entre las neuronas. Un modelo sináptico puede ser definido para varias neuronas de una misma población. Se utiliza la función "*Synapses"* en la que se debe introducir como argumentos de entrada la población fuente y la población receptora. Además, se debe incluir el efecto que ocurre, por ejemplo si se quiere que cuando se produce un impulso en la neurona presináptica se produzca un cambio en la neurona postsináptica, se especifica como: *on_pre='v_post += 0.2*'. Además, se puede incluir pesos y retrasos temporales. Brian también ofrece la posibilidad de crear modelos mucho más complejos como *short-term plasticity* (STP) or *spike-timing dependent plasticity* (STDP).

Una vez definido el modelo sináptico, a través del comando *"S.connect"* se establece cuáles son las neuronas que se conectan. Para redes pequeñas se puede establecer directamente que neurona se conecta a cual, aunque es mejor utilizar una condición y una probabilidad de conexión. Por ejemplo, *"S.connect(condition='i!=j', p=0.2)"* indica que una neurona se puede conectar con otra que no sea ella misma con una probabilidad del 0.2. Para redes más grandes, se puede utilizar una expresión matemática para especificar directamente las proyecciones de cada neurona.

b) Multiescalaridad

Una de las principales desventajas de este simulador, como ya se ha comentado, es que no es capaz de emular redes neuronales de gran tamaño, por lo que su extensibilidad para el estudio de escalas más grandes a partir de la microescala se va a ver reducido. A

continuación, se presentan algunas simulaciones encontradas en la literatura, en las que se puede comprobar que este simulador es útil para representar redes más pequeñas.

La primera simulación que se realiza en [37] presenta un modelo simplificado de la red pilórica del ganglio estomatogástrico de los crustáceo. Esta red está formada por tres tipos de neuronas que son: neuronas de estallido anterior (AB), dilatador pilórico (PD), pilórico lateral (LP) y pilórico (PY) (la combinación de AB y PD se considera un único tipo: AB/PD). Estas generan un patrón motor trifásico y controlan el movimiento de los músculos del estómago. El modelo usa ecuaciones diferenciales para describir la dinámica de las neuronas y sus conexiones sinápticas, que son graduadas y dependen del calcio. El simulador Brian permite implementar este modelo de forma sencilla y directa, usando un lenguaje matemático y expresivo. El modelo reproduce el patrón de actividad observado experimentalmente y muestra cómo la regulación del calcio modula la frecuencia y la fase de las neuronas.

La segunda simulación describe un modelo de movimientos oculares suaves, donde el ojo sigue un objeto que se mueve frente a él. El modelo incluye neuronas sensoriales y motoras, músculos oculares y la dinámica del estímulo. Este es un claro ejemplo de la incorporación de elemento no neuronales que se puede hacer en Brian, pero que en otros simuladores resultaría más complicado. El muestra cómo el ojo puede seguir el objeto con un mecanismo de retroalimentación simple, basado en la excentricidad de la posición del objeto en la retina.

En [39] se muestra cómo Brian puede ser una herramienta eficaz a la hora de modelar la fisiología de las células de la glía. Se pone como ejemplo la simulación de un modelo en el que se describe una población de astrocitos y como estos pueden afectar a la actividad neuronal que se produce entre un grupo de neuronas conectadas mediante sinapsis. Debido a que la mayoría de los modelos ignoran cómo la glía puede interactuar con las neuronas, debido a su flexibilidad, Brian permite establecer nuevos modelos que sí tengan en cuenta estos elementos.

3.4.Comparación de simuladores microescalares

En [40] se realiza una detallada comparativa entre los distintos simuladores que se han descrito anteriormente. Se evalúan varias características como el rango de modelos que pueden aceptar, la arquitectura y eficiencia computacional. Además, se realizan dos experimentos, utilizando los mismos modelos adaptados para cada simulador, y se compararon los resultados de cada uno.

Respecto a los tipos de modelos que soportan, como se explicará a continuación, NEURON es el que admite mayor diversidad de modelos. Todos ellos son capaces de ejecutar los modelos más simples como los no-dinámicos, dinámica discontinua, dinámica continua, modelos de un compartimento o puntuales y modelos de dos compartimentos. En cuanto a la incorporación de nuevos canales en el modelo, NEURON permite hacerlo desarrollando un módulo, mientras que NEST y Brian necesitan modificar el modelo neuronal completo, lo que supone una desventaja al requerir ciertos conocimientos en programación.

En cuanto a modelos más complejos, como los que reconstruyen la morfología neuronal y la distribución de los canales y las dendritas, en NEURON soporta de forma predeterminada, sin necesidad de desarrollar módulos externos o modificar el código fuente. En Brian se puede realizar definiendo objetos que representan distintas partes de la neurona. El simulador que puede resultar más complejo es NEST, donde es necesario tener un buen nivel a la hora de programar y de trabajar con métodos numéricos.

Otro tipo de modelos como los de reacción-difusión intracelular son completamente soportados por NEURON, mientras que en NEST se debe crear un módulo específico, y en Brian debe ser modelado añadiendo variables dinámicas a las ecuaciones, complicando el proceso. En cuanto a la estimulación extracelular que puede representar ciertas patologías, tan solo es admitida por NEURON, necesitando en Brian extensos conocimientos en neurociencia, resultando excesivamente complicado en NEST.

En cuanto a la arquitectura y eficiencia computacional, se puede destacar las siguientes características de cada uno de estos softwares:

- NEURON: Tiene la arquitectura más flexible y modular, que permite describir neuronas y redes con diferentes niveles de detalle y complejidad. Tiene una buena eficiencia computacional, ya que compila los módulos en código binario y utiliza métodos numéricos avanzados. Tiene herramientas para la paralelización de modelos en *clusters* o computadores *multicore*, pero requiere modificar el código si el modelo se desarrolló para un solo computador.

- NEST: Tiene la arquitectura más orientada a redes de gran escala con neuronas simples, que se describen mediante módulos en C++. Tiene una alta eficiencia computacional, ya que utiliza una biblioteca científica (GNU) y un solucionador Runge-Kutta para las ecuaciones diferenciales. Tiene herramientas para la paralelización transparente de modelos en *clusters* o computadores *multicore*, sin necesidad de modificar el código, lo que supone su mayor ventaja.

- BRIAN: Tiene la arquitectura más sencilla y concisa, que permite describir neuronas y redes mediante expresiones matemáticas en Python. Tiene una eficiencia computacional variable, dependiendo del método numérico elegido y del régimen de compilación. No tiene herramientas para la paralelización en *clusters*, y tiene un soporte limitado para computadores *multicore*.

Respecto a los experimentos que se llevaron a cabo, se llegó a la conclusión que NEURON es la mejor capacitada para modelar redes de pequeño tamaño, pero con modelos muy detallistas, mientras que NEST trabaja mejor con redes de mayor tamaño, pero con modelos neuronales más simplificados.

3.5. Experimentos realizados con simuladores microescalares

a) Materiales y métodos

A continuación, se va a describir las diferentes pruebas y experimentos que fueron realizados con estos softwares, con el objetivo de realizar una comparación entre ellos. Para el diseño de los modelos se hizo un extensivo uso de las GUI de NEST y de NEURON NetPyNe, con el fin de visualizar las capacidades de cada una y poder evaluar cuáles son sus posibilidades, estableciendo hasta que nivel de complejidad podría tener una red neuronal diseñada por un investigador con escasos conocimientos en programación.

El primer caso fue obtenido de [40] y consiste en dos poblaciones, una de células excitatorias (80% del total) y otra de células inhibitorias (20%), que van a ser modeladas como *leaky-integrate-and-fire*. Además, también se cuenta con un conjunto de generadores de impulsos del tipo Poisson. Cada población tiene ciertas probabilidades de conexión, así como un peso y un retraso asignado en estas. La realización de este caso permitirá estimar la incorporación de modelos neuronales simples de células puntuales en cada uno de los softwares.

En NEST Desktop se van a incorporar los distintos nodos directamente sobre el editor, pudiendo visualizar dinámicamente la red que se está creando, mientras que en NetPyNe GUI, se especifican los parámetros de la red y posteriormente se crea dicha red que va a poder ser visualizada gráficamente. Para NEST se usó el modelo ***'iaf psc delta'***. En NetPyNe-UI fue utilizado el modelo celular ***'IntFire2'*.**

En el segundo caso, también obtenido de [40], se presenta una población de 400 neuronas modeladas mediante Hodgkin–Huxley (HH), un modelo más complejo que el anterior. Cada neurona se conecta de forma aleatoria con 40 neuronas de la propia población. En NEST Desktop se utilizó el modelo '***HH psc alfa',*** que permite incluir los distintos parámetros de HH. En NetPyNe-UI se modeló un tipo de célula consistente únicamente de un compartimento somático que incluye el mecanismo '***HH*'**, en el que son introducidos los parámetros.

Además, se va a evaluar la posibilidad de distribuir las neuronas del modelo en el espacio y hasta qué punto se permite especificar la posición de estas en cada uno de los softwares. Para ello, se emplearán las herramientas disponibles en las GUI y se evaluará si es posible disponer distintos tipos neuronales en determinadas capas superpuestas.

También se pretende explorar la capacidad de crear modelos compartimentales. En NetPyNe-UI existe la posibilidad de especificar de forma muy detallada la geometría de cada uno de los compartimentos, pudiendo crear modelos de un alto nivel de realismo biológico.

Por último, se explorará el módulo RxD de NetPyNe-UI. Esta GUI cuenta con un apartado específico para este módulo y desde allí se pueden especificar los valores para distintos parámetros. El modelo evaluado consiste en tres capas corticales (2, 4 y 5), cada una con una población inhibitoria y otra excitatoria. Las células piramidales del modelo siguen la siguiente simulación: ante un incremento de IP3 en el citosol, sus

receptores del retículo endoplasmático se abren, liberando calcio al citosol. Los canales de potasio dependientes de calcio se abren y el potasio se introduce en la célula, produciendo una hiperpolarización y produciéndose el disparo.

b) Resultados obtenidos

En el caso 1, el diseño de las redes en ambos softwares se muestra en **Fig. 3.5.** En **Fig. 3.5.(A)** podemos observar que en NEST Desktop se puede obtener una visión general de la red, distinguiendo los dos tipos de poblaciones existentes y cómo interactúan entre ellas. En NetPyNe-UI, se puede ver una representación tridimensional de la red, aunque en este caso al ser modelos neuronales puntuales no aporta información relevante. Además, se puede observar cómo se disponen las distintas neuronas en el plano **Fig. 3.5. (C)**, distinguiendo las dos poblaciones y la forma en la que están conectadas a través de la matriz de conexión **Fig. 3.5. (B)**. Aunque NetPyNe-UI nos da una información más específica que puede resultar muy útil en casos en los que la red es más compleja, NEST Desktop nos muestra de una forma más simplificada, obteniendo a simple vista la información general de cómo están conectadas las distintas poblaciones.

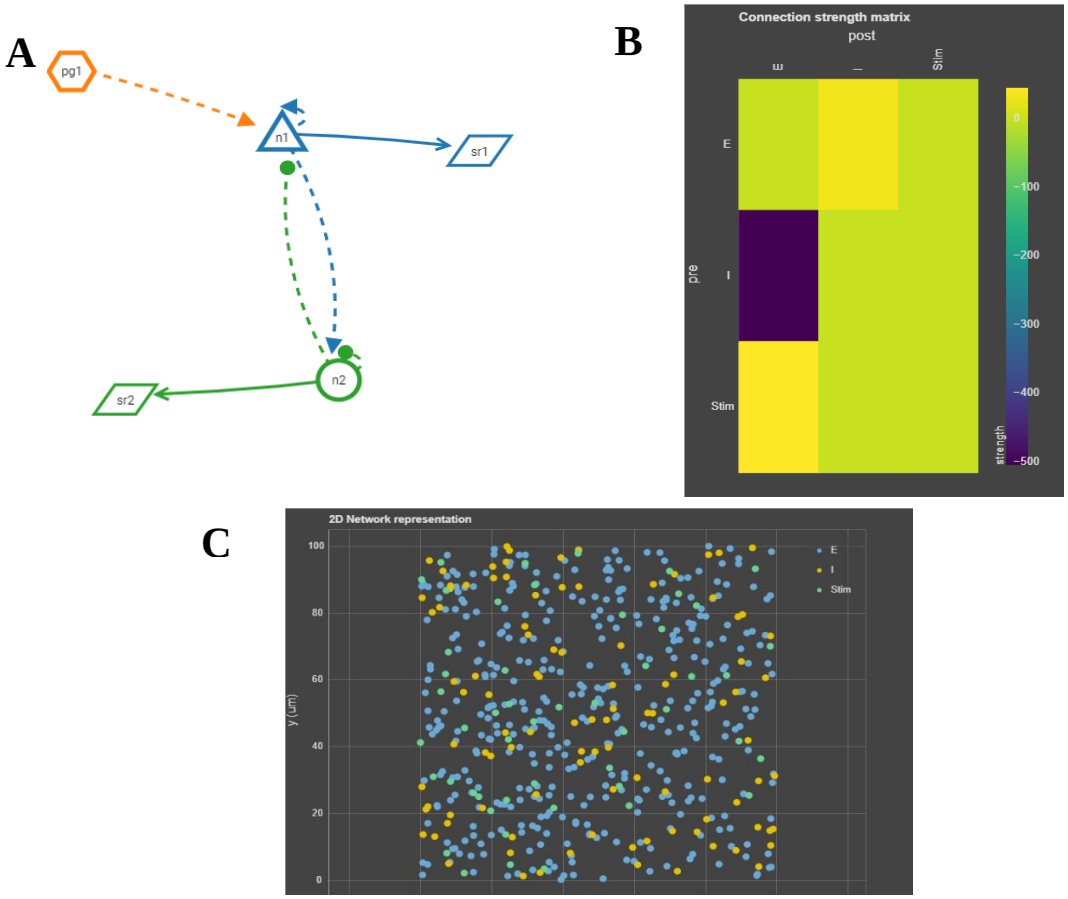


Figura 3.5. Visualización de la red. **(A)** Red de nodos en NEST Desktop, siendo el hexágono naranja el estimulador, el triángulo azul y el círculo verde las poblaciones excitatorias e inhibitorias respectivamente, los paralelogramos los recorders y las líneas las conexiones, de pico triangular excitatorias, pico circular inhibitorias y líneas discontinúas asociadas a una probabilidad de conexión. **(B)** Matriz de conectividad de NetPyNe-UI. **(C)** Distribución de las neuronas en el espacio en NestPyNe-UI, siendo azules excitatorias, naranjas inhibitorias y verdes estimuladores.

En cuanto a los resultados del caso 1, al ser modelos de *spiking neurons*, en los dos simuladores se puede obtener los impulsos de cada una de las células, tanto el histograma con el número de impulsos en un determinado tiempo, como el tiempo en el que se dispara cada célula (**Fig. 3.6.**).

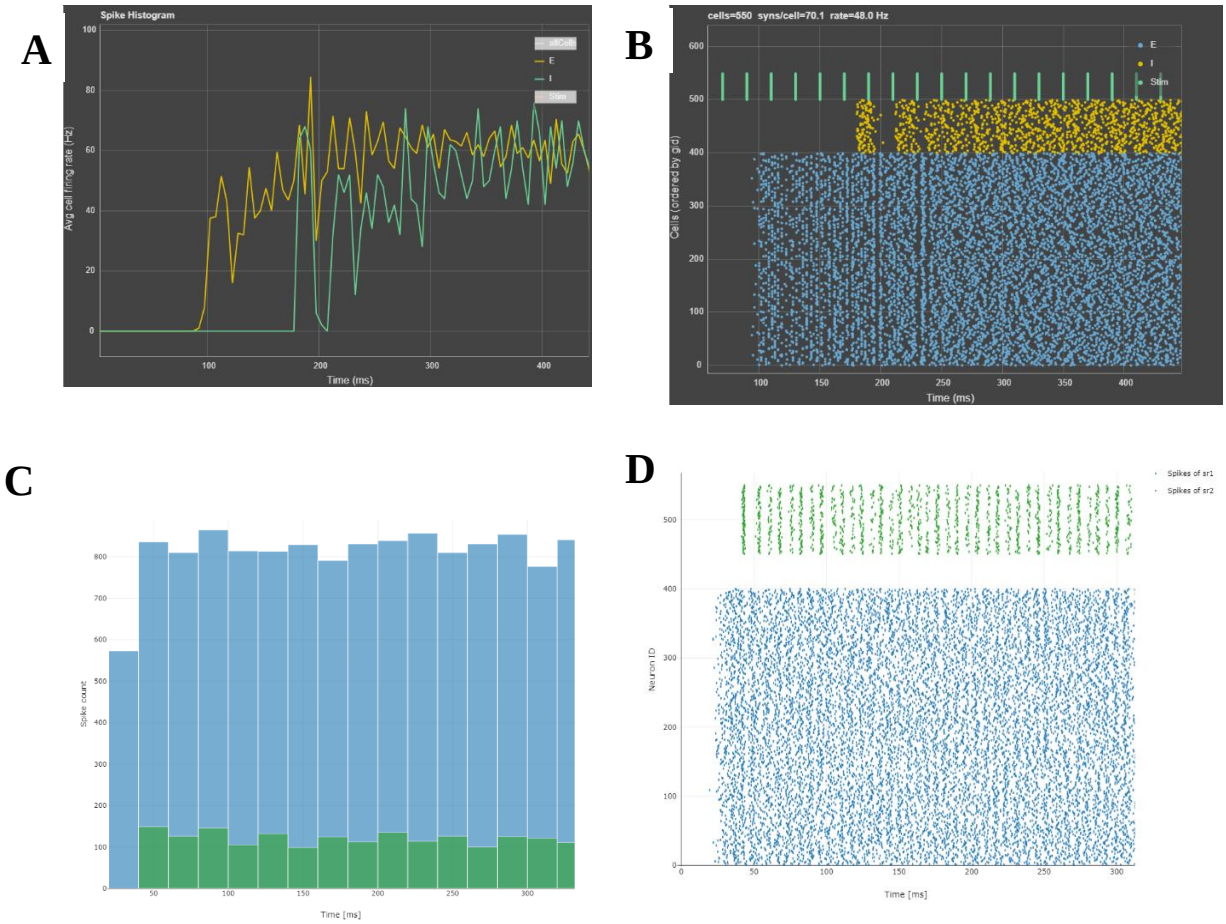


Figura 3.6. Caso 1. Histograma de número de impulsos a lo largo del tiempo en **(A)** NetPyNe-UI **(C)** NEST Desktop. Impulsos producidos por cada una de las neuronas de cada población a lo largo del tiempo en **(B)** NestPyNe-UI, siendo azules excitatorias, naranjas inhibitorias y verdes estimuladores y **(D)** NEST Desktop, siendo azul excitatorias y verde inhibitorias.

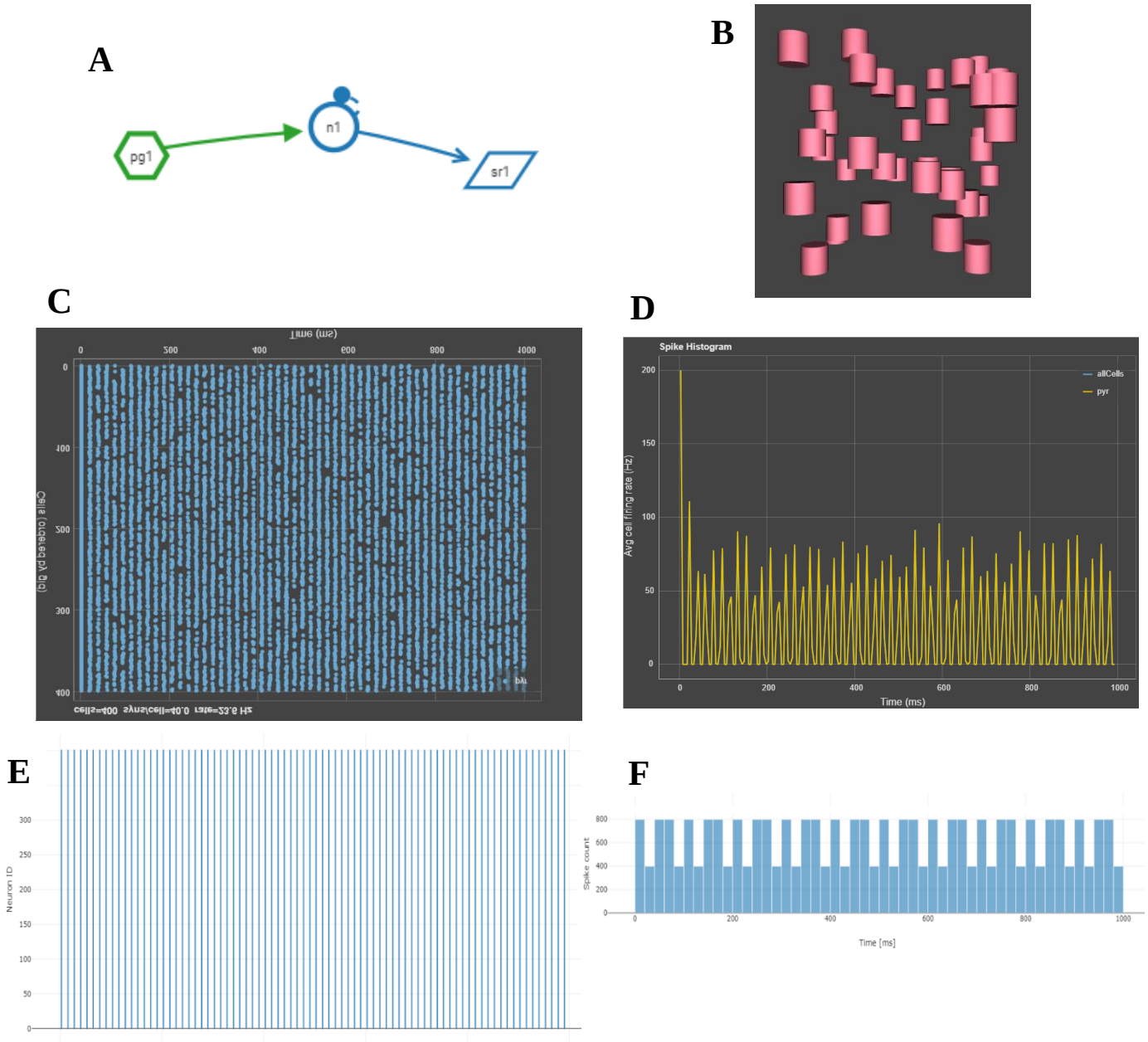


Figura 3.7. Caso 2. **(A)** Red de nodos en NEST Desktop siendo hexágono verde el estimulador, circulo azul población inhibitoria y paralelogramo el recorder **(B)** Representación tridimensional de la distribución de las neuronas en el espacio en NetPyNe-UI. Impulsos producidos por cada una de las neuronas de cada población a lo largo del tiempo en **(C)** NetPyNe-UI y **(E)** NEST Desktop. Impulsos producidos por cada una de las neuronas de cada población a lo largo del tiempo en **(D)** NetPyNe-UI y **(F)** NEST Desktop.

Los resultados del caso 2 se muestran en **Fig. 3.7.** se puede observar el modelo de red de NEST **(Fig. 3.7.(A))** y la representación tridimensional de la red en NetPyNe-UI, aunque al ser modelos sencillos no aporta demasiada información. Al igual que en el anterior caso, se puede observar el número de impulsos en un determinado intervalo de tiempo y cuando se realizan dichos impulsos.

Las GUI de ambos permite situar las poblaciones neuronales en un cierto lugar del espacio. Basado en [29], donde se dispone un modelo de las distintas capas de la corteza cerebral, caracterizada cada una de ellas por un cierto grosor, se puede simular un modelo como ese en NetPyNe-UI (**Fig. 3.8. (A)**). Para cada par de población excitatoria e inhibitoria, se puede especificar la distribución espacial en el eje Y, en este caso, y así se puede obtener una población de cada tipo en cada una de las capas. Por otro lado, en NEST Desktop, se pueden distribuir las distintas neuronas en una rejilla, donde se establece el punto central de la población y el número de columnas y filas. En la **Fig. 3.8. (B)** se muestra un modelo constituido por dos poblaciones excitatorias y otra inhibitoria, cada una sitúa en un lugar específico del espacio. Respecto a esta plataforma, cabe destacar que puede presentar la estructura espacio-temporal del modelo a través de grafos animados que representan cuando una neurona está activada.

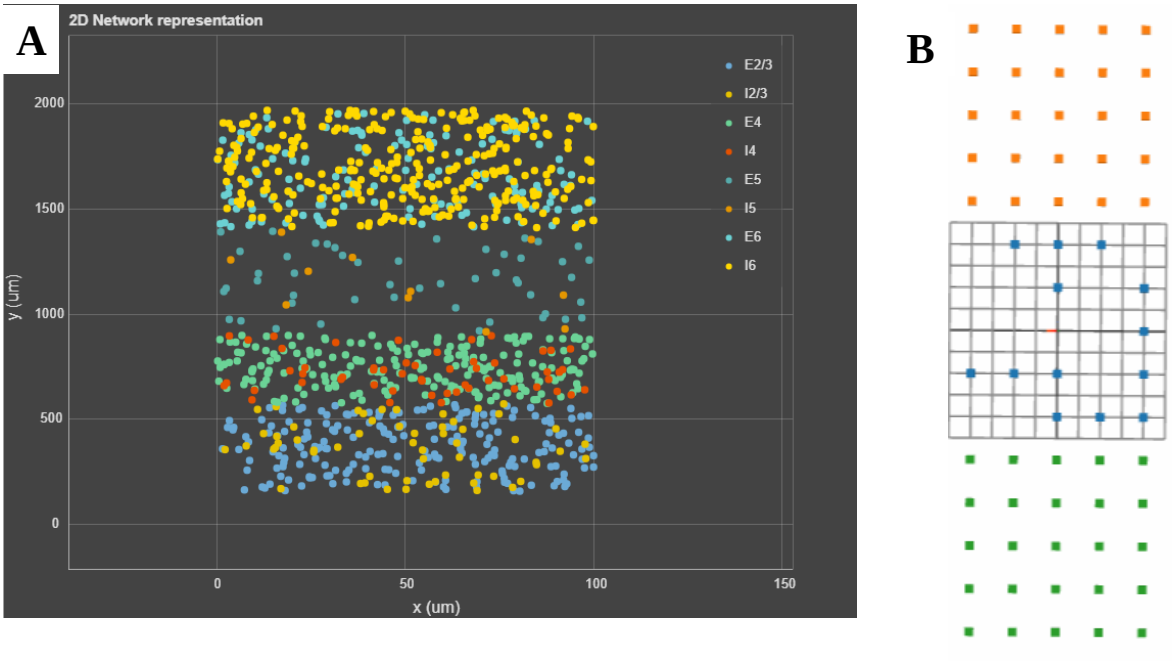


Figura 3.8. Disposición espacial. **(A)** Distribución de las neuronas en NetPyNe-UI a lo largo de cuatro capas de la corteza cerebral. Cada capa se compone de dos poblaciones neuronales (excitatoria e inhibitoria) y están enmarcadas en un rango concreto del espacio a lo largo del eje y. **(B)** Distribución espacial de tres poblaciones neuronales en NEST Desktop. En un tiempo concreto se puede observar qué neuronas están ejerciendo un potencial de acción.

La simulación de modelos compartimentales es una tarea para la que NetPyNe-UI está plenamente preparada. A la hora de definir un tipo de células se puede especificar las distintas secciones (soma, número de dendritas, axón…) y su respectiva morfología, permitiendo elaborar modelos neuronales muy complejos, como se muestra en **Fig. 3.9.**, siendo esta la mayor fortaleza de este software.

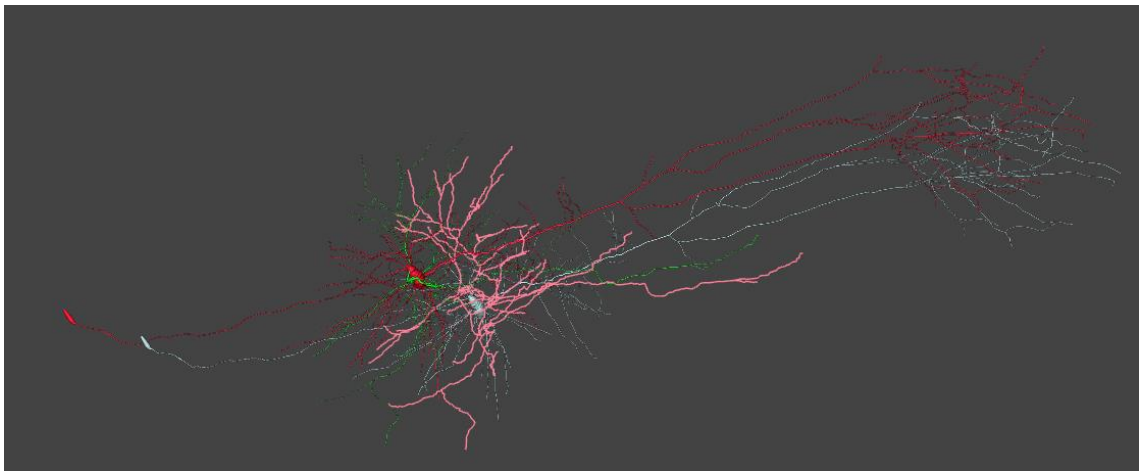
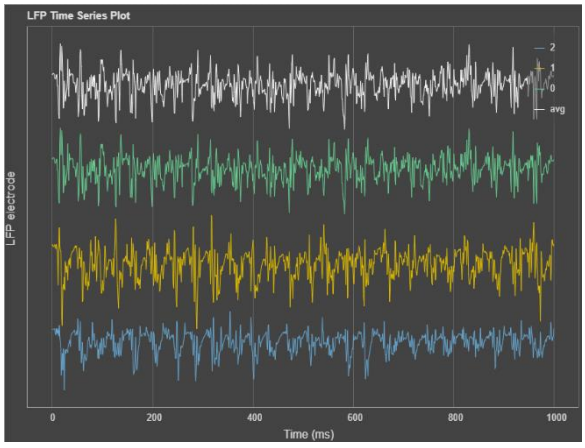


Figura 3.9. Representación tridimensional de cuatro neuronas, cada una de un color, con una morfología detallada y compleja.

Por su parte, NEST Desktop también permite definir distintos compartimentos para la neurona con el uso del modelo neuronal **cm_default,** aunque no de una forma tan detallada como el caso anterior. En este modelo, se puede incluir en cada uno de esos compartimentos receptores del tipo AMPA, GABA y NMDA. De esta forma, también se puede abordar una escala menor, indicando como pueden participar los iones de sodio y potasio a través de estos receptores. Por su lado, para abordar esta capa subcelular NetPyNe-UI cuenta con el módulo RxD.

A continuación se explicará un ejemplo en el que se utiliza el módulo RxD y se puede apreciar cómo interactúan las distintas escalas, desde la subcelular hasta el potencial de campo local, el cual puede ser grabado por NetPyNe-UI especificando la localización virtual de un electrodo. En **Fig. 3.10.**, se puede apreciar cómo cuando se dispone de una mayor concentración inicial de IP3, la actividad se reduce, en este caso a partir de los 500 ms (B) en comparación cuando la concentración inicial es menor (A).

A



B

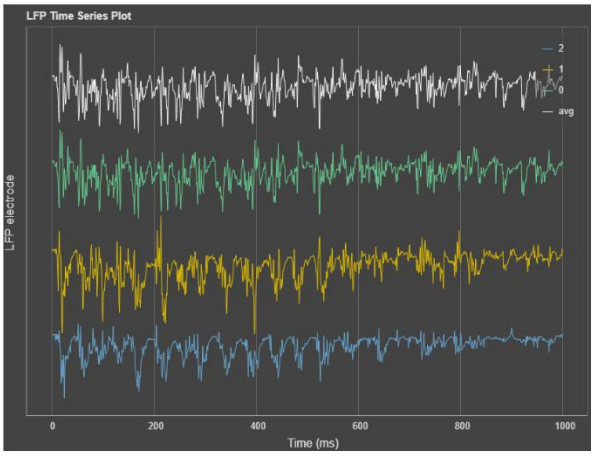


Figura 3.10. Potencial de campo local registrado por cada uno de los tres electrodos simulados y la media de estos con **(A)** una concentración inicial de IP3 de 0 y **(B)** con una población inicial de IP de 0.1.

c) Conclusiones

Tanto NEST Desktop como NetPyNe-UI son interfaces gráficas fáciles de usar que permiten el diseño de modelos de redes neuronales, aunque cada una de estas parece estar más orientada a un tipo de modelos u otros y cuenta con sus ventajas e inconvenientes. Es importante que el investigador conozca las características de este tipo de simuladores, antes de realizar sus experimentos computacionales, puesto que el diseño que se vaya a realizar puede ser mejor administrado por uno o por otro.

Ambas interfaces permiten ser utilizadas por parte de usuarios que no tengan altos conocimientos de programación. Sin embargo, como se irá comentando a lo largo de este apartado, un conocimiento nulo puede limitar severamente al investigador a la hora de diseñar este tipo de redes. La mayor parte de ocasiones, para definir de una forma más exacta la red que se quiere modelar, es necesario trabajar simultáneamente con el script de Python que la propia interfaz va a generar, pudiendo realizar los cambios en este directamente y, posteriormente importarlo a la propia interfaz para actualizarlo. Por eso, una de las características importantes es la interoperabilidad entre las GUI y el código que se desee utilizar.

En el caso de NEST Desktop, a medida que se van haciendo cambios en el editor de la red como la especificación de modelos o la adición de nodos, el código se va a ir generando automáticamente el terminal que se muestra en la GUI, permitiendo realizar las modificaciones necesarias directamente. Esto puede resultar útil en casos en los que se necesite diseñar redes más complejas, en donde la GUI puede ser usada como una primera toma de contacto para después, incorporar el código generado al script de Python y ya, desde allí, hacer los cambios pertinentes para obtener un mayor nivel de complejidad en la red. La generación simultanea del código a medida que se incorporan elementos en el editor, puede suponer una herramienta educativa a la hora de aprender los conceptos básicos de PyNEST. Además, se debe resaltar que el código es sencillo de entender y de manejar, resultando altamente comprensible para saber cuáles son las poblaciones que están siendo conectadas, que modelos se están utilizando, etc. Sin embargo, no permite ni importar ni exportar scripts de Python desde la GUI, por lo que si se quieren realizar cambios en el código se deben realizar directamente en el terminal. El problema que presenta este procedimiento, es que, aunque permite simular el código con los cambios realizados en el terminal; tras la ejecución este no se va a guardar ni va a crear una red incorporando, por ejemplo, nuevos nodos al editor. Tan solo permite importar y exportar los proyectos en formato "*.json",* siendo difícilmente interpretados por el usuario.

Por otro lado, en NetPyNe-UI, al contrario que en NEST Desktop, el código no se genera de forma dinámica a medida que se van incorporando nuevos elementos a la red, si no que una vez que esta ha sido diseñada en la GUI, se puede importar a Python el código generado y almacenarlo en el ordenador. Además se pueden realizar cambios en el script y posteriormente exportar las actualizaciones. Aunque no se genere el código de forma simultánea con la incorporación de elementos en la red, como se hace en NEST Desktop, la facilidad para importar y exportar el código permite una mayor libertad a la hora de trabajar con las modificaciones que el usuario crea necesarias.

El diseño de la red del **caso 1** pudo ser elaborada por completo y simulada mediante NEST Desktop. Esta GUI permite generar estas redes de forma sencilla e intuitiva, pudiendo visualizar claramente los distintos nodos, incluyendo las poblaciones y como estas interactúan entre sí (**Fig. 3.5 (A)**). Las distintas formas geométricas de los nodos permiten distinguir si son poblaciones neuronales estimuladores o "*recorders*". Además, tanto las poblaciones como las conexiones se pueden diferenciar si son excitatorias o inhibitorias y si dichas conexiones se conectan por completo o siguiendo una cierta probabilidad de conexión.

Por otro lado, en NetPyNe-UI no permite indicar el modelo celular que se quiere utilizar para una población. Esto puede deberse a que está más orientada a diseñar un modelo celular complejo, compuesto por varios compartimentos en los que se puede especificar su morfología. Para diseñar la población, sí se puede indicar el tipo celular que haya sido diseñado previamente, pero en este caso, al querer evaluar un modelo puntual de célula, fue necesario indicar el modelo celular directamente en el script de Python (*'cellModel': 'IntFire2'*,) y luego importarlo a la interfaz para continuar con el diseño. Además, la visualización de las poblaciones neuronales, aunque más detallada, puede no ser tan informativa como en NEST Desktop.

NEST Desktop se caracteriza por lo sencillo que resulta incorporar a la red modelos sinápticos y neuronales que se encuentran en repositorios de GitHub, como en el caso 1 que fue importado el modelo ***'iaf psc delta'***. NetPyNe-UI también permite la incorporación de nuevos modelos neuronales y sinápticos, sin embargo, no resulta tan intuitivo como en NEST Desktop. En ambos, se puede cambiar fácilmente de modelos para realizar la simulación, sin alterar el resto de la red, lo que les aporta cierta flexibilidad. Sin embargo, el usuario puede verse limitado a la hora de definir un modelo propio que se adapte a los requisitos de sus experimentos, puesto que estos modelos deben ser escritos en lenguaje NESTML en el caso de NEST y en NMOLD en NetPyNe, lo que ya requeriría tener ciertos conocimientos de programación y esta no es una competencia incorporada en las GUI.

En ambas GUI se pudo diseñar el **caso 2** sin ningún tipo de complicación reseñable. Sin embargo, si observamos los resultados de ambos casos se puede observar que, aunque son parecidos no resultan idénticos. Esto puede ser debido a que los modelos utilizados, aunque simulen el mismo fenómeno, pueden existir sutiles diferencias que den lugar a resultados distintos. Por eso, es necesario trabajar en la implementación de modelos que se ejecuten de la misma forma en distintos simuladores para facilitar la comparación entre estos.

La localización espacial de las poblaciones se puede hacer de una forma más sencilla en NetPyNe-UI, donde se puede especificar el rango en los tres ejes en el que se quiere disponer la población. En NEST Desktop, las poblaciones se pueden colocar en el espacio, o bien en unas coordenadas concretas, que se calcula indicando el centro y el número de filas y columnas de cada población, o bien de forma libre en el plano. Sin embargo, no parece permitir desde la GUI una distribución intermedia, es decir asignar un rango en un eje para una población, teniendo libertad para disponerse en las otras dimensiones.

NetPyNe-UI resulta ideal para el diseño de neuronas con una alta complejidad biofísica, compuestas por varios compartimentos con distintos parámetros estructurales. Además, es capaz de realizar una representación 3D del modelo, lo que da idea de la fidelidad con la que se puede reproducir las neuronas. Esto es algo en lo que NEST Desktop puede verse algo más limitado, aunque tiene modelos neuronales que sí permiten especificar diversos compartimentos.

En los dos softwares es necesario indicar que es lo que se quiere medir, a partir de los cuales se obtendrán las representaciones necesarias para el análisis. En NEST Desktop se hace mediante el uso de un tipo especial de nodos como los "*recorders*", los cuales simplemente deben ser conectados a los nodos que representan las poblaciones. Por otro lado, en NetPyNe-UI, la especificación de lo que quiere ser registrado puede resultar algo más difícil, aunque esta complejidad está ligada a una mayor exactitud de lo que se está registrando, puesto que se puede indicar el potencial de un compartimento de una célula de cierta población exactamente, por ejemplo. Además permite registrar a través de electrodos virtuales los LFP que se generan, algo que no es posible en NEST Desktop.

Como posibles mejoras NEST Desktop debería facilitar la importación y exportación de scripts de Python escritos en PyNEST para poder facilitar la elaboración de redes algo más complejas mediante código, y no limitarse a tener un terminal de código modificable que , aunque ejecutable, no tiene una aplicación directa en el editor de los nodos. Otra posible mejora es crear nodos espaciales que permitan distribuir las poblaciones neuronales con un grado de libertad adecuado al tipo de red que se quiera crear. En cuanto a NetPyNe-UI, sería ideal la posibilidad de generar código de forma simultánea a la que se va creando la red, aunque posiblemente esta función sea muy improbable de ser aplicada debido al flujo de trabajo explicado en 3.2. b). Además, estaría bien que se pudiese obtener un esquema simplificado de cómo está distribuida la red, incluyendo información de forma mucho más resumida, pero más comprensible para el usuario. Además, debería de incorporar una forma mucho más sencilla de importar modelos neuronales y sinápticos de una forma mucho más sencilla, por ejemplo directamente desde Github.

En conclusión, es altamente recomendable que cualquier usuario posea unos mínimos conocimientos en programación si se pretende realizar modelos algo más complejos a los que las GUI no llegan. NetPyNe-UI es muy útil si se quieren diseñar modelos neuronales complejos, con morfologías y estructuras muy detalladas y para abordar problemas en un mayor rango de escalas, puesto que como se ha demostrado, la inclusión en el modelo de moléculas a nivel subcelular puede llegar a afectar LFP, todo ello reproducible en la GUI. Sin embargo, para modelos más sencillos puede ser más recomendable el uso de NEST Desktop, puesto que a pesar de que NetPyNe-UI también es capaz de realizarlos, puede dar la sensación al diseñarlos que el usuario "se está perdiendo en los detalles".

4. Revisión de modelos macroescala

Los modelos macroescala permiten emular la actividad del cerebro en diferentes situaciones y condiciones, simulando cómo interactúan grandes grupos neuronales entre sí. Existen varias iniciativas que tratan de estudiar el cerebro en su conjunto a través de simular su actividad. Entre estos, se encuentran la plataforma *Neurogrid* que es capaz de simular el comportamiento de más de un millón de neuronas y que pretende describir cómo surge el comportamiento inteligente a partir de procesos bioeléctricos a través de diferentes escalas [2] o Blue Brain que trata de obtener la actividad neuronal del cerebro completo, simulando las sinapsis y las neuronas que se disponen en circuitos [13].

De entre todas estas iniciativas destaca *The Virtual Brain*, sobre el cual se va a desarrollar este apartado. *The Virtual Brain* no trata de simular la actividad neurona a neurona, sino que obtiene el comportamiento de varios grupos neuronales en conjunto que están conectados siguiendo una conectividad estructural basada en DTI. Además, cuenta con una interfaz gráfica de fácil manejo y una serie de características que serán explicadas a continuación.

4.1. Introducción a *The Virtual Brain*

The Virtual Brain (TVB) es una plataforma neuroinformática de simulación computacional de la dinámica cerebral utilizando conectividad biológica realista. Dicha simulación se realiza a lo largo de distintas escalas, desde poblaciones neuronales locales hasta la dinámica macroescalar que puede ser medida mediante la obtención de señales como electroencefalografía (EEG), magnetoencefalografía (MEG) e imágenes por resonancia magnética funcional (fMRI) [41] [42].

TVB contiene información real tractográfica (DTI/DSI) o conectoma con la que genera matrices de conectividad y construye redes constituidas por varios nodos interconectados. Las matrices de conectividad van a definir las intensidades de conexión y el tiempo de retraso entre nodos. La dinámica de estos nodos va a ser calculada a través de los varios modelos disponibles de masa neuronal o *mean-field models*. Se concluye entonces que, *The Virtual Brain* define las regiones a través de un núcleo matemático robusto dado por matrices de conectividad anatómicamente realistas y un modelo fisiológico de poblaciones neuronales. La inferencia de los mecanismos fisiológicos que se dan en el cerebro a distintas escalas se realiza en TVB en base al descubrimiento de los parámetros críticos que definen la red que representaría el cerebro o sus regiones. El estudio de organización de la red se puede realizar a través del análisis llevado a cabo mediante la teoría de grafos. [43]

La importancia de *The Virtual Brain* radica en su potencial uso en la investigación y en la clínica. La simulación puede ayudar a los investigadores a comprender mejor cómo se relacionan las diferentes partes del cerebro y cómo se coordinan para realizar tareas complejas. Esto permite a los investigadores estudiar cómo los cambios en una región pueden afectar a otras regiones y al comportamiento global del cerebro.

Realizar un estudio directamente sobre los humanos de un sistema tan complejo como el cerebro, resulta muy difícil debido a la gran cantidad de variables que puede afectar su funcionamiento. Esta herramienta permite hacer dicho estudio en un ambiente controlado, de tal forma que conociendo las condiciones bajo las que se realiza la simulación, los usuarios pueden explorar cómo funciona el cerebro normalmente o cómo se puede ver afectado cuando sufre distintas enfermedades, pudiendo ayudar a identificar patrones de actividad cerebral que están asociados con diferentes estados mentales o patologías neurológicas.

Uno de los objetivos principales de *The Virtual Brain* es encontrar parámetros que actúen como biomarcadores y sean capaces de relacionarse con estas situaciones de normalidad o enfermedad, teniendo capacidad predictiva para el diagnóstico y la progresión de la enfermedad y que se relacionen con procesos biofísicos subyacentes. Además, TVB cuenta con la ventaja de poder estudiar posibles biomarcadores a partir de ciertos parámetros que se establecen en los modelos que simulan la dinámica local que resultan invisibles para técnicas de imagen no invasivas. Por esto se dice que TVB actúa como un microscopio. [44]

Además, en la clínica, *The Virtual Brain* también puede resultar muy útil debido a que se puede incorporar datos individuales, como EEG, MEG y fMRI, y crear modelos cerebrales para cada paciente. Así, se posiciona como un elemento de gran relevancia en la medicina personalizada.

En su afán de integrar a la mayor parte de la comunidad científica, TVB distingue dos tipos de usuario en función del nivel de conocimiento que posean en lenguajes de programación. TVB está construido utilizando lenguaje Python, por lo que aquellos que posean cierto nivel pueden interactuar directamente con el código pudiendo crear modelos de alta complejidad. Por otro lado, para aquellos usuarios que no tengan estos conocimientos, TVB pone a su disposición una serie de herramientas interactivas con las que se puede llevar a cabo un gran número de distintas simulaciones.

En su interfaz web de fácil acceso están recogidas todas estas herramientas pensadas para el uso de este último grupo de usuarios. Desde aquí se puede configurar distintos parámetros que definen la simulación que se quiere realizar. Además, incluye un gran número de funciones con las que se puede visualizar y analizar los resultados obtenidos como gráficos interactivos y mapas de color. Entre estas herramientas se encuentran técnicas de análisis de series de tiempo, de análisis de conectividad estructural y funcional, y la posibilidad de explorar los parámetros que definen las simulaciones [41].

TVB organiza la información en proyectos, los cuales están orientados para poder ser utilizados por varios usuarios a la vez, en donde cada uno de ellos cumpliría un papel. Por ejemplo, como se indica en [42], un médico podría importar datos de pacientes referidos a puntos de lesión del conectoma. Posteriormente, alguien encargado de llevar a cabo la simulación, puede comprobar la viabilidad de dichos puntos y analizar las características dinámicas.

Esta plataforma pretende tener un impacto positivo en la comunidad científica promoviendo la comunicación entre los investigadores. TVB puede ser utilizada como un sistema de validación de modelos, debido a que fomenta la reproductibilidad de aquellos modelos ya existentes, pudiendo comparar los resultados de estos con los que se obtienen de enfoques novedosos, tanto propios como de otros investigadores. La reproducibilidad se alcanza debido a la capacidad del sistema de ser ampliado con la agregación de módulos, consiguiendo esta característica esencial para la validación. De esta manera TVB otorga fiabilidad al trabajo científico que se realiza con este software.

TVB está compuesto principalmente de un marco de trabajo (*framework*) y el simulador o núcleo de computación científico propiamente dicho. Desde el marco del trabajo se administran los proyectos, que a su vez incluyen datos, sujetos de estudios y usuarios que participan en él. Además, posee una base de datos donde se encuentran las operaciones realizadas por los usuarios y los datos de neuroimagen estructural y funcional. Por otro lado, el simulador proporciona los métodos numéricos con los que se construyen los modelos de los que se obtendrá la actividad cerebral simulada. Además, a partir de esta puede calcular datos de neuroimagen que podrán se comparados con las imágenes obtenidas en la clínica. Estos dos componentes pueden ser utilizados a través de una interfaz gráfica [42]. Además, es compatible con diferentes sistemas operativos, incluyendo Windows, Mac OS X y Linux.

a) Multiescalaridad

Como ya se ha dicho en repetidas ocasiones a lo largo de este trabajo, el cerebro es un sistema dinámico y complejo que opera en diferentes escalas, tanto espaciales como temporales. Por esta misma razón, si se pretende realizar una simulación del cerebro en su conjunto, no se puede obviar como interactúan las diferentes escalas para dar lugar a la dinámica general macroescalar que emula TVB.

El abordaje de este problema se puede hacer desde distintas perspectivas como se explica en [45]. La primera consiste en crear un “marco multiescalar unificador”, en donde cada escala está gobernada por un conjunto de ecuaciones y existe una regla subyacente única que vincula cada escala iterativamente con la siguiente. El problema de esta aproximación es que se asume que todas las escalas siguen las mismas reglas de organización, algo que no ocurre en el cerebro donde cada escala puede tener distintas normas de organización.

Otro enfoque es el que llama “fuerza bruta”, donde se construyen modelos detallados de microcircuitos neuronales individuales. A través de la interacción de estos microcircuitos se va avanzando cada vez en redes de mayor escala hasta poder simular redes cerebrales completas. El principal problema de esta aproximación es su difícil implementación y la gran potencia computacional que es requerida.

Debido a los inconvenientes que presentan las dos estrategias anteriores, TVB opta por una tercera vía. Esta consiste en utilizar las leyes mesoscópicas que rigen el comportamiento de las poblaciones neuronales para descubrir las leyes que impulsan los

procesos a nivel macroscópico de la red cerebral [41]. Este avance de abajo hacia arriba a través de las escalas, no se hace a "fuerza bruta", sino que se consigue mediante la reducción de dimensiones como la utilización de los *mean-field models*. De esta forma, fenómenos que se producen en la microescala pueden ser introducidos en estos modelos mesoscópicos en forma de parámetros, pudiendo mantener la influencia de las pequeñas escalas en las más grandes. Además, también se puede "cavar" a través de las escalas de arriba hacia abajo, añadiendo parámetros más detallados.

Esta última aproximación representa un balance entre operabilidad del modelo y complejidad. Se pueden construir modelos sofisticados, pero no excesivamente complejos y explorar como afecta cada parámetro presente en distintas escalas a la generación de la dinámica cerebral.

Los modelos de redes cerebrales en TVB se puede dividir en tres niveles escalares principales. El primero sería la propia dinámica por la que está gobernada cada nodo. A esta se le debe añadir la interacción de dicho nodo con los que está conectado, que representaría los dos niveles restantes. El segundo nivel sería la interacción del nodo con las redes locales vecinas. El ultimo nivel representaría las conexiones del nodo con estructuras que se encuentran a gran distancia, mediadas por fibras de sustancia blanca, representado en TVB por matrices de conectividad. La implementación de esta multiescalaridad en TVB permite incluir todos estos niveles a la vez o, hacer una combinación del primero y segundo o del primero y del tercero. [18]

Sin embargo, estos tres niveles de organización escalar no deben ser estudiados aislados y bien definidos, sino que más bien sus límites son difusos debido a la relación intrínseca entre estos (**Fig. 4.1.**). Por ejemplo, en TVB los nodos pueden representar un nivel mesoscópico al describir la dinámica de una población de neuronas, pero a su vez los parámetros que están incluidos en las ecuaciones que definen a estos nodos se pueden asociar a fenómenos biofísicos que ocurren a nivel microscópico. De la misma forma, las conexiones que se producen a larga distancia son macroscópicas, pero algunas de estas se pueden producir dentro de la propia región, lo que se considera mesoscópico. [44]

b) Estructura espacio-temporal

Uno de los principales conceptos con los que trabaja TVB es con el de "estructura espacio-temporal" (**Fig. 4.2.**). A diferencia de las señales producidas en las redes locales que se puede considerar que se transmiten de forma instantánea, las señales que comunican diferentes regiones cerebrales van a sufrir retrasos temporales, debido a la distancia espacial existente y que dichas señales deben recorrer. El acoplamiento entre estas dos regiones va a depender de una serie de propiedades que van a ser recogidas por la estructura espacio-temporal, como son la fuerza de conexión, de la dirección de la señal y de los retrasos temporales. En resumen, la estructura espacio-temporal representa cómo las diferentes regiones del cerebro están conectadas entre sí y cómo interactúan a lo largo del tiempo.

Diversos estudios han indicado que la estructura espacio-temporal es esencial para poder generar una dinámica cerebral adecuada. Por ejemplo, en [46] se demuestra que la inclusión de retrasos temporales es esencial para la comprensión del cerebro en estado de reposo. Por su parte, *Honey et al.* [47] trata de relacionar características de la dinámica cortical espontánea con la conectividad anatómica subyacente. En [45] se describen dos efectos de la inclusión de estos retrasos temporales: enriquece la dinámica de la red generando comportamientos ausentes de otra forma y manipulando adecuadamente la estructura es posible generar representaciones simplificadas del comportamiento de la red, es decir, es posible buscar parámetros estables (saludables) de la red.

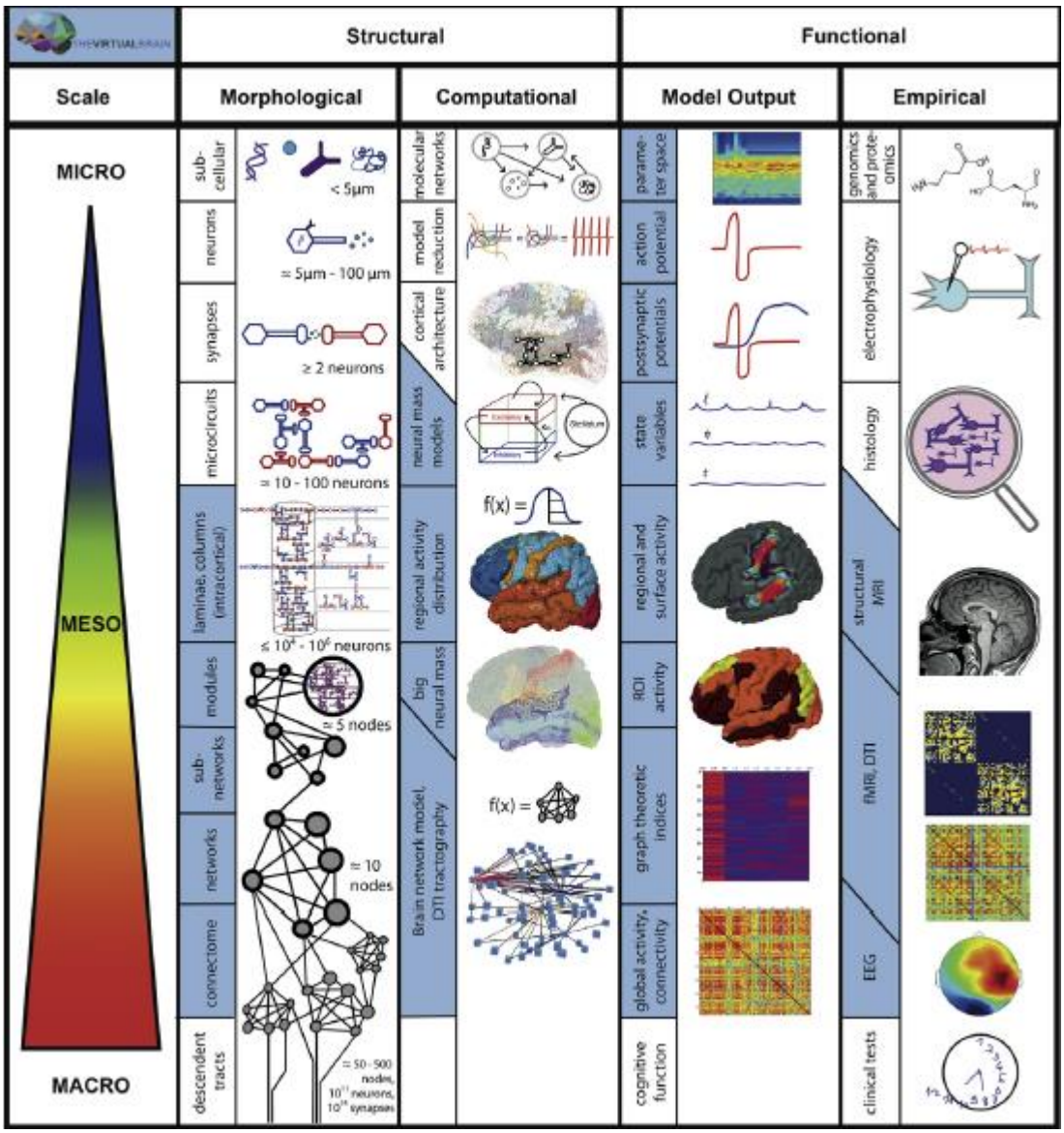


Figura 4.1. Evolución de las diferentes escalas, desde la micro (arriba) hasta la macro (abajo), tanto a nivel estructural como funcional [44].

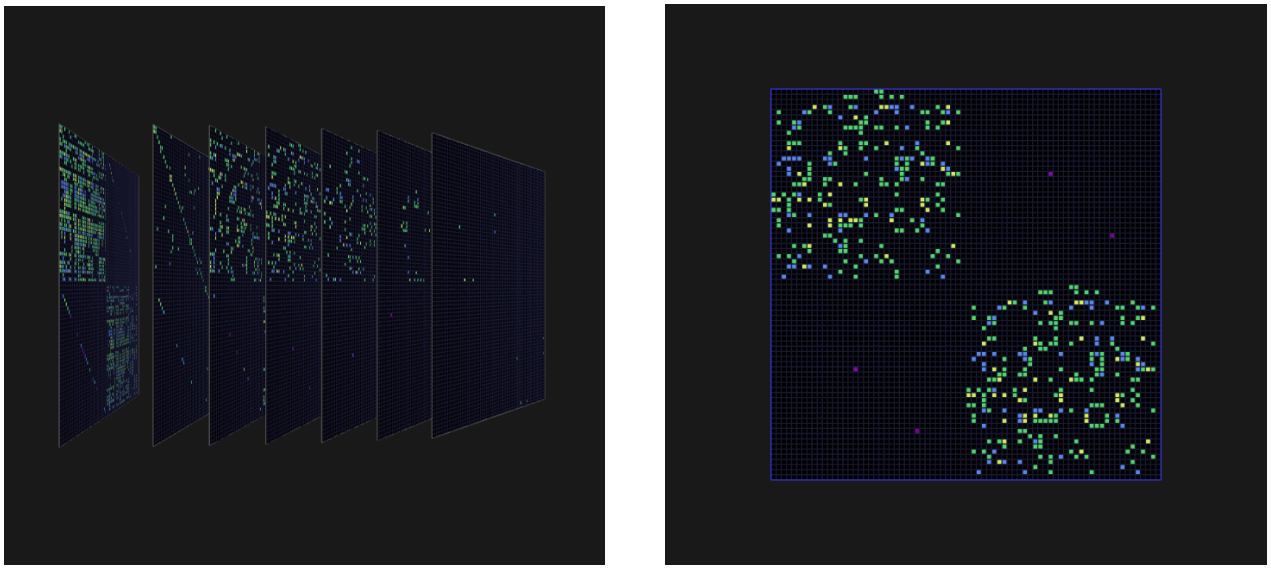


Figura 4.2. Estructura espacio-temporal de TVB.

c) Resting-State

La actividad cerebral puede ser medida, básicamente, cuando el sujeto se encuentra en un estado de reposo (*resting-state*), o bien tras pedirle que realice una actividad en concreto (*evoked activity*). Tradicionalmente, se ha considerado que la actividad espontánea que se produce en el cerebro en estado de reposo era simplemente ruido. Sin embargo, siguiendo las últimas investigaciones sobre las redes de reposo, se ha demostrado la presencia de patrones espacio-temporales que revelan la estructura funcional del cerebro, pudiendo representar reminiscencias de comportamiento anterior o intentos de hacer predicciones sobre decisiones futuras [45][43]. Esto quiere decir que la estructura espacio-temporal en reposo va a ser regulada por el ruido,

4.2.Arquitectura

Uno de los principales objetivos del diseño de la arquitectura de TVB es conseguir la integración de distintas herramientas computacionales permitiendo diferentes tipos de datos ser manejados con un único sistema y que sean utilizados para los procesos que se llevan a cabo [42].

Este enfoque se ve reflejado en cómo se comunican sus dos elementos principales [41]:el núcleo de computación científica y el marco de apoyo con la interfaz gráfica de usuario (*framework*). Dicha comunicación se realiza a través del uso de TVB-***Datatypes.*** Además, también se utilizan los ***adapters*** para facilitar dicha integración. Los TVB-***Datatypes*** y ***adapters*** son estructuras y funciones que permiten la interoperación con la base de datos y la generación de una interacción con el interfaz de usuario.

El marco de trabajo (*framework*) es el componente con el que se manejan los proyectos. Cada uno de estos proyectos contiene los datos con los que se trabaja, los sujetos, los usuarios que trabajan sobre él y sus respectivos roles [42]. Dentro de este podemos encontrar una base de datos en la que se registran las operaciones que se realizan y los datos asociados a dichas operaciones. El simulador contiene diferentes métodos numéricos para procesar los diferentes modelos basados en la anatomía cortical y subcortical y en la dinámica humana. Este es el componente principal y se detallará su funcionamiento posteriormente. Finalmente, ambos componentes son manejados por el usuario a través de la interfaz gráfica.

Como ya se ha mencionado anteriormente, se puede clasificar a los usuarios en función de sus conocimientos en programación: el usuario gráfico (Usuario-G) con un nivel bajo y el usuario con conocimientos o *scripting user* (Usuario-S). Esto va a afectar a la arquitectura de TVB, puesto que cada usuario va a utilizar un interfaz distinto. Cada uno de estos accederá al sistema *back-end*, que lleva a cabo los distintos procesos, de una forma diferente.

El Usuario-G utilizará la interfaz gráfica web (GUI), desde donde no necesita realizar ningún tipo de programación. Esta interfaz es una página web interactiva desde donde se puede introducir los datos necesarios para llevar a cabo las simulaciones. Además desde aquí también se puede visualizar los resultados obtenidos. Sin embargo, este tipo de usuarios están más limitados a la hora de utilizar todas las funcionalidades disponibles que ofrece TVB.

El Usuario-S, por el contrario, va a ser capaz de diseñar modelos más complejos y trabajar con el código directamente pudiendo realizar modificaciones en él. La interfaz usada por este grupo es IDLE (*Integrated Development and Learning Environment*), el entorno de desarrollo integrado por Python.

Es posible, también, establecer un tercer grupo al que pertenecerían los usuarios con ciertos conocimientos en lenguajes de programación que pueden hacer modificaciones en el código y, posteriormente comprobar estos cambios desde la página web.

a) TVB Framework

El *framework* o marco de trabajo es el componente que sirve de apoyo al software TVB y presenta una serie de funcionalidades importantes para su ejecución. Presenta *un back-end* de base de datos con la capacidad de almacenar y recuperar datos utilizados, permitiendo a los usuarios guardar y compartir sus proyectos. Se encarga de la gestión de flujos de trabajo, organizando las tareas y permitiendo a los usuarios gestionar y automatizar los procesos de trabajo.

Además permite una de las características de *The Virtual Brain* más destacadas, el trabajo colaborativo de un mismo proyecto por varios usuarios, los cuales se pueden corresponder con profesionales de diferentes disciplinas. El *framework* permite que distintos usuarios compartan datos y proyectos.

Otro de los rasgos particulares de TVB que consigue a través del *framework* es permitir que la interfaz gráfica de usuario esté basada en la web. Esta permite que los usuarios puedan acceder al simulador mediante un navegador web y utilizarlo de forma remota. En su diseño se utiliza HTML 5, WebGL, CSS3 y Java Scrip. Esta interfaz está diseñada para el Usuario-G, sin muchos conocimientos de programación.

La interfaz gráfica web de TVB se distingue por ser interactiva y poder de una forma sencilla ejecutar las distintas simulaciones que se quieran realizar en los proyectos correspondientes, así como de utilizar una herramienta que permite la visualización de la conectividad y dinámica de la red. Además dispone de una serie de recursos a su alcance para realizar un análisis más exhaustivo como el análisis de series de tiempo, de conectividad estructural y funcional y la exploración de parámetros.

b) Integración de los elementos en la arquitectura

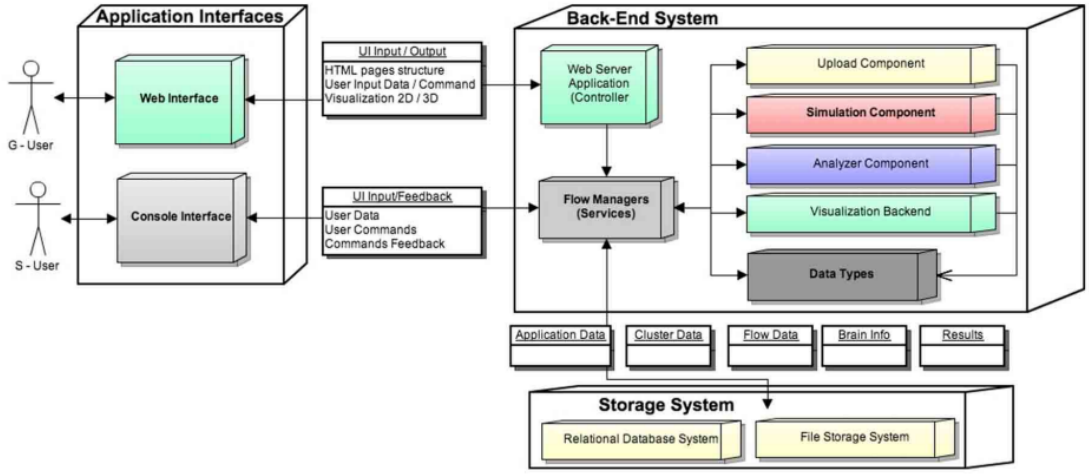


Figura 4.3. Arquitectura de TVB [41].

En la **Fig. 4.3.** se observan cómo están integrados los distintos elementos de TVB en una única arquitectura. Dependiendo del tipo de usuario que vaya a realizar una simulación, se va a utilizar un interfaz independiente u otro. Los inputs que introducen van a parar al gestor de flujo, de donde también saldrán los outputs que podrán recibir los usuarios. Este gestor de flujo se refiere a cómo TVB trabaja internamente. Dentro de cada proyecto, el usuario va a poder realizar diferentes operaciones. Una operación es la ejecución de un algoritmo. Estos algoritmos pueden ser *uploders*, que permite importar datos al marco de trabajo; *analyzers* que operan con series temporales y comprenden técnicas como Fast Fourier Transform (FFT), ICA, PCA, entre otras; y por último *visualizers* que permiten mostrar el contenido de los *Datatypes.*

Los *Datatypes* son estructuras de datos anotadas que contienen atributos de datos e información descriptiva asociada y métodos para operar con dichos datos [41].Desde el punto de vista de la programación, es una clase de Python. Hay que recordar que Python

es un lenguaje de programación orientado a objetos. En este tipo de lenguajes, una clase provee una forma de empaquetar datos y funcionalidad juntos, es decir, es una plantilla o modelo que define las propiedades y el comportamiento de un objeto.

La definición de *Datatype* se obtiene a partir del sistema de caracterización (*traiting system*), que contiene los mecanismos para anotar datos o asociar información adicional con el dato propiamente dicho que suele ser un número o un vector. Algunos de estos atributos son rasgos (*traits*), donde un rasgo especifica tanto el tipo de datos esperados para el atributo correspondiente como metadatos adicionales para ayudar en el almacenamiento y la construcción de la interfaz de usuario. Los *Datatypes* pueden ser básicos o más complicados, teniendo estos últimos como atributos los más básicos. Algunos de estos *Datatypes* más complejos pueden ser Connectivity, Surfaces, Volumes, Sensors... Además, los Datatypes pueden incluir metadatos que incluyen una descripción técnica de los datos y otras propiedades [42].

A continuación se muestra un ejemplo con el que explicar este sistema. Un TVB-*Datatype* básico es TVB-***FloatArray*** que se deriva del sistema de caracterización del tipo Array. Un TVB-*Datatype* complejo podría ser TVB-***Connectivity,*** el cual tiene como atributos otros *Datatypes* básicos, como varios ***FloatArrays*.**

4.3.Flujo de trabajo

TVB, como se mencionó en 4.1., pretende inferir los mecanismos fisiológicos que ocurren en el cerebro a partir de la determinación de distintos parámetros que caracterizan la red que ha sido modelada. Estos parámetros pueden utilizarse como valores indicativos del estado del paciente que puedan ayudar en el diagnostico. Para conseguir esto, cuando se trabaja con TVB, es necesario realizar una optimización de los parámetros que definen esta red simulada por el software a partir de datos empíricos recogidos al paciente. Esto se realiza siguiendo un flujo de trabajo que pretende ser explicado en esta sección. A continuación se muestra un pequeño resumen de los pasos, mostrados en **Fig. 4.4.**, que serán explicados con mayor precisión en cada uno de los subapartados que se desarrollan posteriormente.

1. Definición de las matrices estructurales. A partir de la realización dw-MRI se reconstruye el conectoma y se crean las matrices estructurales, que van a especificar tanto la distancia entre los distintos nodos de la red, como la intensidad de la conexión. Estas van a actuar como una restricción espacial en nuestro modelo.
2. Definición de las matrices de funcionalidad empíricas. A partir de la realización de fMRI en estado de reposo (*resting-satate*), se obtiene dicha matriz funcional empírica.
3. Construcción de la red. El conectoma es remapeado a partir de los diferentes atlas cerebrales disponibles. De esta forma se construye la red que tiene dos componentes principales: nodos y aristas. Los nodos estarán modelados por masas

neuronales o *mean-field models,* teniendo distintas opciones para elegir. Las aristas representan esa restricción espacial.

4. Ajuste del modelo a partir de información previa (EEG, MEG, PET…)
5. Obtención de la matriz funcional simulada. Los nodos de la red van a generar una dinámica cerebral, a partir de la cual se obtiene la matriz funcional simulada.
6. Optimización de parámetros. A través de un proceso iterativo los parámetros se van a optimizar comparando las matrices simuladas y empíricas, hasta que estas estén correlacionadas estadísticamente.
7. Análisis de los parámetros obtenidos y de la simulación realizada.

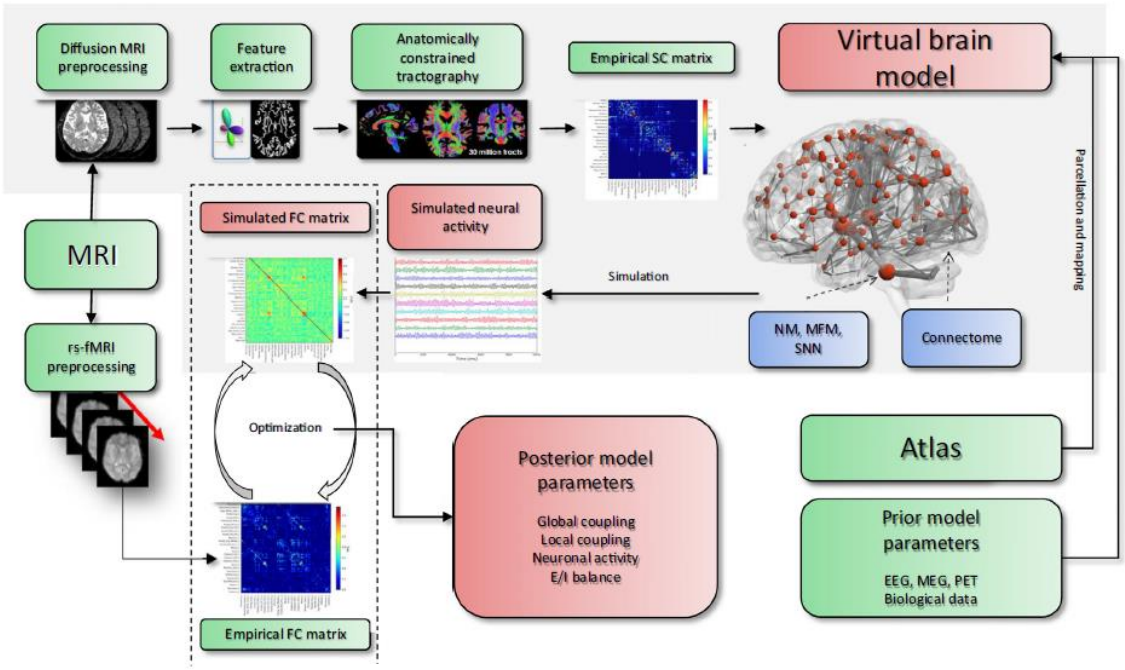


Figura 4.4. Flujo de trabajo de TVB [4].

TVB aborda el problema el problema de la inferencia de parámetros usando una estrategia top-down (arriba-abajo) puesto que a partir de las señales obtenidas de un paciente (EEG, fMRI, MEG, PET…) obtiene conclusiones de cómo se desarrolla la actividad cerebral. Sin embargo, incorpora en este enfoque una estrategia *buttom-up* (abajo-arriba), puesto que la simulación consiste en modelar primero las poblaciones neuronales y posteriormente ver como estas interactúan entre ellas a lo largo de distintas escalas espaciales y temporales, conformando finalmente una simulación total del cerebro. De esta forma obtenemos una simulación híbrida en distintas escalas [4].

a) Modelos de redes cerebrales

TVB realiza sus simulaciones en base a la definición de modelos de redes cerebrales. Estas redes van a generar la dinámica neuronal simulada del cerebro virtual. Van a estar compuestas por una serie de nodos conectados entre sí. Los nodos van a representar

poblaciones neuronales, es decir, simulan actividad cerebral a nivel mesoescalar. Las conexiones que existen entre estos nodos van a describir cómo la dinámica generada por un nodo va a afectar a aquellos con los que está conectado. Es decir, la actividad de las diferentes poblaciones neuronales no es independiente, sino que interacciona con otras, surgiendo a partir de esta interacción la dinámica cerebral a macroescala [42].

Por lo tanto, TVB va a crear una red compuesta por nodos y las aristas que los unen. Cada uno de estos nodos va a representar un población neuronal que va a generar su propia dinámica interna mediante un modelo previamente definido. Sin embargo, esta dinámica se va a ver afectada tanto por la actividad producida en los nodos más cercanos o de conectividad local (mesoescala) y por los nodos que se encuentran en zonas más lejanas de la red o conectividad a gran escala (macroescala). Es aquí donde se fundamenta la interacción multiescalar característica de TVB.

La evolución de este sistema dinámico se computa mediante ecuaciones diferenciales [44]:

$$\frac{dx_i(t)}{dt} = N\big(x_i(t)\big) + G\sum_j SC_{ij}x_j\big(t - {d_{ij}}^{glob}\big) + L\sum_j LC_{ij}x_j\big(t - {d_{ij}}^{loc}\big) + I_i(t) + \nu_i(t)$$

Eq. 4.1.

Donde:

- N( ): modelo de masa neuronal
- $x_i(t)$: estado actual de las variables de estado modeladas
- G: factor de escala de acoplamiento global
- SC_{ij}: matriz de conectividad estructural (larga-distancia)
- ${d_{ij}}^{glob}$: retrasos temporales
- LC_{ij}: matriz de conectividad local
- $I_i(t)$: introducción de un input
- $\nu_i(t)$: ruido

Esta ecuación va a describir cómo cambian las variables de estado del modelo de cada nodo, dependiendo de la actividad de otros nodos tanto vecinos como lejanos. Además aparece otro elemento en la ecuación, la introducción de un input, TVB permite realizar experimentos introduciendo un estímulo a la red. Otro elemento es el ruido, que como ya se ha explicado, regula la estructura espacio-temporal.

b) Conectividad estructural

Uno de los aspectos más importantes para poder realizar una simulación realista de la actividad cerebral es tratar de comprender la relación existente entre la conectividad estructural y funcional del cerebro. Se trata de comprender cómo la forma en la que están

distribuidas las regiones en el cerebro y las interacciones entre estas, afectan a la dinámica de la red; y a su vez, cómo dicha dinámica regula estas conexiones estructurales [45] [48].

Esta cuestión esencial surge debido al avance de la neuroimagen. Debido a la estructura anatómica del cerebro, la señal que se genera en una región, y que va a afectar la dinámica neuronal de otra, debe recorrer una distancia a una velocidad asociada, por lo que van a surgir retrasos temporales que van a resultar esenciales para caracterizar la estructura espacio-temporal de la red [49]. Mediante la utilización de imagen de difusión y tractografía, podemos medir las distancias de las fibras neuronales.

TVB incluye una conectividad biológicamente realista para simular redes cerebrales a gran escala de las distintas regiones del cerebro. Esta conectividad realista viene dada por el conectoma, que es un mapa de las conexiones entre distintas zonas del cerebro y describe tractos de fibra de sustancia blanca. Este conectoma va a estar definido por dos tipos de matrices de conectividad: la matriz de tramos de longitud y la matriz de pesos de conectividad. La primera define la distancia de las fibras neuronales entre las regiones y se caracteriza por su simetría (la distancia entre dos regiones es la misma independientemente de desde donde se comience a medir). La segunda, por su lado, indica la fuerza con la que están conectadas las distintas regiones y, al contrario que la anterior es asimétrica, puesto que esta sí es sensible a la direccionalidad. La dimensión de ambas matrices es de *regiones* x *regiones* [41] (**Fig. 4.5.**).

TVB distingue entre dos tipos de conectividad estructural: la conectividad a largo alcance la conectividad a corto alcance [41] [18].

La **conectividad a largo alcance** o heterogénea, describe las conexiones entre las diferentes regiones cerebrales. Esta viene dada por las matrices de conectividad que describen la distancia a la que están separadas distintas regiones (matriz de tramos de longitud), que junto a una velocidad de transmisión finita determinada dará lugar a los retrasos temporales, y cómo de fuertes son dichas conexiones.

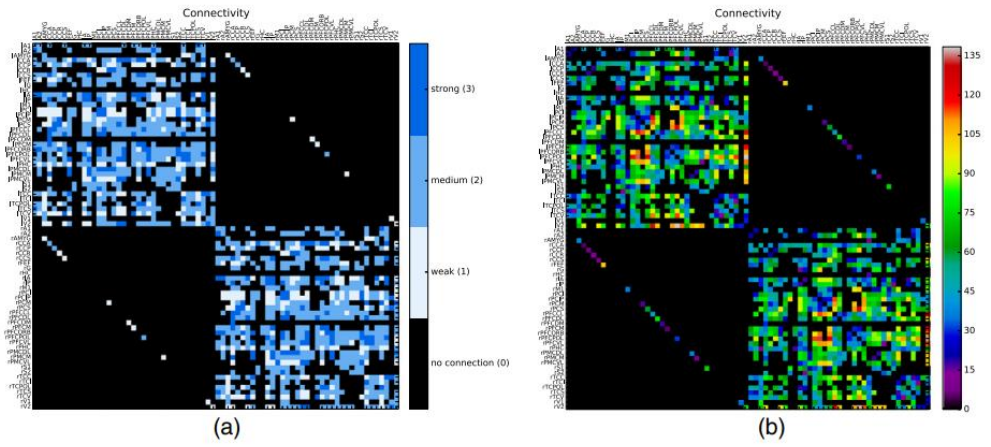


Figura 4.5. (A) Matriz de conectividad de pesos. **(B)** Matriz de conectividad de distancias [18].

Por su parte, la **conectividad a corto alcance**, local u homogénea; describe las conexiones entre nodos contiguos. Estas conexiones son representadas por "*kernels* de conectividad" locales, que describen cómo las distintas poblaciones neuronales dentro de una región, o que se encuentran en la frontera de regiones, están interconectadas. Estos *kernels* son funciones que describen cómo la conectividad decae exponencialmente con la distancia hasta 0, es decir, cuanto más lejos se encuentre la población, menos se verá afectada. Los *kernels* pueden ser completamente positivos, lo que biológicamente se interpretaría como excitación, o pueden tener componentes tanto positivos como negativos (excitación e inhibición). La distancia entre los vértices y sobre la que se aplican los *kernels* es la distancia geodésica (distancia a lo largo de la superficie cortical). Se asume que los *kernels* son traslacional y rotacionalmente invariantes, por eso a esta conectividad también se le llama homogénea (**Fig. 4.6.)**.

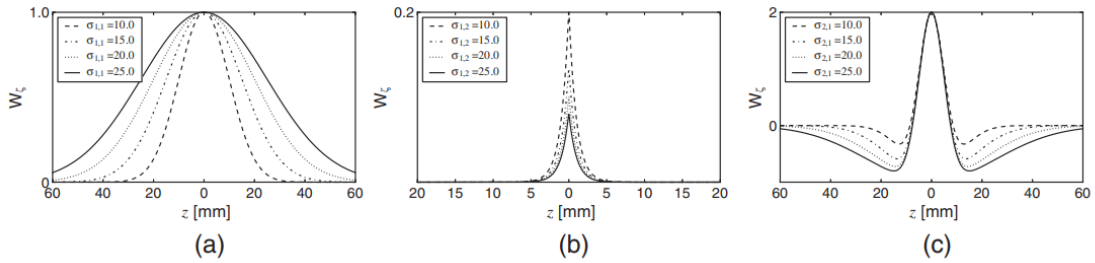


Figura 4.6. Funciones usadas para kernel de conectividad. (A) Gaussiana (B) Laplace (C) Sombrero mexicano [18].

Las simulaciones que realiza TVB puede obviar la conectividad local, consiguiendo de esta forma un menor coste computacional. Dependiendo si las simulaciones consideran o no este tipo de conexión, las simulaciones pueden llevarse a cabo en dos **escalas espaciales** distintas: basada en regiones o basada en superficie.

La simulación **basada en regiones** se enfoca en las diferentes áreas (regiones de interés, ROIs) en las que está dividido el cerebro. Las redes con las que se trabaja están compuestas por tantos nodos como regiones tiene el cerebro. Estos nodos discretos van a modelar la actividad de cada región. La conectividad entre regiones se define mediante fibras interregionales que tienen una longitud del orden de unos pocos centímetros, que viene representadas en las matrices de conectividad. Cómo sólo se van a tener en cuenta las regiones, y estas están separadas lo suficiente como para que existan retrasos temporales, tan sólo se tiene en cuenta la conectividad de largo alcance. Suele resultar útil para modelar cómo diferentes áreas del cerebro interactúan entre sí y cómo se propagan las señales a través de la red neuronal.

Cuando se resuelve numéricamente este tipo de simulaciones, la actividad proveniente de otra región se introduce en una ecuación de acoplamiento, previamente a ejercer una influencia en la dinámica del nodo al que se dirige. Esto se hace con el objetivo de conseguir que entre de una forma adecuada en el modelo. Hay varias funciones de

acoplamiento que pueden ser usadas en TVB, dependiendo del modelo con el que se esté haciendo la simulación, será más adecuado uno u otro. Algunas de ellas son acoplamientos lineales, tangente hiperbólica o sigmoide.

Por otro lado, la simulación **basada en superficie** utiliza una representación tridimensional del cerebro que tiene en cuenta su estructura anatómica realista. La corteza cerebral puede ser considerada como una superficie de 2D dispuesta en un espacio tridimensional. Esta superficie o malla va a estar conformada por distintos triángulos (**Fig. 4.7.**). Cada uno de los vértices va a ser un nodo que representará a una población neuronal más específica que en la simulación por regiones. En este caso, varios nodos, que representan una población neuronal, forman parte de una misma región cerebral y las aristas de la red tienen una distancia de unos pocos milímetros.

Para la obtención de esta malla existen distintos softwares como *FreeSurfer*, que puede generar una superficie de alta resolución de hasta 250.000 vértices [50]. Debido a que un gran número de vértices supone mayor gasto computacional, es necesario simplificar la red. Para ello existen diversos algoritmos que consiguen reducir el número de vértices, siendo una cantidad adecuada de entorno a los 10.000 – 20.000. Además se debe realizar un mapeado del cerebro utilizando atlas, que nos va a permitir dividir el cerebro en regiones y asociar cada nodo a una región concreta. El mapeado es previo a la simplificación de la red, por ello tras este proceso puede haber errores en los que ciertas zonas de la malla no estén asociadas correctamente a la región. Para corregir estos errores, se puede asignar el vértice de la superficie diezmada a la misma región a la que pertenece el vértice más cercano de la superficie de alta resolución.

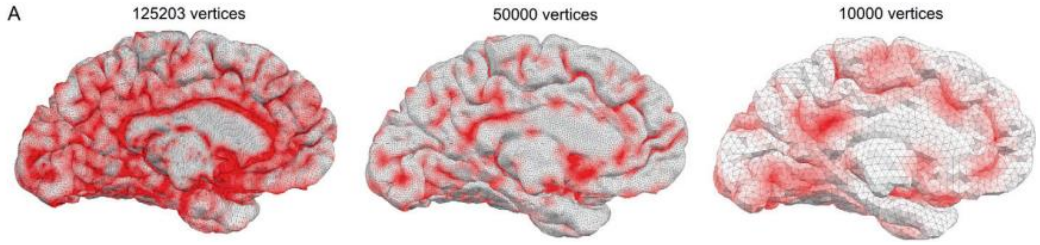


Figura 4.7. Superficie tridimensional del cerebro compuesta por distinto número de nodos [50].

En este tipo de simulaciones la conectividad a largo alcance juega un papel importante, incorporándose la actividad proveniente de otras regiones con su retraso correspondiente. Por esto mismo, en este tipo de simulaciones, participan tanto la conectividad a largo alcance como la de corto alcance, mientras que en la basada en regiones sólo la de largo alcance. Por lo tanto, la actividad neuronal generada por un nodo se va a ver afectada tanto por los nodos próximos mediante la conectividad local, asociada a fibras intracorticales, y por la conectividad de largo alcance proveniente de otras regiones, que serán la actividad media del conjunto de vértices que conforman la región (**Fig. 4.8.**).

Este tipo de conectividad es útil para modelar cómo las señales neuronales se propagan a través del cerebro y cómo interactúan con su estructura anatómica realista.

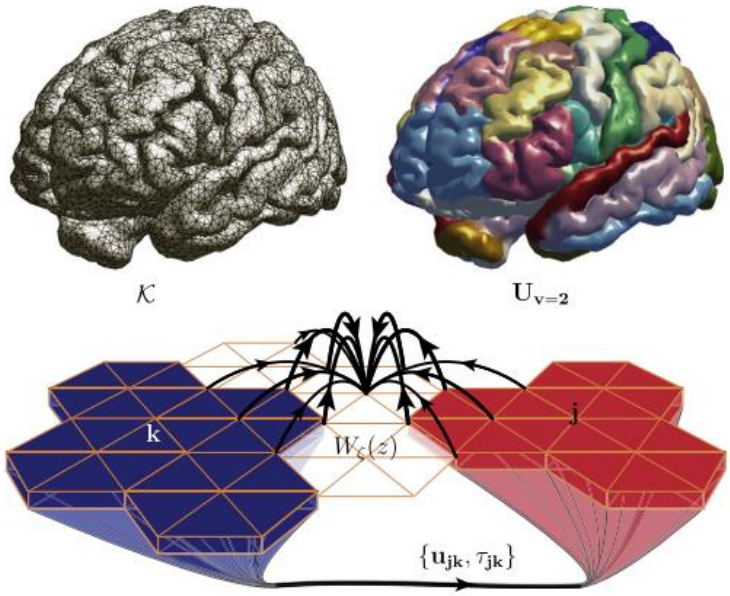


Figura 4.8. *Conectividad a corto y largo alcance. En la conectividad a corto alcance el cerebro se considera una malla triangular compuesta por nodos conectados entre sí (arriba izquierda). En la conectividad a largo alcance el cerebro se divide en regiones (arriba derecha). En las simulaciones basadas en superficies, un nodo se va a ver afectado, tanto por la actividad de los nodos vecinos (corto alcance), como por la actividad conjunta de las regiones conectadas representadas por un color distinto (largo alcance).*

c) Generación de señales y neuroimágenes simuladas: EEG/MEG, fMRI-BOLD *contrast*

Los datos de la dinámica del modelo de red cerebral se caracterizan por ser de muy alta dimensionalidad y, por lo tanto, presenta dificultades a la hora de ser almacenado e incluso para trabajar. Estos datos crudos que se obtienen son un vector de 4 dimensiones (corresponde a un *datatype TimeSeries*). Estas 4 dimensiones representan el número de puntos en el tiempo, el número total de variables de estado que caracterizan un modelo que determina la dinámica de cada nodo, el número de modos del modelo y el espacio muestreado, que puede ser regiones cerebrales, vértices de la malla o una combinación de ambas [41].

TVB integra en su arquitectura el módulo MONITORS que contiene la clase *monitors*, que permite transformar estos datos crudos con el objetivo de reducir su dimensionalidad y de generar datos que sean directamente comparables con señales obtenidas experimentalmente. Así pues, podemos definir dos tipos de *monitors*, crudos o de bajo nivel y biofísicos o de alto nivel.

El monitor más simple, el monitor crudo, simplemente devuelve los datos simulados. Es utilizado para la verificación de modelos y simulaciones. El resto de *monitors* de bajo nivel llevan a cabo un submuestreo en algunas de las dimensiones, como puede ser el espacio de estado (seleccionando un conjunto reducido de variables de estado), el tiempo

(devolviendo solo cada k puntos), del espacio (devolviendo k vectores). También se puede incluir el promediado en el espacio o en el tiempo. La salida de estos *monitors* de bajo nivel va a ser también de 4 dimensiones [tiempo, variables de estado, espacio, modos].

Por otro lado, los *monitors* biofísicos o de alto nivel permite generar señales de neuroimagen a partir de la dinámica simulada. Este tipo de señales han sido muy utilizadas para evaluar a los pacientes, tanto en la clínica como en la investigación, permitiendo elaborar un diagnóstico y estudiar diferentes opciones terapéuticas. La generalización del uso de señales que evalúan el cerebro a nivel macroscópico, era una de las principales motivaciones por las que crear TVB. Aquí radica la importancia de crear "*forward solutions*" que nos permita obtener dichas señales simuladas. Esto permitirá realizar una comparación con otras señales de sujetos reales. De esta forma, se pueden obtener señales de electroencefalografía (EEG), magnetoencefalografía (MEG) e imagen por resonancia magnética funcional (fMRI).

Forward solutions

Cálculo de EEG

El objetivo para el cálculo de la señal de EEG es obtener el potencial eléctrico del cráneo a partir de una distribución de corriente primaria dada. Se debe calcular cuánto contribuye cada una de las fuentes (los datos crudos) que van a constituir dicha distribución en cada una de las señales que es recogida por el sensor de EEG [51].

Cada una de las fuentes va a ser tratada como dipolos generados por neuronas excitatorias que tienen sus dendritas orientadas verticalmente a la superficie cortical y representa en torno al 85% de las neuronas. No se tiene en cuenta la contribución realizada por células inhibitorias, ya que representan una cantidad mínima (~15%). Los dipolos van a estar definidos por 6 grados de libertad que representan su posición y su orientación, la cual va a ser normal a los triángulos que componen la superficie cortical de la simulación. Por lo tanto, se define la distribución de la corriente en el tiempo y en el espacio.

Una de las características de las señales de EEG es que el sensor no se encuentra en contacto directo con la fuente, sino que entre medias hay tejidos biológicos con propiedades conductivas muy distintas. Este va a ser el origen del problema de conducción de volumen, que consiste en que el sensor no va a capturar solamente la información de las fuentes que se sitúan inmediatamente debajo de él, sino que también va a capturar la dinámica proveniente de otras fuentes, cuya corriente se ha transmitido a través de estos tejidos biológicos. Por eso, como se indicaba al principio del apartado, el propósito es conocer en qué medida cada una de las fuentes va a afectar a la señal captada por un sensor.

Este problema se resuelve mediante un modelo de elementos de frontera (*boundary-element mode*, BEM), en donde se considera a la cabeza como un conductor homogéneo con forma de cerebro dividido en distintos elementos. Se asume la conductividad homogénea dentro de las capas del modelo, pero no a través de las fronteras que le

dividen. De esta forma se establecen tres superficies con propiedades distintas que representan la transición entre cuero cabelludo-aire, el cráneo-cuero cabelludo y el cráneo-cerebro. Estas son obtenidas mediante el uso de MRI.

Los métodos numéricos que se utilizan para calcular el modelo de conducción de volumen pasan por calcular una matriz de transferencia, de proyección *lead-field matrix* (LFM). Esta representa la relación lineal entre las fuentes y las medidas de la señal, es decir, la sensibilidad de cada uno de los sensores a la distribución de las fuentes. Para calcular esta matriz existen diferentes herramientas como *Fieldtrip* o *OpenMEEG*.

Por lo tanto, el potencial de un electrodo será la suma ponderada de la actividad de cada una de las fuentes tratadas como dipolos y viene dado por [18]:

$$\Psi_{EEG}(canal, t) = P \cdot S \qquad \text{Eq. 4.2.}$$

Donde, S es la actividad neuronal fuente y P es la matriz de transferencia, $P \in \mathbb{R}^{s\,x\,l}$, siendo s el número de sensores y l el número de regiones cerebrales tratadas como fuentes.

Cálculo de MEG

Este caso va a ser análogo que el anterior de EEG, pero esta vez la distribución de dipolos de corriente electica generada por la actividad neuronal va a provocar la formación de campos magnéticos que son los que va a medir MEG. Por lo tanto, también va a ser necesario el cálculo de una matriz de transferencia que represente la contribución de cada una de las fuentes al campo magnético capturado por los sensores. La cabeza también se va a modelar como un conductor.

La implementación consiste en calcular el campo magnético en el punto r, fuera del conductor, producido por un dipolo Q en r0 [18].

Cálculo iEEG

EEG intracortical es una técnica de electroencefalografía en la que se colocan los electrodos directamente en la superficie del cerebro. Son utilizados en la clínica, por ejemplo, para determinar la localización focal de crisis epilépticas, que en algunos casos puede requerir de resección quirúrgica. De forma análoga a los casos anteriores, su implementación en TVB consiste el cálculo del potencial en el electrodo localizado en un punto producido por un dipolo caracterizado por su momento Q, siendo r-r0 la distancia entre el electrodo y el dipolo y σ la conductividad del medio [18]:

$$\frac{1}{4\pi\sigma} Q \frac{(r - r_0)}{|r - r_0|^3} \qquad \text{Eq. 4.3.}$$

Cálculo señal BOLD fMRI

La señal BOLD capturada en fMRI mide la actividad neuronal que se produce en cierta región del cerebro, donde el aumento de la actividad de las neuronas va a provocar un aumento en el consumo de oxígeno, produciéndose un desbalance entre la hemoglobina oxigenada y desoxigenada, que al ser la primera diamagnética y la segunda paramagnética va a poder ser capturado por la técnica fMRI. Si existiese correlación de esta señal entre dos regiones diferentes se puede decir que están funcionalmente relacionadas [52].

Sin embargo, existe un gran debate sobre cual es exactamente la relación entre la señal BOLD y la actividad neuronal. Se ha demostrado que distintos procesos pueden afectar de manera diferente a la señal [51].

El cálculo de la señal BOLD se obtiene de aplicar a la actividad neuronal a lo largo del tiempo, obtenida en la simulación, un filtro de respuesta finita (FIR). Recordemos que un filtro FIR es aquel cuya respuesta al impulso tendrá un número finito de términos no nulos. Cuando se emplea un filtro FIR, en realidad lo que se está haciendo es una operación de convolución en el tiempo entre una señal y la respuesta al impulso de dicho sistema. En nuestro caso dicha señal será la actividad neuronal simulada y la respuesta al impulso será la función de respuesta hemodinámica (*Haemodynamic Response Function*, HRF), que representa la forma en que dicha actividad afecta al flujo de la sangre y a los niveles de hemoglobina. Una de las características de HRF es que es invariante en el tiempo [18].

TVB ofrece hasta cuatro posibles HRF basadas en distintos estudios, teniendo cada una, una serie de parámetros característicos [18]:

- Función Gamma
  Estimado de estudios de contraste en la región V1

$$h(t) = \frac{\frac{t^n}{\tau} \exp\left(-\frac{t}{\tau}\right)}{\tau(n-1)!} \qquad \text{Eq. 4.4.}$$

  $\tau = 1.08\ n = 3$

- Mezcla de osciladores amortiguados

$$h(t) = exp\left(\frac{t}{\tau_1}\right)\sin(2\pi f_1 t) - a \cdot exp\left(\frac{t}{\tau_2}\right)\sin(2\pi f_2 t) \qquad \text{Eq. 4.5.}$$

  τ_2 = 7.4 s, f_2 = 0.12 Hz, a = 0.1, τ_1 = 7.22 s, f_1 = 0.03 Hz.
  a ∈ [0.1–0.12]; τ_1 ∈ [7.22–7.27] f_1 ∈ [0.03–0.05] Hz

- Mezcla de Gammas
  Diferencia de dos funciones de probabilidad de densidad gamma

$$h(t) = \frac{\lambda t^{a_1-1}\exp(-\lambda t)}{\Gamma(a_1)} - 0.5\frac{\lambda t^{a_2-1}\exp(-\lambda t)}{\Gamma(a_2)} \qquad \text{Eq. 4.6.}$$

- Volterra

  El modelo Balloon–Windkessel describe la interacción entre la perfusión sanguínea y la señal BOLD, la cual será tratada como una función estática no lineal definida por dos variables de estado el volumen (v) y el contenido en desoxihemoglobina (q). La entrada al sistema será la actividad neuronal de una región determinada que provocará un aumento en la señal vasodilatadora que se retroalimenta y provocará también un cambio proporcional al flujo, el cual en última instancia provocará los cambios en las variables de estado volumen de sangre y contenido de desoxihemoglobina. De esta forma, el modelo quedará definido por una serie de ecuaciones diferenciales que describirán todas estas variables [53].
  En [54] se utiliza los Volterra *kernels* como una representación matemática para capturar y modelar las relaciones no lineales entre la actividad neuronal y la respuesta BOLD medida por la fMRI. Estos *kernels* se ajustan a partir de datos experimentales para describir cómo la actividad neuronal se relaciona con la respuesta hemodinámica observada. Las series de Volterra son una extensión de las series de Taylor y se utilizan para describir la respuesta del sistema en función de las entradas pasadas y presentes, y capturar las características no lineales del sistema.

  La señal BOLD se obtiene como:

$$h(t) = V_0 \cdot (k_1(1-q) + k_2\left(1 - \frac{q}{v}\right) + k_3(1-v) \qquad \text{Eq. 4.7.}$$

Se ha demostrado que existe una gran variabilidad entre distintas zonas cerebrales de un mismo sujetos, entre sujetos y entre sujetos pertenecientes a distintas poblaciones, por lo que los valores de los parámetros que caracterizan cada HRF deben ajustarse a los adecuados.

d) Ajuste de los datos simulados con los empíricos

El ajuste de los datos simulados con los empíricos constituiría el último paso para conseguir un modelo de cerebro virtual. Las distintas ecuaciones diferenciales que permiten el cálculo de la dinámica neuronal a gran escala, están determinadas por una serie de parámetros que deben ser definidos previamente, antes de realizar la simulación.

Esta última etapa consiste en averiguar cuál sería el valor más adecuado para estos parámetros.

Uno de los métodos que se pueden utilizar para ello, como se hace en [55] y [56], es el ajuste iterativo de los parámetros para optimizar la concordancia entre los datos simulados y empíricos. Para ello se realizan simulaciones de la red, calculando la señal BOLD a partir de la cual se obtendrá la matriz funcional simulada (sFC). En cada una de las simulaciones se irán variando el valor de los parámetros que quieran ser estudiados. Se define previamente una rango de valores para cada uno de estos y un paso de integración que indicará cuánto varía en cada una de las simulaciones. Posteriormente, se realizará una correlación entre cada una de estas sFC y FC de la señal BOLD empírica medida a través de fMRI (eFC). Aquella sFC que presente una mayor correlación con eFC indicará que la simulación se aproxima más a la realidad y, por lo tanto, se escogerán los valores para los parámetros que se hayan utilizado en esa simulación. Además como se indica en [57] y [58] también se pueden comparar tanto la amplitud como la frecuencia de los datos empíricos y de los simulados.

El rango de valores en el que se va a realizar las simulaciones puede ser elegido a partir de lo revisado en la literatura como se hace en [55] y [56] o puede ser definido a partir de un análisis del espacio de parámetros. En [57] y [58] se realiza calculando la varianza global (varianza media de las series temporales a lo largo de todas las regiones) y representándolo en un mapa de calor. El rango de valores se elige para aquellos que presentan una mayor varianza global. Realizando esta exploración del espacio de parámetros se puede obtener información del grado de variabilidad y sensibilidad que los parámetros tienen sobre las señales simuladas.

4.4.Aplicación clínica

La siguiente cuestión a tratar es cómo puede ser utilizado TVB en la clínica con pacientes reales. La incorporación de datos individuales en el modelo (*data-driven models)* permite crear una red cerebral característica para cada sujeto. Esta condición se enmarca dentro la medicina personalizada [59][60]. En neurología es de suma importancia tratar las enfermedades que afectan a los pacientes de forma individualizada debido a la gran variabilidad que existe entre la lesión propiamente dicha y las consecuencias funcionales que esta conlleva. Por esta razón, es necesario elaborar análisis y tratamientos personalizados, para lo cual TVB se muestra como una herramienta con mucho potencial. TVB permite evaluar distintas estrategias terapéuticas y determinar cuál sería la más adecuada para cada paciente, además de inferir cuales son los mecanismos patológicos a través de los cuales se produce una enfermedad.

Uno de los elementos que son utilizados en la medicina personalizada son los biomarcadores. Los biomarcadores se pueden definir como "una medida clínica sustituta que idealmente refleja los procesos biológicos subyacentes a nivel celular/molecular que se puede utilizar como indicador del estado de la enfermedad"; o como una “amplia subcategoría de signos que pueden medirse con precisión y reproducibilidad” y una “característica que se mide y evalúa objetivamente como un indicador de procesos

biológicos normales, procesos patogénicos o respuestas farmacológicas a una intervención terapéutica" [44]. Estos biomarcadores se deben caracterizar por ser capaces de determinar el avance de la enfermedad o la respuesta al tratamiento, contener información clínica relevante y estar asociado a mecanismos de la vía patológica.

TVB utiliza modelos que cuentan con variables de estado y otros parámetros a los que se les puede otorgar una interpretación biológica. Las investigaciones que se llevan a cabo consisten en definir estos parámetros biofísicos como biomarcadores que puedan caracterizar el estado de una enfermedad en un paciente concreto y ser utilizado como un sistema de ayuda al diagnóstico. Debido a la multiescalaridad a la que opera TVB, se puede obtener parámetros globales y locales.

TVB ha sido aplicado en pacientes que sufren distintas enfermedades. A continuación, se va a realizar una revisión de distintos artículos, agrupados por enfermedades, en los que se ha utilizado.

a) Epilepsia

La epilepsia se caracteriza por la perturbación de la dinámica cerebral que se origina en una red local o zona epileptogénica y desde ahí se extiende a otras regiones o zona de propagación [61]. Uno de los tratamientos que se realizan a pacientes epilépticos refractarios a fármacos es la resección quirúrgica de la zona epileptogénica, sin embargo su localización puede resultar muy difícil. Para resolver este problema en [62] se describe la caracterización del Paciente Epiléptico Virtual (VEP) a través de TVB. Para ello, se incluye datos empíricos y se usa como modelo de masa neuronal el *Epilleptor*. La aplicación de VEP puede mejorar los resultados de la cirugía indicando cual sería el lugar para la mejor colocación de los electrodos SEEG; delimitando de una forma más precisa, a través de diversas simulaciones, la zona epileptogénica; y evaluando distintas estrategias quirúrgicas.

Otra de las mejoras que se pueden conseguir al realizar este proceso quirúrgico es tratar de reducir lo máximo posible la zona de extirpación. En [61] se hace una simulación de cómo se propaga la perturbación desde la zona epileptogénica hasta las zonas de propagación identificando las vías más inestables a través de las que este fenómeno se produce. Realizando un estudio de la red se propone un método de resección quirúrgica menos invasivo y más efectivo, interrumpiendo solamente aquellas vías nerviosas que permiten que se propague una convulsión.

b) Ictus

El ictus o accidente cerebrovascular (ACV) es una afección cerebral que ocurre cuando este se queda sin suministro de sangre, bien causado por la obstrucción de un vaso o una hemorragia. Esta situación puede comprometer la supervivencia del tejido afectado. En [58] se trata de encontrar biomarcadores indicativos de ictus o de su recuperación a partir de los parámetros biofísicos con los que trabaja TVB, buscando diferencias entre los

pacientes sanos y los que hayan recibido un ataque de este tipo. Se encontraron diferencias tanto en parámetros globales como en locales. Entre los parámetros globales destaca la disminución de la velocidad de conducción y el aumento en el acoplamiento global en la mayoría de los pacientes. Para la simulación a nivel local se utilizó el modelo Stefanescu-Jirsa3D, cuyo parámetro K_{21} que representa la dinámica inhibitoria local mostraba una disminución. Además se buscó predictores potenciales de recuperación, comparando diferentes parámetros antes y después de terapias de recuperación, encontrando una correlación negativa entre la excitación local y la recuperación motora; además de una correlación positiva entre la dinámica local y la recuperación motora.

Por otro lado, en [57] se realizó un análisis de grafos con el objetivo de vincularlo al acoplamiento de largo alcance, un parámetro ofrecido por la simulación en TVB. Las tres medidas de grafos calculadas fueron *degree centrality* (número de nodos adyacentes al nodo i promediado entre el número de nodos de la red), *betweenness centrality* (probabilidad de que un camino corto pase por un nodo), and global *efficiency* (número mínimo de aristas atravesadas para conectar un nodo con otro). Fue con esta última con la que se encontró una correlación negativa con el acoplamiento a largo alcance.

Por último, en [59] se estableció la terapia virtual. La terapia virtual, (*The Virtual Therapy*, TVT), se refiere a un enfoque terapéutico que utiliza simulaciones por computadora para influir en la dinámica cerebral y potencialmente mejorar la recuperación después de un ACV u otra lesión cerebral. La terapia virtual busca restaurar los valores de ciertos parámetros cerebrales que se ven afectados después de un ACV y pueden ser susceptibles de ser marcadores de esta enfermedad, como el acoplamiento global, la velocidad de conducción y el acoplamiento entre las poblaciones inhibitorias y excitatorias. Esta consiste en volver a simular las redes cerebrales de pacientes que habían sufrido un ictus, pero esta vez dando a estos parámetros valores que se asemejan a la normalidad. Se obtenían las matrices de conectividad funcional antes y después de la terapia virtual y se comparaban con las matrices empíricas de conectividad funcional de pacientes sanos. Se mostró que en la gran mayoría de pacientes hubo un restablecimiento hacia la normalidad de la dinámica neuronal.

c) Alzheimer

La enfermedad del Alzheimer es una enfermedad neurodegenerativa que afecta principalmente la memoria y otras funciones cognitivas. Se caracteriza por la acumulación anormal de una proteína llamada beta-amiloide en el cerebro. En [63] se usa TVB para modelar cómo la enfermedad del Alzheimer afecta la dinámica neuronal, teniendo por objetivo uno de las principales finalidades de TVB, inferir mecanismos patológicos a través de distintas escalas. Para ello el modelo de masa neuronal utilizado es Jansen-Rit, que incluye varios parámetros que describen las conexiones sinápticas entre distintas poblaciones y otros que reflejan la acumulación de dicha proteína, la cual se obtiene de forma individual para cada paciente a partir de imágenes PET. A mayor acumulación, estos parámetros inducirían en el modelo una mayor excitabilidad. Así, se

puede explorar el efecto de esta proteína a diferentes escalas, vinculando la inhibición sináptica que esta produce con un efecto observado a gran escala como el enlentecimiento del EEG simulado.

Por su parte, en [56] se realiza una búsqueda de parámetros biofísicos del modelo que puedan ser usados como biomarcadores indicativos de la severidad de la enfermedad. Los parámetros que se estudiaron fueron globales (velocidad de conducción, acoplamiento global) y locales del modelo Wong Wang reducido. Se realizaron simulaciones de la red completa, la subred límbica y de esta última integrada en la primera. Se observó que estos parámetros se correlacionaban con diferencias en la cognición de cada individuo y eran mejores predictores de la cognición que los datos estructurales y funcionales empíricos.

d) Terapia de estimulación cerebral

Uno de los tratamientos que se aplican en pacientes que sufren trastornos neurológicos como epilepsia o Parkinson es la terapia de estimulación cerebral. Esta terapia la conforman un conjunto de técnicas de neuromodulación que han demostrado resultados positivos en pacientes, pero todavía se están investigando el porqué de estas mejoras. Estas investigaciones se pueden realizar mediante TVB. Existen distintos tipos de terapias de estimulación. A continuación se explican algunas de ellas y cómo pueden ser simuladas con TVB.

La estimulación cerebral profunda (*Deep Brain Stimulation*, DBS) es una técnica de neuromodulación que consiste en la implantación de electrodos en el cerebro que producen impulsos eléctricos con el objetivo de corregir las dinámicas anormales. Para investigar cuáles son los mecanismos por los que actúa en [64] se hace uso de la plataforma TVB y se indica cómo pueden participar distintas estructuras subcorticales. Las simulaciones obtenidas son biológicamente verosímiles tanto en estado de reposo como en DBS. También se indica que TVB puede ser útil para valorar distintas estrategias a la hora de colocar los electrodos, puesto que la posición de estos puede resultar determinante en el éxito del tratamiento.

Otra técnica de neuromodulación es la estimulación transcraneal de corriente continua (tDCS, *Transcranial direct current stimulation*). Esta es una técnica no invasiva, en la que se aplica un flujo de corriente continua de hasta 2 mA a través del cerebro entre electrodos. En [65] se realizan simulaciones con TVB en estado de reposo y aplicando tDCS para comparar los resultados. En este estudio se demuestra la relación entre la estimulación y la organización funcional del cerebro.

Por último la estimulación magnética transcraneal (TMS) utiliza pulsos magnéticos suaves para estimular determinadas zonas del cerebro. La simulación de experimentos con TVB de TMS en [66] se hace en dos estados dinámicos distintos: estados asincrónicos (vigilia) y de onda lenta (sueño). Se observó que la respuesta en las ondas lentas es fuerte y localizada, pero en los estados asincrónicos la respuesta es más débil y generalizada.

e) Envejecimiento

El envejecimiento se puede manifestar al realizar un análisis de la complejidad de las series temporales de fMRI. Una medida de complejidad es la entropía multiescalar (MSE) que puede ser utilizada como biomarcador, puesto que se ha demostrado que los cerebros jóvenes y sanos están descritos por series temporales más complejas, mientras que aquellos que presentan algún tipo de patología se caracterizan por una mayor regularidad. En [67] se realizan simulaciones de fMRI utilizando TVB y se muestra que existe una disminución de la complejidad, medida con MSE, en aquellos adultos de mayor edad que presentan un decremento en la conectividad estructural. Por lo tanto, este tipo de medidas también pueden resultar útiles para vincular el conectoma estructural con la dinámica cerebral.

f) Predicción de efectos de la resección tumoral

La resección tumoral es una técnica quirúrgica que consiste en retirar tejido dañado. Para ello, es esencial realizar un plan prequirúrgico que delimite correctamente las zonas de interés, de esta forma mantener el tejido sano lo máximo posible y preservar las funciones más esenciales. Tradicionalmente este plan se ha realizado a través de técnicas de neuroimagen no invasivas, sin embargo estas presentan limitaciones puesto que no se puede estimar cual seria el resultado funcional tras la intervención. Por esta misma razón, en [68] se utiliza TVB con el objetivo de combinar estas técnicas de neuroimagen con modelos biofísicos de la dinámica cerebral. En este estudio se demostró que los modelos individuales presentan una mejora significativa en la capacidad de predecir la conectividad funcional simulada; que parámetros del modelo son capaces de diferenciar entre regiones directamente afectadas por un tumor cerebral, regiones más distantes de un tumor cerebral y regiones de un cerebro sano; y que existe una correlación entre los parámetros del modelo optimizados individualmente, la topología de la red estructural y el desempeño cognitivo.

En un trabajo posterior [69] se trató de evaluar las diferencias de los pacientes pre y postcirugía, además de evaluar cómo cambian los parámetros del modelo realizando una "cirugía virtual" eliminando regiones de la simulación que se retiran durante la resección. Los resultados mostraron que los parámetros del modelo de red cerebral se mantienen relativamente estables a lo largo del tiempo en pacientes que se sometieron a la resección de un tumor cerebral, en comparación con la variabilidad inicial observada en sujetos sanos sin condiciones médicas. También se observó que la conectividad funcional obtenida de la simulación de la cirugía virtual tiene una mayor similitud con la conectividad funcional real observada después de la cirugía en algunos pacientes, mientras que en otros hubo una mejora insignificante. Sin embargo, según se menciona en este trabajo existen bastantes limitaciones que implican una necesaria investigación futura en esta dirección.

4.5.Extensión multiescalar de TVB

La microescala es el nivel en el que TVB no profundiza, debido a que con el uso de masas y campos neuronales medios asume la actividad de grupos neuronales en su conjunto. Con el fin de comprobar cómo cambios en la microescala afectan a nivel macro, se están incorporando modelos neuronales a las simulaciones de TVB. Se debe destacar el marco de trabajo TVB-*multiscale* que se encuentra en desarrollo. Son una serie de *toolboxes* desarrolladas en Python que permiten la co-simulación en la que participan dos softwares sincronizados, TVB como simulador macroescalar y otro como microescalar que puede ser NEST, ANNarchy o NetPyNe [70] https://github.com/the-virtual-brain/tvb-multiscale

a) TVB – AdEx

En [71] la integración multiescalar se realizó incorporando en TVB modelos de campo medio de neuronas exponenciales adaptativas (AdEx). Estas poblaciones neuronales son conectadas siguiendo el conectoma humano. Se demuestra que surge una dinámica a gran escala a partir de adaptaciones a escala microscópica, semejante a la del cerebro humano.

El modelo desarrollado en este artículo tiene tres componentes. El primero es el modelo AdEx, con el que se puede reproducir la actividad de dos tipos de neuronas: excitatorias regulares (RS) e inhibitorias rápidas (FS). Las neuronas AdEx tienen la capacidad de integrar entradas sinápticas y disparar potenciales de acción, seguidos de un período refractario. Las redes de neuronas AdEx tienen una serie de propiedades que les permite mostrar estados asincrónicos o sincrónicos, semejantes a los que se producen en el cerebro en estados de vigilia y sueño profundo. El segundo componente es el modelo de campo medio derivado de simulaciones de redes AdEx. Este está compuesto por dos poblaciones neuronales, RS y FS, y describe las tasas de activación de cada una de estas. El tercer componente es la integración de estos modelos de campo medio basados en AdEx en TVB, siguiendo los datos de tractografía humana. De esta forma se simula actividad a nivel macroscópico emergente de niveles inferiores.

En este artículo se demostró que cambios en el modelo a escalas microscópicas van a repercutir en el surgimiento de los patrones medidos a escala macroscópica. Además, este modelo tiene la capacidad de reproducir las características esenciales de los estados de vigilia y sueño profundo en el cerebro humano, tanto en la actividad espontánea como en la evocada por estímulos externos.

b) TVB-ANNarchy

En [64] se desarrolla una co-simulación TVB-ANNarchy, que es un software que simula neuronas del tipo *spiking*. La red diseñada en ANNarchy se compone de ocho poblaciones neuronales con neuronas tanto excitatorias como inhibitorias modeladas según el modelo Izhikevich. El acoplamiento entre los dos simuladores se produce mediante el concepto de nodo “proxy”. Este puede ser un nodo “proxy estimulante”, el cual recibe la actividad de campo medio de un nodo de TVB y la transforma en trenes de impulsos

correlacionadas que estimulan a una población neuronal de ANNarchy. Por otro lado, los nodos "proxy" de salida registra los impulsos generados por una población neuronal de ANNarchy y las convierte en una actividad de campo medio que se envía a un nodo de TVB. De esta forma los dos simuladores están acoplados entre sí, generando una actividad cerebral a lo largo de distintas escalas.

En este artículo se implementa un modelo de ganglios basales basado en neuronas de impulso y se desarrolla una co-simulación TVB-ANNarchy. El modelo permite explorar los efectos de la estimulación cerebral profunda (DBS) en diferentes regiones del cerebro, tanto a nivel local como global, y predecir los posibles resultados clínicos para pacientes individuales. Además, el modelo muestra que la DBS virtual puede mover las tasas de disparo del circuito tálamo-ganglio basal hacia el régimen saludable y alterar la actividad en regiones corticales distribuidas, especialmente en las regiones frontales.

c) TVB- NEST

Las co-simulaciones TVB-NEST también son posibles. En este caso, como se plantea en [72], este tipo de simulación se da mediante el trabajo conjunto de cinco módulos: dos simuladores (TVB y NEST), un lanzador y dos módulos de transferencia. Estos módulos de transferencia son esenciales puesto que conectan los simuladores, transfiriendo datos entre las escalas y adaptando los retrasos de comunicación durante la simulación. Cada módulo de transferencia consta de interfaces específicas para cada simulador y un transformador independiente que realiza la función de transformación necesaria para lograr la coherencia en la simulación.

La ejecución de esta co-simulación se realiza en varias etapas. Primero el lanzador prepara la simulación, permitiendo que haya coherencia. Posteriormente se realiza la simulación propiamente dicha, donde los relojes de tiempo de los simuladores se sincronizan usando mensajes asincrónicos. En cada paso de sincronización, los simuladores reciben datos de entrada y simulan el siguiente paso. Por último, se da la terminación donde la simulación concluye de manera autónoma por los simuladores.

En este caso, se desarrolla un modelo de la región del hipocampo CA1 a nivel celular del ratón integrado en una red cerebral completa. Este modelo compuesto a varios niveles permite el registro de componentes a nivel macro y micro. Por ejemplo, a nivel microscópico se pueden obtener medidas como el voltaje de membrana de las células, o potenciales de campo medio. Por otro lado a nivel macro, se puede obtener la tasa de disparo de la red o electroencefalografía intracraneal. Esto demuestra la posibilidad que ofrecen este tipo de co-simulaciones de poder realizar un análisis a diferentes escalas y observar como una puede afectar a la otra.

4.6.Otros simuladores macroescalares

Como ya se comentó en la introducción de este apartado, existen más simuladores cerebrales a parte de TVB que tienen por objetivo reproducir la actividad del cerebro en

su conjunto. Estos simuladores utilizan otras técnicas para alcanzar dicho propósito que deben ser comentadas.

El primero de ellos, que ya ha sido mencionado varias veces a lo largo de este trabajo es *The Blue Brain Project* [73]. Este programa tiene como finalidad la reconstrucción digital biológicamente detallada y simulación del cerebro de un roedor y en última instancia del humano. La estrategia empleada por este software es la fuerza bruta [45], es decir, pretende describir modelos detallados de circuitos neuronales, en los que estén incluidos características propias de las neuronas como su morfología tridimensional o la distribución de canales iónicos. Esto hace que el modelo cerebral este conformado por un gran número de neuronas y las sinapsis que las conectan. Para procesar toda esta información es necesario el uso de supercomputadoras. Estas se caracterizan por su gran capacidad de almacenamiento y poder procesamiento. Este enfoque es completamente distinto al de TVB, puesto que este último no emplea modelos detallados, sino modelos de campo medio y, además, sus simulaciones pueden realizarse en ordenadores personales más disponibles para el usuario.

Neurogrid [74] es otro de estos simuladores que trata de explicar cómo surgen los comportamientos cognitivos a partir de grandes redes altamente interconectadas. Defiende que para la comprensión de este fenómeno es necesario realizar simulaciones a gran escala, pero de redes biofísicamente realistas. Por eso, al igual que en Blue Brain, los modelos empleados están compuestos por millones de neuronas. Sin embargo, esta plataforma si realiza una división más clara de las distintas escalas y es capaz de realizar simulaciones que abarcan cinco niveles: biofísico, dendrítico, neuronal, columnar y de área.

El último simulador que se va a mencionar es BrainX3 [75]. Esta es una herramienta que permite la exploración, análisis, modelado y simulación de datos cerebrales a través de visualizaciones interactivas de fácil uso para los investigadores. Es capaz de generar simulaciones de la actividad cerebral a partir de modelos neuronales que incluyen datos empíricos. Sin embargo, esta característica no es la más relevante de esta plataforma, sino que se centra más bien en la integración de datos multimodales y la visualización de estos.

5. Trabajo experimental TVB

5.1.Disposición de los paneles de la GUI

La GUI de TVB se caracteriza por su sencillez y fácil manejo para un investigador que no posea altos conocimientos de programación. En la barra inferior se dispone de las seis secciones principales en las que se encuentra dividida la GUI (**Fig. 5.1**). Cada una de estas están diseñada para realizar una función específica que, en última instancia, permita realizar las simulaciones. Estas secciones son:

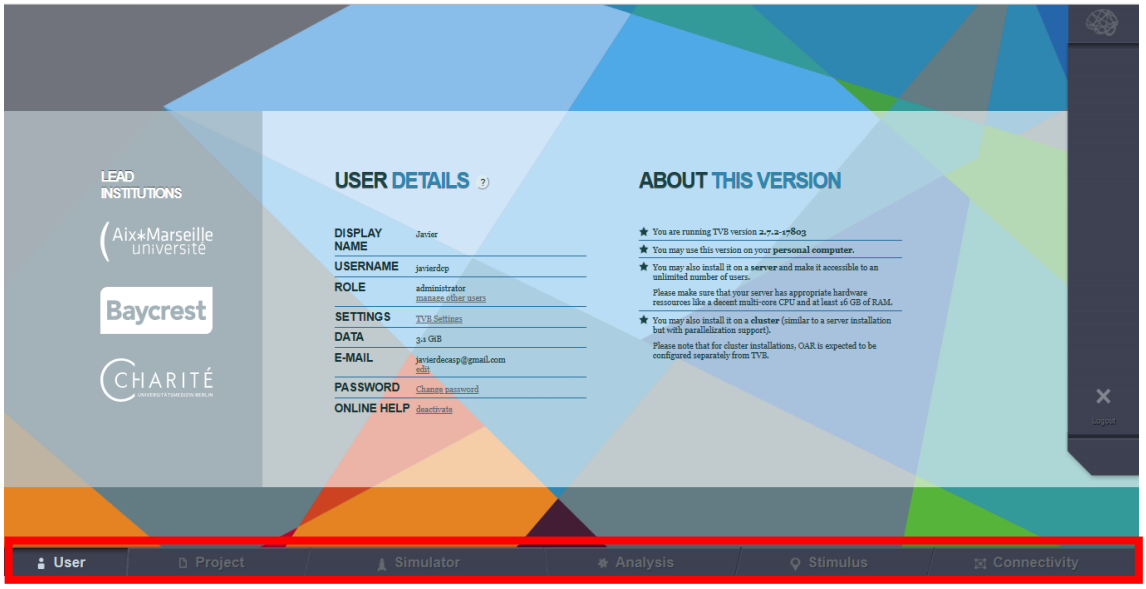


Figura 5.1. Pantalla de usuario de TVB. Se resalta dentro del cuadrado rojo las seis secciones distintas en las que se divide la GUI.

- **Usuario.** Se encuentran los datos básicos del usuario, así como características sobre la versión de TVB.

- **Project.** Este es un apartado importante puesto que es desde donde se van a administrar los diferentes proyectos de un usuario. Estos pueden ser enumerados en una lista en donde se va a obtener una pequeña información de estos. Dentro de cada proyecto se van a distinguir dos elementos principales: las operaciones y la estructura de datos.
  Las operaciones son los distintos procesos que han sido ejecutados por TVB (simulaciones, creación de datos de conectividad…). Todos estos son almacenados en el historial y se puede ver, en cualquier momento, los resultados que se han obtenido. TVB establece un sistema de colores asignando uno a cada estado de la operación: verde finalizada, rojo inacabada debido a algún error, azul en ejecución, amarillo pendientes y gris cancelada.
  La estructura de datos permite visualizar como están estos organizados dentro del proyecto. A su vez se pueden distinguir dos tipos de datos principales, aquellos que se deben incluir en la creación del proyecto y que son necesarios para ejecutar cualquier operación; y los datos que se obtienen como resultado de dichas operaciones. Para cada tipo de datos también existe un código de colores: verde

son los datos "crudos" (conectividades, superficie, sensores…), amarillo son datos adyacentes como anotaciones, rojo son series temporales, azules son resultados de análisis y morados algunos datos creados por TVB (estímulos, conectividades locales…).

- **Simulator.** Esta es la sección principal desde donde se van a realizar la configuración de la simulación, indicando los datos con los que se va a trabajar y distintos parámetros. El manejo de esta sección será explicado a lo largo de este capítulo, donde se indicará como realizar distintos tipos de simulaciones.

- ***Analysis*.** Desde aquí se van a poder ejecutar distintas operaciones a partir de los datos obtenidos mediante la simulación, que van a permitir, como el propio nombre indica, realizar un análisis más profundo de los resultados. Estos *"analyzers"* también van a ser explicados a lo largo de este capítulo a medida que se vayan realizando las simulaciones.

- ***Stimulus*.** Aquí se van a poder diseñar los distintos estímulos, o bien aplicados sobre regiones o sobre la superficie, que van a poder ser aplicados posteriormente en la simulación. Esto será explicado en el apartado 5.7.

- **Conectividad.** Desde aquí se van a poder manejar los diferentes datos de conectividad sobre los que se van a realizar las simulaciones. Esto incluye desde una simple visualización, hasta una manipulación de los datos y modificación de estos. Esta será explicada en el apartado 5.3.

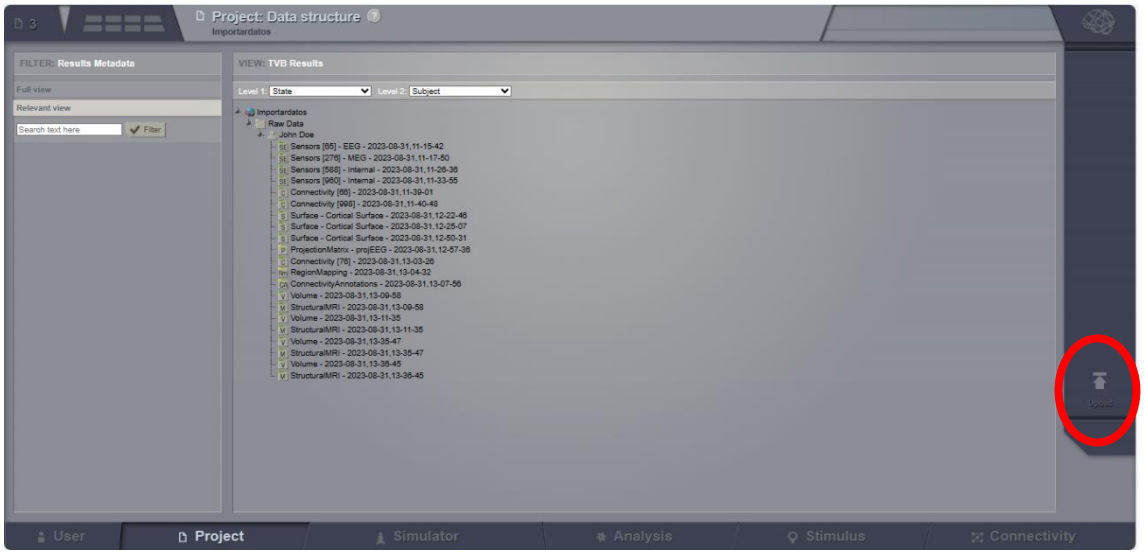
5.2.Importar datos

Las simulaciones en TVB se van a realizar sobre una serie de datos que deben ser almacenadas en el proyecto previamente. Estos datos pueden se pueden obtener de las mediciones que realice un investigador para un paciente en concreto, o de la web. En este ejemplo se importarán los datos desde la plataforma de GitHub de TVB, https://github.com/the-virtual-brain/tvb-data .

Para importar datos se accederá al subapartado *DataStructure* de la pantalla Project y se pulsará en el botón de la derecha "*Upload*" (**Fig. 5.2. (A)**). Una vez allí, aparecerá un cuadro de diálogo (**Fig. 5.2. (B)**)., en el cual a la izquierda aparecerá que tipo de dato queremos importar (conectividad, superficie, matriz de proyección…) y en qué formato. Es recomendable subir primero las estructuras de conectividad, superficie y sensores, puesto que estos se pueden subir directamente, sin necesidad de estar ligados a otros datos. Por otro lado, otros tipos de datos como la matriz de proyección o el mapeo de regiones sí están enlazados a otros tipos de datos. Por ejemplo, la matriz de proyección, como ya se comentó en 4.3. c), sirve para relacionar las distintas fuentes o regiones del cerebro con los sensores de EEG/MEG/iEEG. Por esto, cuando vaya a ser incorporado se debe indicar con que conectividad y con que sensor está ligado. Además debe existir una coherencia, es decir, el número de regiones y de sensores de la matriz de proyección para los que está diseñada, debe coincidir con la superficie y los datos de los sensores

propiamente dichos. Por ejemplo, para importar la matriz de proyección *"projection_eeg_65_surface_16k"* se debe especificar que está ligada a los datos de sensores de EEG que presenta 65 puntos de referencia y a la superficie que cuenta con 16000 nodos. Lo mismo ocurre con el mapeo de regiones que debe coincidir el número de nodos de la conectividad a gran escala y el número de nodos de la superficie con lo que se especificaría para el archivo en concreto.

A



B

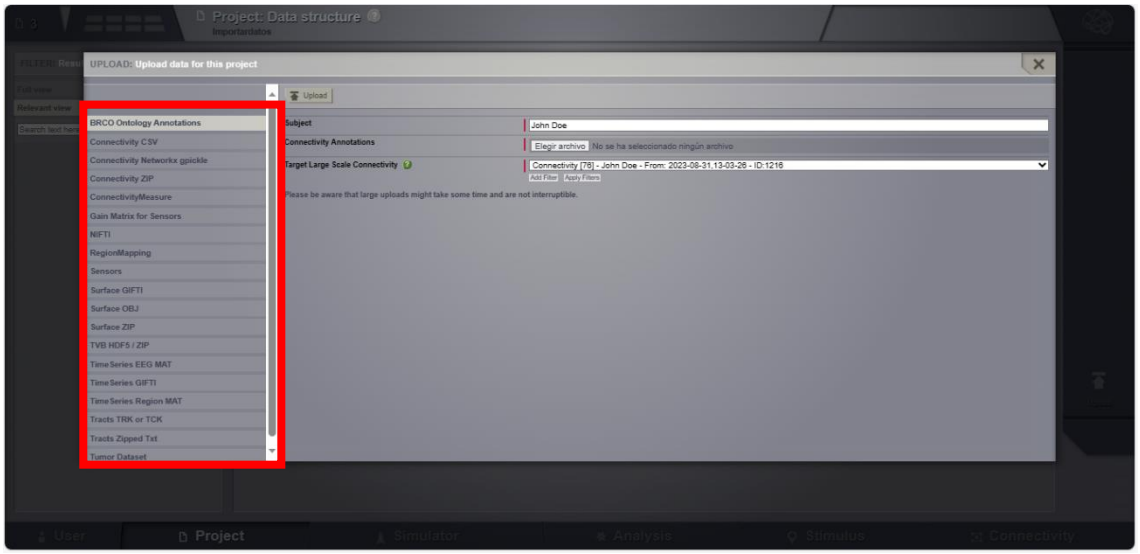


Figura 5.2. Importar datos. (A) Botón de selección para importar. (B) Cuadro de diálogo donde elegir qué tipo de datos se quiere importar.

5.3.Conectividad

a) Conectividad a gran escala

La conectividad a gran escala o de largo alcance describe las conexiones entre las diferentes regiones cerebrales (Ver 4.3. b)). En esta sección la pantalla quedará dividida en dos mitades, la izquierda contiene varios modos de visualización de la conectividad y la derecha el editor de las matrices de conectividad (**Fig.5.3.**).

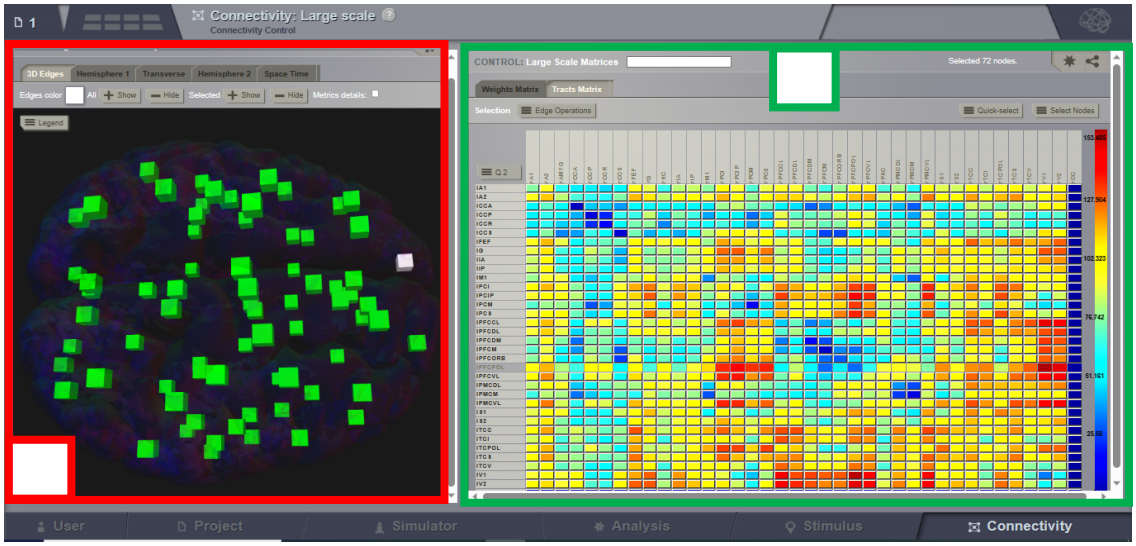


Figura 5.3. Conectividad gran escala. **(A)** Visualización de la conectividad. **(B)** Editor de matrices de conexión

El editor de matrices permite fácilmente cambiar los valores deseados para ajustarlos a los requisitos necesarios para el experimento que se vaya a realizar. En un código de colores se muestra la fuerza de dichas conexiones. Se pueden observar la matriz de pesos y matriz de tramos de longitud. Tan solo se va a mostrar un cuadrante de la matriz, pudiendo seleccionar cual se quiere visualizar. Los cuadrantes 1 y 4 representan conexiones intra-hemisféricas y los 2 y 3 inter-hemisféricas.

Para hacer modificaciones es posible realizar operaciones sobre los nodos seleccionados. Las operaciones disponibles son: indicar el número directamente (*set*), sumar (*add*), restar (*decrease*), multiplicar (*multiply*) y dividir (*divide*). Para aplicar este tipo de operaciones es necesario indicar sobre que conexiones se va a hacer a partir de los nodos seleccionados. Van a existir dos conjuntos de nodos: los seleccionados y los no seleccionados. Recordando que las columnas indican desde donde parte la conexión y las filas hacia dónde va esa conexión, a partir de estos conjuntos se determinan las cuatro categorías de nodos:

- *In → Out:* aristas que conectan los nodos del conjunto seleccionado (filas) con los nodos del conjunto no seleccionado (columnas).
- *In → In:* aristas que conectan los nodos del conjunto seleccionado.
- *Out → In*: aristas que conectan los nodos del conjunto no seleccionado (filas) con los nodos del conjunto seleccionado (columnas).
- *Out → Out*: aristas que conectan un par de nodos en el 'conjunto no seleccionado.

Otra forma de editar la matriz es seleccionando directamente una conexión y escribir el valor que se le quiera dar. También es posible, seleccionando una conexión, eliminar directamente todas las conexiones que parten o que se dirigen a cierto nodo.

En esta sección también están disponibles una serie de *visualizers* que permite ver gráficamente cómo se disponen las conexiones (**Fig. 5.4.)**.

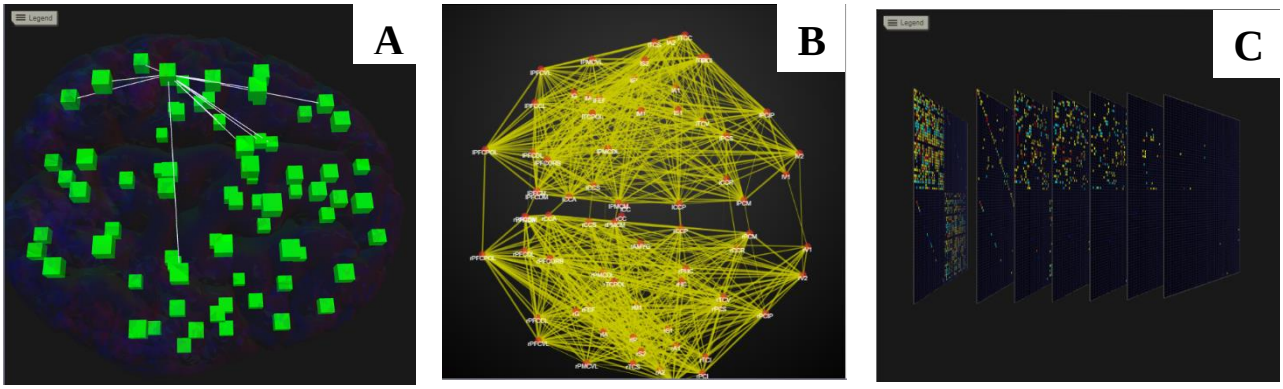


Figura 5.4. Visualizers de la conectividad disponibles. **(A)** Visualización en 3D de los nodos en el espacio, las líneas en blanco indican las conexiones que parten del nodo IS1. **(B)** Representación transversal en 2D de las matrices de conectividad. **(C)** Representación de la estructura espacio temporal y cómo varía la matriz de pesos en el tiempo.

b) Conectividad a escala local

Este tipo de conectividad va a ser utilizada en las simulaciones basadas en superficie (ver 5.4 b)). En esta sección la pantalla queda dividida en dos **Fig. 5.5. (A).** A la izquierda se encuentra la configuración del kernel, donde se puede establecer su tipo (Gauss, sombreo mexicano y sigmoide), así como sus parámetros que van a definir la conectividad local, es decir, como interactúa un nodo con sus contiguos. También se puede especificar la distancia de corte, distancia a la cual el valor del kernel se va a hacer cero. En la izquierda se puede observar la gráfica que describe al kernel y cómo se va modificando junto con sus parámetros. Además, se puede visualizar la conectividad local sobre la superficie (círculo azul), seleccionando un nodo en concreto y viendo su fuerza de conexión con los nodos contiguos. Por ejemplo, se creó un kernel del tipo sombrero mexicano con los valores reflejados en la **tabla 5.1.**

Tabla 5.1. Configuración kernel de conectividad local

Surface			Surface - Cortical Surface - John Doe		
Spatial			*Mexican-Hat*		
Amp_1	0,5	**Sigma_1**	20	**Midpoint_1**	0
Amp_2	1,00	**Sigma_2**	10,0	**Midpoint_2**	0,0
***Cutoff distance* (mm)**			40		
Display name			*Mexican-hat-surface*		

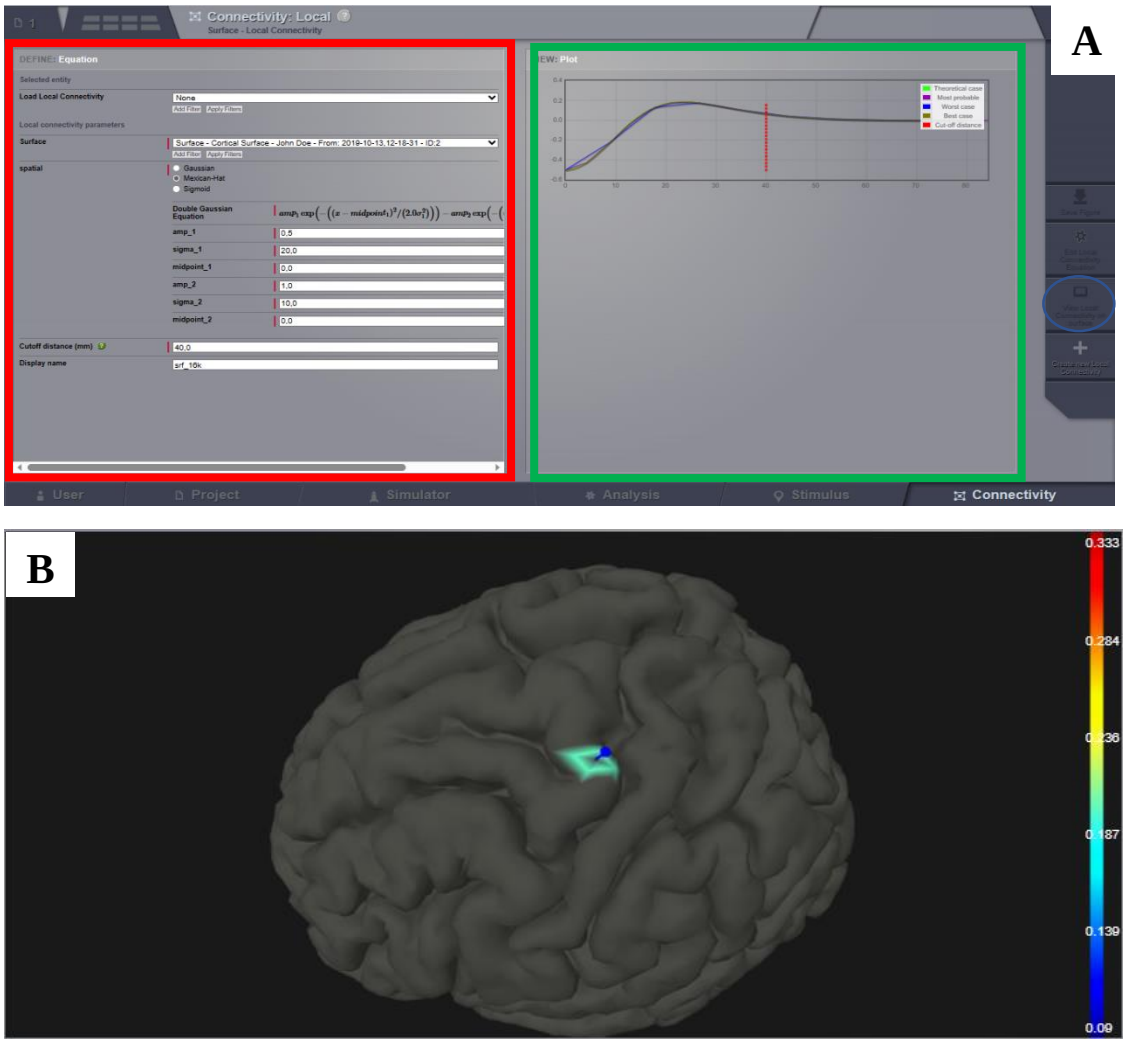


Figura 5.5. Conectividad a escala local. **(A)** Panel de configuración de la conectividad local, en rojo configuración de parámetros y en verde la gráfica del kernel. **(B)** Representación en 3D de un nodo y sus conexiones de la conectividad local.

Cuando se haya realizado la configuración, se pulsa sobre *Create new local connectivity* y se lanza una operación del tipo *LocalConnectivityCreator*.

5.4.Simulaciones básicas

a) Basadas en regiones

Las simulaciones basadas en regiones son aquellas que consideran cada región cerebral como una población neuronal, de forma que la red estará compuesta por tantos nodos como regiones sean consideradas en las matrices de conectividad con las que se vayan a realizar la simulación (Ver 4.3. b)). A continuación se van a describir los diferentes campos que deben ser completados para configurar una simulación, junto con los comandos introducidos para realizarla **(tabla 5.2.)**.

- *Long-range connectivity*: se debe elegir el conectoma con el que se quiere realizar la simulación de todos los disponibles en el proyecto.

- *Conduction speed*: velocidad a la que se propaga la señal generada en cierto nodo a lo largo del conectoma, desde donde van a surgir los retrasos temporales. Está medida en mm/ms.
- Función de acoplamiento: elegir qué función se quiere utilizar. La función de acoplamiento se usa para reescalar la actividad proveniente de otra región antes de introducirla en el modelo local. Posteriormente se indican el valor de los parámetros, los cuales van a ser distintos dependiendo de la función de acoplamiento utilizada.
- Cortical Surface: especificar con que superficie cortical se quiere modelar para las regiones basadas en superficie. En este caso no será utilizada.
- *Spatiotemporal stimulus*: estímulo que se quiera introducir en el modelo. En este caso no será utilizado.
- Local Dynamic *model*: modelo que definirá la dinámica de cada uno de los nodos de la red. Posteriormente se pedirá especificar el valor de cada parámetro del modelo. Por último se pide que variables del modelo van a ser registradas por los monitores.
- *Integration scheme*: es el método numérico a partir del cual el sistema no lineal va a ser resuelto. Es necesario especificar el tamaño de paso con el que se va a resolver.
- *Monitors*: se selecciona los *monitors* que van a ser empleados para la simulación. Indican qué es lo que se quiere grabar. Por último se indican los parámetros asociados a los monitores.
- *Simulation Length*: tiempo de simulación (ms).

Una vez establecidos los parámetros se puede lanzar la simulación. Como resultado, se va a obtener un dato del tipo serie temporal, al cual se le va a poder hacer uso de varios *visualizers*.

- *Time Series Visualizer* (SVG/d3): se muestra la señal de la actividad de cada región a lo largo del tiempo. En la parte inferior, la línea sólida muestra la media de todos los canales y la parte sombreada la desviación típica. **Fig. 5.6. (A)**
- *Animated Time Series Visualizer*: se muestra la actividad de cada una de las regiones. Se le llama animado debido a que existe una línea roja vertical que avanza a lo largo del gráfico. **Fig. 5.6. (B).**
- *Brain activity visualizer*: se muestra tanto la animación en 3D anterior a la izquierda, como las tres vistas de una región concreta a la derecha. **Fig. 5.6. (C).**
- *Brain dual activity visualizer* (3d and 2d): se presenta a la izquierda la representación en 3D de la actividad cerebral, mostrando en cada región un color que varía en el tiempo. A la vez en la derecha se muestra *Animated Time Series Visualizer*, en donde la línea roja que se va moviendo está sincronizada respecto a la evolución de la animación en 3D. **Fig. 5.6. (D).**

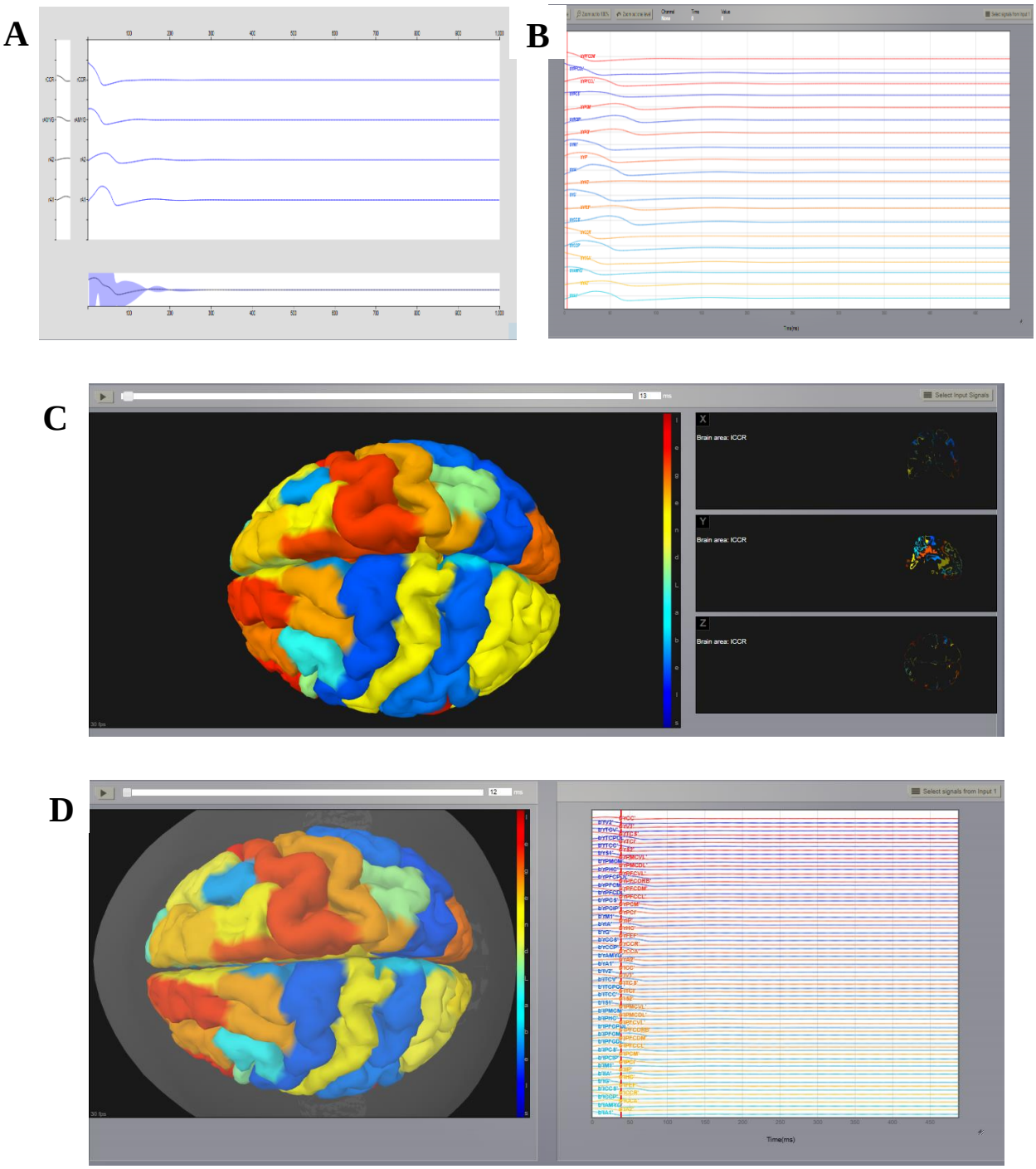


Figura 5.6. Visualizers de la simulación. **(A)** Time Series Visualizer. **(B)** Animated Time Series Visualizer. **(C)** Brain activity visualizer. **(D)** Brain dual activity visualizer.

Tabla 5.2. Configuración simulación basada en regiones

Long-range connectivity			Connectivity [76]-John Doe		
Conduction speed			3.0		
Coupling			Linear		
a	0.05		b	0	
Cortical Surface			*None*		
Spatiotemporal stimulus			*None*		
Local Dynamic model			*Generic 2D Oscillator*		
r	1	c	0	g	0
Iext	0	d	0.02	alfa	1
a	-2	e	3	beta	1
b	-10	f	1	gamma	1
Variables or quantities available to Monitors			V		
Integration scheme			Heun		
Integration-step size (ms)			0.5		
Monitors			*Temporally Sub-Sample*		
Sampling period (ms)			0,9765625		
Model variables to watch			V		
Simulation Length (ms)			1000		

b) Basadas en superficie

Las simulaciones basadas en superficie consisten en aportar al modelo una malla triangulada que representa una estructura anatómica realista del cerebro, donde cada nodo va a ser una población neuronal, definida por su correspondiente modelo matemático (Ver 4.3. b)). Para ello, se va a utilizar los mismos valores que se muestran en la **tabla 5.3,** pero esta vez, en el apartado *Cortical Surface* se va a elegir una superficie que esté integrada en el proyecto. Cuando completamos esta opción, aparecen más campos que deben ser completados para realizar la simulación:

- *Region mapping*: relaciona las regiones del conectoma con los nodos de la malla, de forma que se establece a cada región cortical un grupo de nodos concretos.
- *Local Connectivity*: permite indicar como van a interactuar los nodos vecinos. Aquí se puede establecer una conectividad local que haya sido definida en la sección de conectividad escala local (Ver 5.3. b)).
- *Local coupling strength*: factor que reescala los pesos de la conectividad local.

Tabla 5.3. Configuración de la simulación basada en superficie

Cortical Surface	Surface - Cortical Surface - John Doe
Region Mapping	RegionMapping - John Doe
Local Connectivity	LocalConnectivity - Mexican-hat-surface
Local coupling strength	1,0

Además, cuando se vaya a especificar el modelo de dinámica local, aparecerá una nueva opción *"Set up Surface model".* Esta sirve para especificar que algún parámetro de dicho modelo local pueda variar espacialmente. Para ello se puede especificar la ecuación del que define esta variación de cierto parámetro y especificar directamente sobre la imagen sobre qué nodo se quiere aplicar. Se pueden establecer distintas ecuaciones describiendo cada una la variación de un parámetro diferente. En un mismo nodo se pueden aplicar varias de estas variaciones. Como se muestra en la **Fig.5.7. (D),** se puede observar que a la izquierda de la pantalla se puede seleccionar sobre qué parámetro se quiere aplicar, y posteriormente definir la función para la que hay dos posibilidades: Gauss o sigmoide. Dependiendo de cuál se elija habrá que definir unos parámetros u otros. En la parte inferior se muestra una gráfica con la función creada. Cuando aplicamos la ecuación (círculo rojo en **Fig.5.7. (D)**), podemos indicar, seleccionando directamente en la imagen del cerebro, sobe qué nodo se quiere aplicar. Para añadirlo se pulsa sobre *Add focal point* (circulo verde en **Fig.5.7. (D)**). Por último para aplicar los cambios se pulsa en *Submit Surface Parameters.* (círculo azul en **Fig.5.7. (D).**

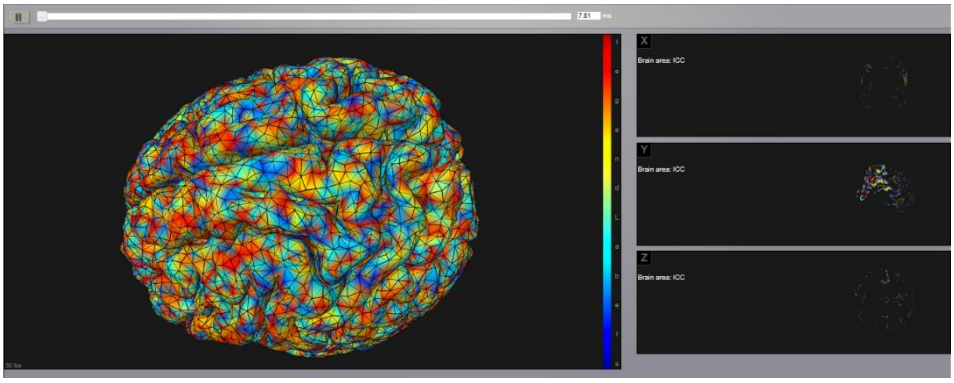
En cuanto a los *Visualizers*, son los mismo que en el basado en regiones, pero ahora muestran la señal de cada nodo. Por ejemplo, en la representación tridimensional **Fig.5.7.**, se puede observar como ahora la dinámica no es igual para toda la región, sino que cada nodo tiene la suya propia.

c) Incorporación de ruido

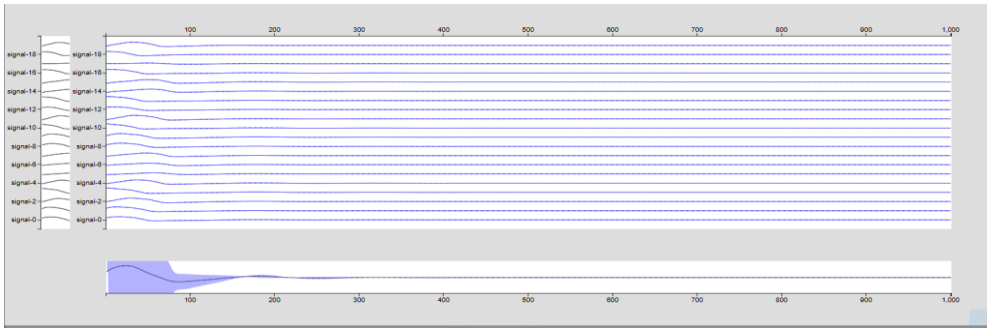
Uno de los elementos más importantes en las simulaciones cerebrales es el ruido, que como ya ha sido explicado anteriormente, puede jugar un papel importante, sobre todo en los estados de reposo. En TVB, la resolución de los modelos de población neuronal se hace mediante los esquemas de integración, métodos numéricos para la resolución de las ecuaciones diferenciales, los cuales pueden ser determinísticos o estocásticos. Estos últimos son los que incorporan fuentes de aleatoriedad, permitiendo la incorporación del ruido en el modelo cerebral.

Si un modelo estocástico es elegido, estarán disponibles nuevas opciones para configurar el ruido. Primero se debe elegir si este es aditivo o multiplicativo. En ambos casos se debe elegir la correlación del ruido con el tiempo, una semilla aleatoria y la dispersión del ruido. Esta dispersión puede ser asociada una región cerebral concreta. En las imágenes se muestra el resultado de la simulación aplicando ruido en las regiones prefrontales (**Fig. 5.8**). En el caso del ruido multiplicativo se debe definir una ecuación para modelar la variabilidad.

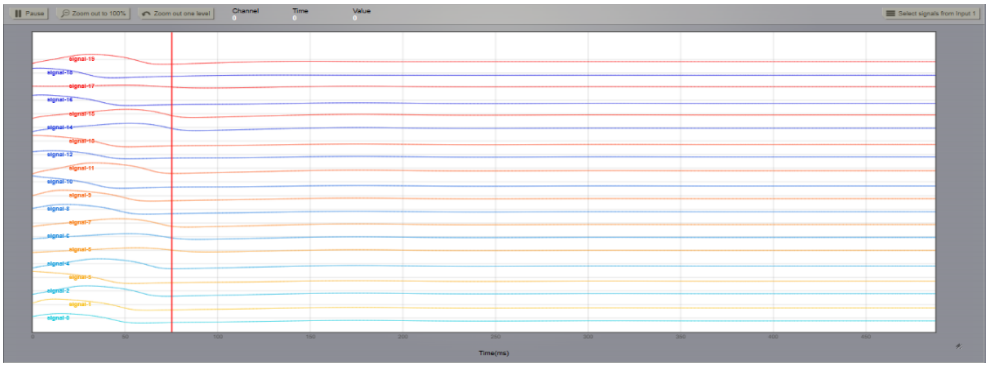
A



B



C



D

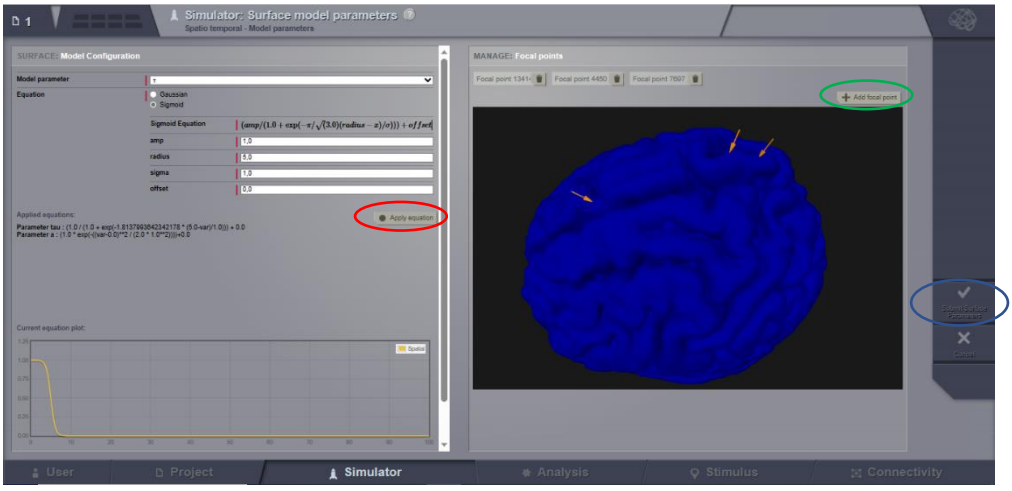


Figura 5.7. Visualizers simulación de superficie. **(A)** Brain activity visualizer. **(B)** Time Series Visualizer. **(C)** Animated Time Series Visualizer. **(D)** Panel Set up Surface model

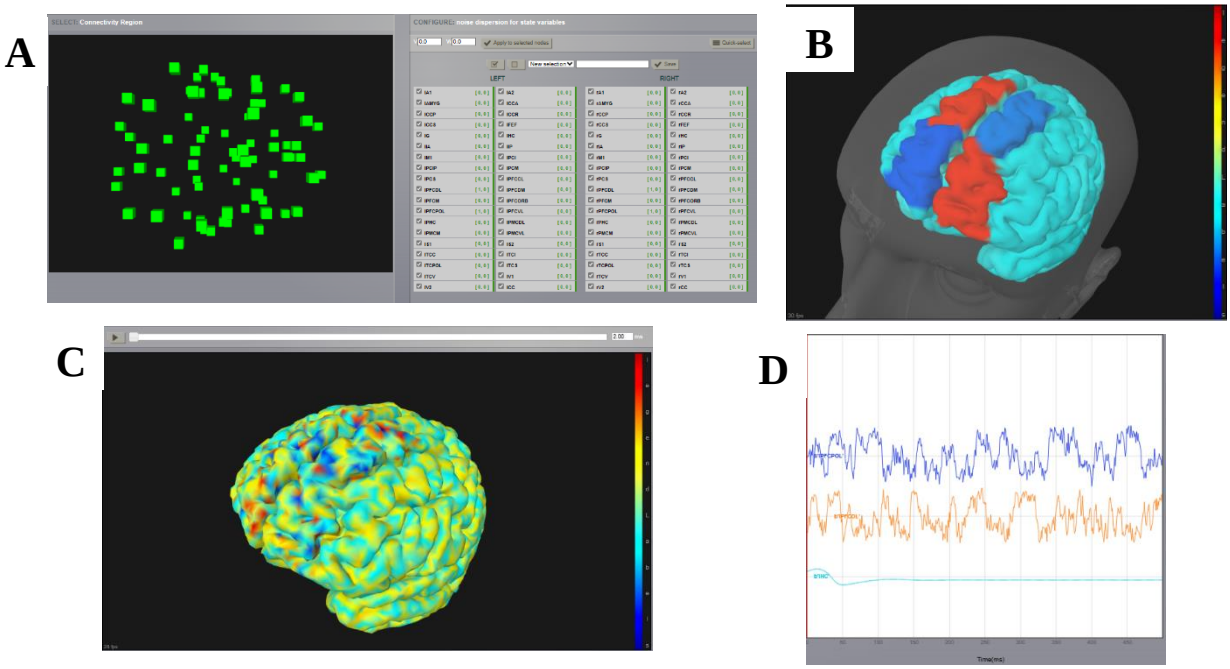


Figura 5.8. Simulación con ruido. **(A)** Configuración del ruido. **(B)** Simulación basada en regiones con aplicación de ruido. **(C)** Simulación basada en superficie con aplicación de ruido. **(D)** Señales azul y roja son nodos con ruido, y azul claro nodo sin ruido

5.5.Monitores de alto nivel

a) EEG – MEG – iEEG

El cálculo de las señales de EEG fue explicado en el apartado 4.3. c). Aquí se explica que el problema de la conducción de volumen se resuelve mediante un modelo de elementos de frontera en el que la cabeza queda dividida en tres superficies con propiedades distintas que representan la transición entre cuero cabelludo-aire, el cráneo-cuero cabelludo y el cráneo-cerebro. Por eso, lo ideal para obtener las señales de EEG de una simulación, es considerar un modelo de estas tres superficies. Estas se pueden incorporar como datos al proyecto en el que se está trabajando. En la subsección *Data-Structure*, en los datos

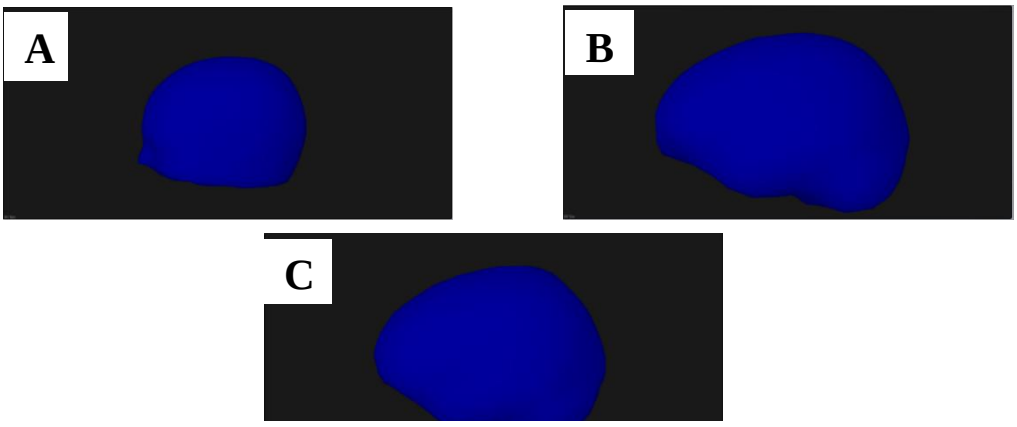


Figura 5.9. Superficies. **(A)** Superficie aire-piel. **(B)** Superficie cráneo-piel. **(C)** Superficie cráneo-cerebro

crudos se pueden observar que estas tres superficies están disponibles y, además pueden ser representadas mediante el "*surface visualizer*". **Fig. 5.9.**

Además, en estos datos crudos también se encuentran las matrices de proyección necesarias para realizar el cálculo de las señales de EEG, MEG e iEEG, junto con la disposición de los sensores de estas tres modalidades. Estos últimos pueden ser representados gracias al "*sensor visualizer*". **Fig. 5.10.**

El cálculo de estas señales en las simulaciones se hace a través de la selección de los correspondientes *monitors*. Una vez realizada la simulación se pueden visualizar los resultados, tanto en tres dimensiones sobre el cerebro, como las señales obtenidas de cada sensor (**Fig. 5.11**).

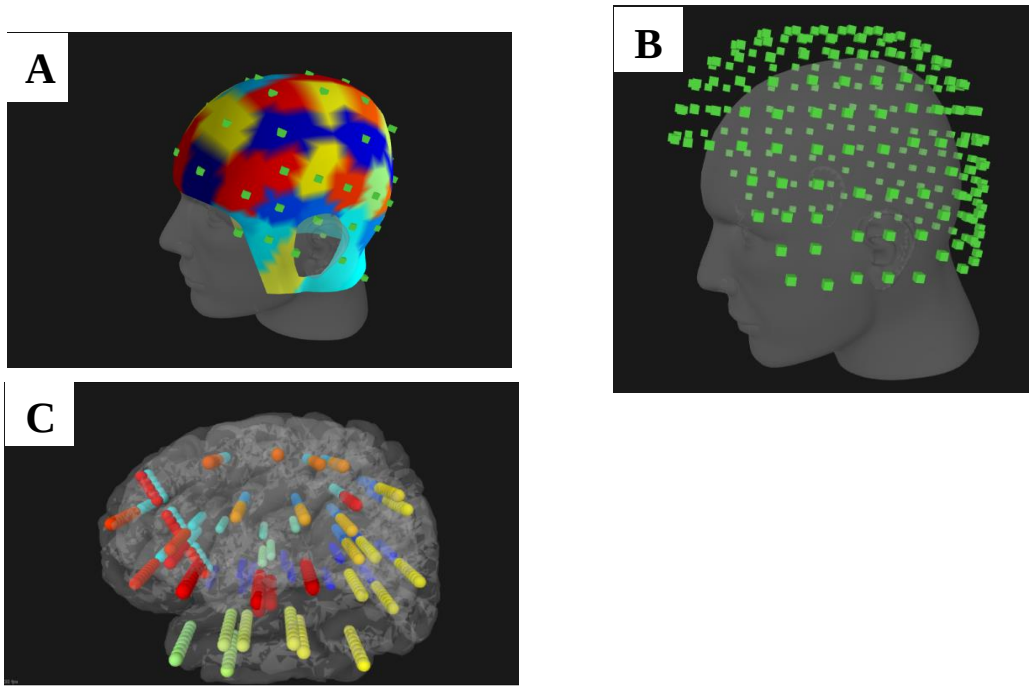


Figura 5.10. Sensor visualizer. **(A)** EEG. **(B)** MEG. **(C)** iEEG.

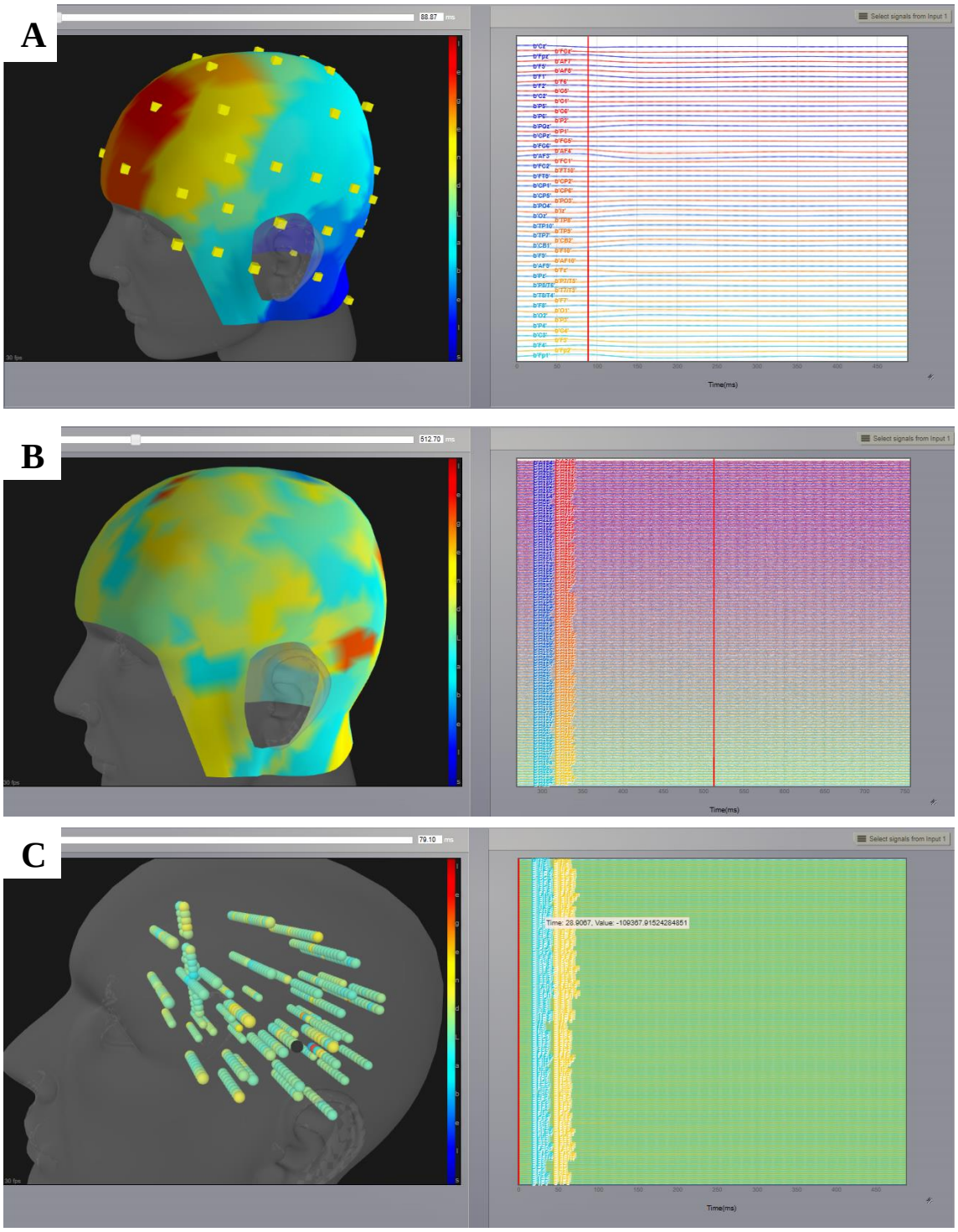


Figura 5.11. Resultados de la simulación. **(A)** EEG. **(B)** MEG. **(C)** iEEG.

b) BOLD – fMRI

La señal BOLD también puede ser calculada a partir de la actividad neuronal simulada en TVB. En el apartado 4.3. c) ya se explicó cómo se calculaba esta señal. A la hora de realizar la simulación, en el apartado *monitors* se debe especificar "BOLD". Si se realiza una simulación basada en superficie también se dispone del monitor "*BoldRegionROI*" que utiliza el mapeo de la región de la superficie para generar señales regionales que son el promedio espacial de todos los vértices de la región (**Fig. 5.12.**).

Se realizó una simulación de una hora de duración en la que se aplicaban estímulos cada 15 minutos de 5 minutos de duración, en distintas regiones del cerebro. En la parte frontal se aplicaban estímulos excitatorios y en algunas regiones occipitales estímulos inhibitorios. Las simulaciones que se generen deben ser relativamente grandes en el tiempo, debido a que la señal BOLD se caracteriza por tener una baja resolución temporal, en este caso tiene un periodo de muestre de 2 segundos.

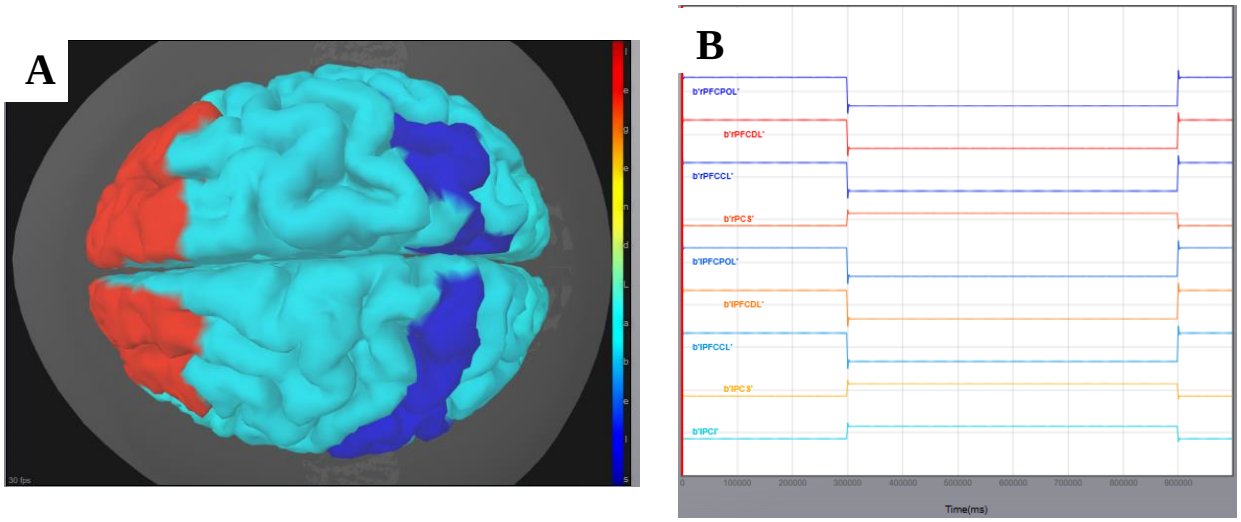


Figura 5.12. BOLD. **(A)** *Visualizer* señal BOLD **(B)** Señal BOLD.

5.6.El plano de fases

a) Sistemas no lineales

En el apartado 4.3. a), se explica que un nodo en TVB va a representar un población neuronal capaz de generar su propia dinámica interna mediante un modelo previamente definido. Estos modelos son, en realidad, sistemas no lineales. Los sistemas no lineales se caracterizan por una serie de ecuaciones diferenciales que describen su comportamiento. Estas ecuaciones involucran variables de estado, que son aquellas cuyos valores cambian con el tiempo y están sujetas a derivación. Las variables de estado representan las condiciones internas o características cambiantes del sistema. Además, los sistemas no lineales pueden contener parámetros, que son valores fijos que influyen en las propiedades del sistema, pero no varían en el tiempo. Estos parámetros actúan como constantes que determinan cómo se comporta el sistema en conjunto.

Una herramienta muy utilizada para el análisis de este tipo de sistemas no lineales, como los modelos de poblaciones neuronales, es el espacio de fases. El espacio de fases es una

representación gráfica del dicho sistema no lineal que permite visualizar y comprender su comportamiento dinámico. Cada uno de los ejes corresponde a una variable de estado del sistema. El espacio de fases muestra cómo evolucionan estas variables de estado a lo largo del tiempo y cómo interactúan entre sí.

En el espacio de fases, una trayectoria representa la evolución del sistema a partir de una condición inicial específica. Esta condición inicial se define por un conjunto particular de valores para las variables de estado. A medida que el sistema evoluciona, la trayectoria describe cómo las variables de estado cambian y se relacionan entre sí, formando un recorrido en el espacio de fases. Cada punto a lo largo de la trayectoria representa un estado del sistema en un momento dado.

Esta representación va a permitir predecir el comportamiento del sistema a largo plazo mediante la identificación de ciertos patrones. Un tipo de estos patrones son los puntos fijos. Los puntos fijos son aquellos en los que las derivadas de todas las variables de estado son cero, es decir, no cambian con el tiempo. Se pueden distinguir dos tipos: los estables y los inestables. Los estables son aquellos hacia los cuales el sistema tiende con el tiempo y pueden estar asociados hacia la estabilización del sistema hacia cierto punto. Los inestables son aquellos de los que el sistema se aleja en su evolución, no convergiendo en ellos. Un segundo tipo de patrón son los ciclos límite, trayectorias cerradas en el espacio de fases que el sistema recorre repetidamente en un patrón periódico. Se asocian a oscilaciones estables o patrones repetitivos.

Otro concepto importante es el de nulclinas (*nullclines*). Estas son líneas en el espacio de fases, donde la derivada de cierta variable de estado es cero, es decir el valor de dicha variable no cambia. Es importante identificar la intersección de varias nulclinas, puesto que puede indicar la presencia de puntos fijos, pues en ese punto las derivadas de esas variables de estado es cero.

Un sistema no lineal se define al asignar los valores adecuados a sus parámetros. Durante el estudio de estos sistemas, es posible encontrarse con situaciones en las que se dé una bifurcación. Una bifurcación representa un cambio drástico que puede ocurrir en un sistema ante pequeñas variaciones de parámetros, lo que puede resultar en la aparición de nuevos patrones como puntos fijos o ciclos límite.

b) **El plano de fases en TVB**

TVB permite realizar un estudio de los modelos de poblaciones neuronales mediante la representación de su plano de fases. De forma interactiva, es posible evaluar cómo afectan diferentes valores de los parámetros que caracterizan dichos modelos, al plano de fases. En el plano de fases va a estar dibujado por defecto las nulclinas de cada variable de estado. Dependiendo de que valores tengan estos parámetros, estas líneas serán de una forma u otra. Además, es posible dibujar una trayectoria en el plano a partir de cierta condición inicial, que se establecería simplemente pinchando en el diagrama.

A continuación, se va a mostrar un ejemplo sencillo de como interactuar con el plano de fases en TVB. Para ello, se va a ser uso del modelo local dinámico ***"Suphopf"***. Para

empezar, en la **Fig. 5.13.** se puede observar que la pantalla queda dividida en tres secciones:

A. Configuración de parámetros del modelo: permite elegir el modelo local, da una explicación del mismo con sus correspondientes ecuaciones diferenciales y se establece el valor de los parámetros del modelo.

B. Configuración del visor del plano de fase: se elige el método numérico con el que se quiere evaluar, las variables de estado que se quieren representar, el rango de valores de estas que se quiere mostrar en los ejes y el número de pasos de integración para el cálculo de las trayectorias.

C. Representación del plano de fases: aparecen representadas dos líneas que serían las nuclinas, o regiones del plano en las que la derivada de la variable de estado es cero. Las flechas azules indican la dirección que seguiría una trayectoria dependiendo de la condición inicial.

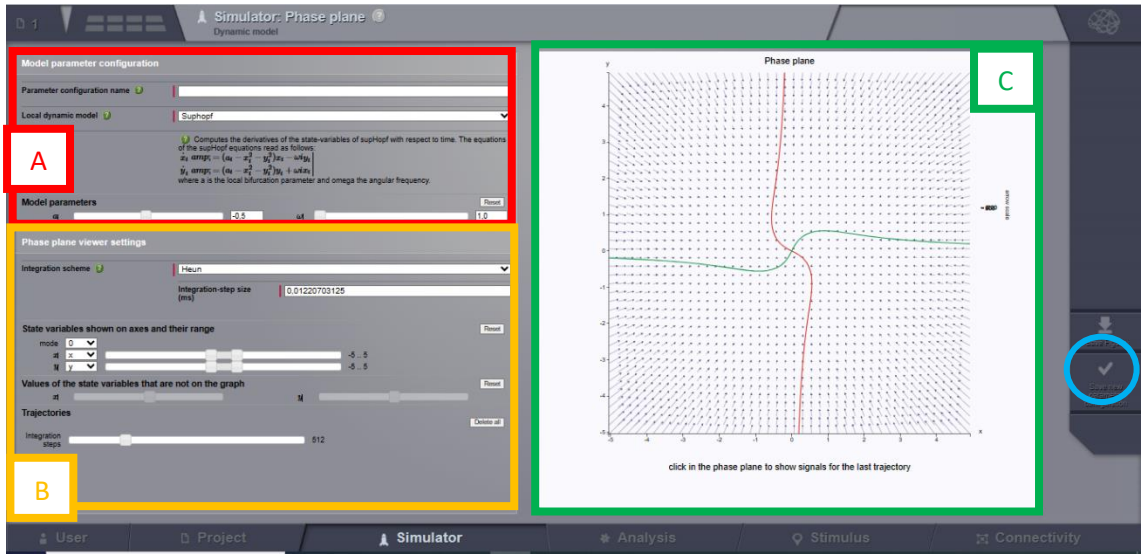


Figura 5.13. Visualización plano de fases. **(A)** Configuración de parámetros del modelo. **(B)** Configuración del visor del plano de fase **. (C)** Representación del plano de fases.

Si realizamos un análisis del sistema (**Fig. 5.14.**), se puede observar que entorno al valor crítico de 0 para el parámetro *a*, tiene lugar una bifurcación. Por debajo de 0, el punto en el que se cruzan ambas nulclinas es un atractor, puesto que las trayectorias dibujadas tienden hacia ese punto. En la representación temporal, se puede ver cómo tienden hacia el equilibrio. Sin embargo, si *a* es mayor que 0 podemos observar que se forma un ciclo límite, en el que la intersección de las nulclinas es su centro, asociándose con una oscilación en el tiempo.

Después de haber escogido los parámetros que hayamos creído más adecuados, es posible guardar esta configuración para posteriormente aplicarla a ciertas regiones de nuestro modelo. De esta forma distintas regiones serán modeladas de diferente forma. Para ello, después de terminar con la configuración de nuestro modelo se guardan los parámetros pulsando el botón resaltado en azul en **Fig. 5.13.** Ya de vuelta en *'Simulation Cockpit'* cuando se debe elegir el tipo de modelo local se debe pulsar sobre *'Set up Region Model'* (**Fig. 5.15.)**. Desde aquí se puede elegir a qué región se quiere aplicar cierta configuración de parámetros. En **Fig. 5.14** se muestra una aplicación del modelo ***"Suphopf",*** asignando una configuración con un valor de *a=-5,21* a las regiones del hemisferio izquierdo y otra con un valor de *a=2,33* a las regiones del hemisferio derecho. Este modelo cerebral tiene fines ilustrativos, para una aplicación clínica real de este proceso ver el apartado 5.6. a).

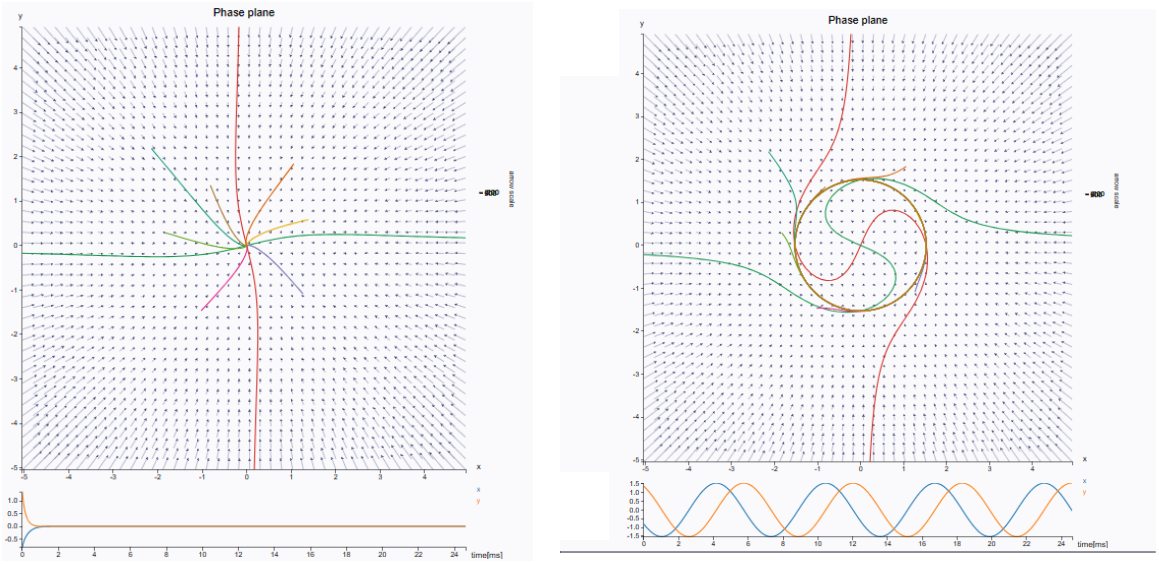


Figura 5.15. Suphopf. **(A)** Plano de fases con a=-5,21. **(B)** Evolución temporal del sistema a=-5,2. **(C)** Plano de fases con a=2,33. **(D)** Evolución temporal del sistema a=2,33.

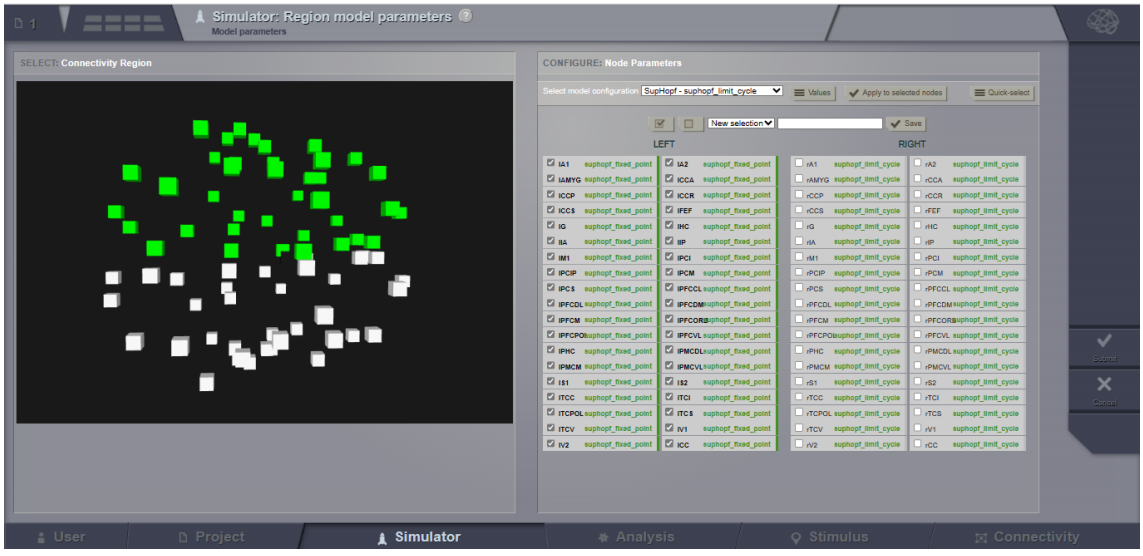


Figura 5.14. Pantalla de configuración *"Set up Region Model"*

c) Modelos más comunes

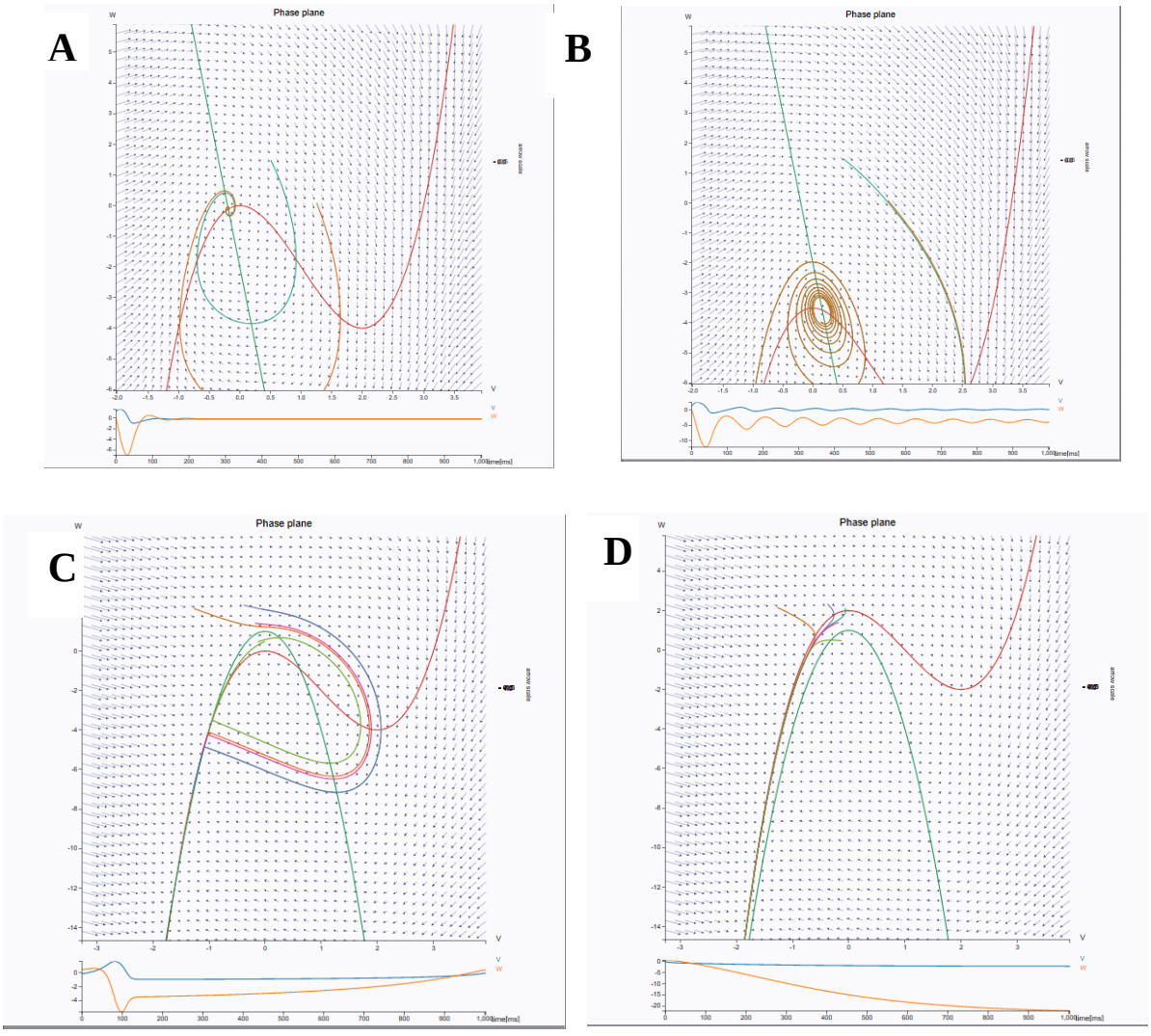
TVB ofrece una amplia gama de modelos a elegir a la hora de realizar las simulaciones. A continuación se mostrarán algunos de los más comunes y se estudiará cuál es su comportamiento en el plano de fases.

Generic 2D oscillator

Este modelo describe el comportamiento de una masa neuronal por medio de dos variables de estado: V, que representa el potencial de membrana de la neurona; y W que es la variable de recuperación. Además cuenta con el parámetro I que representa la introducción de diferentes corrientes, tanto del vecindario local como de fuentes lejanas. Este modelo es capaz de representar una gran variedad de fenómenos observados en el comportamiento de las poblaciones neuronales.

En [18] se pueden encontrar tres configuraciones de parámetros posibles: excitable, biestable y bifurcación de silla de nodo sobre el círculo invariante.

En la configuración excitable, el parámetro I presenta una bifurcación en torno a 3, pasando de un punto fijo cuando es menor que tres a un ciclo límite. En la configuración biestable, I=0 coexisten un punto fijo y un ciclo límite, I=-2 hay solo un punto fijo. En cuanto a la bifurcación silla de nodo, a partir de I=0,2 surge un ciclo límite **(Fig. 5.16)**.



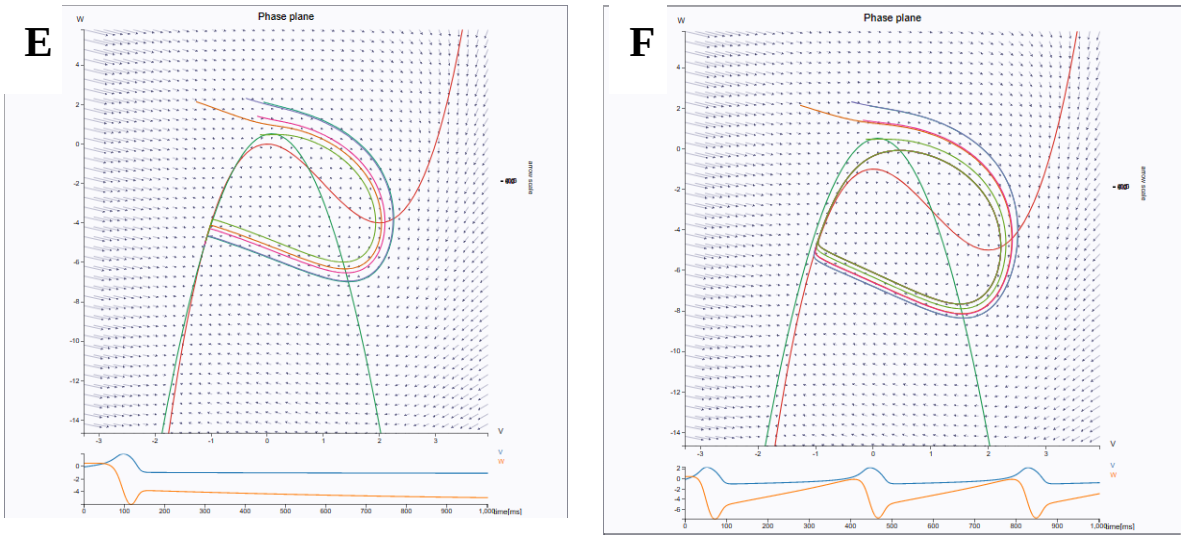


Figura 5.16. *Generic 2D Oscillator*. Configuración excitable con **(A)** I= 0 **(B)** I=4. Configuración biestable **(C)** I=0 **(D)** I=-2. Bifurcación de silla de nodo sobre el círculo invariante con **(E)** I=0 **(F)** I=3.

Wilson-Cowan

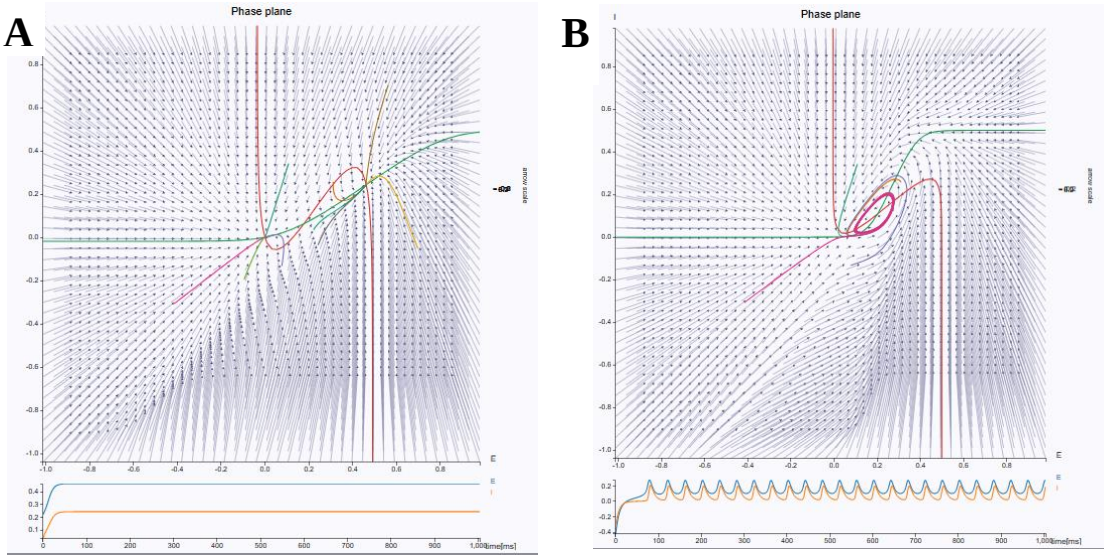


Figura 5.17. Modelo Wilson-Cowen. Modelo Wilson-Cowen. **(A)** Tres puntos fijos. **(B)** Ciclo límite.

Otro modelo que puede ser evaluado es el de ***'Wilson-Cowan'***, en el que se representa la dinámica entre poblaciones neuronales excitatorias e inhibitorias. Este modelo contiene dos variables de estado *E(t)* e *I(t)*, que representan la proporción de células excitatorias e inhibitorias que disparan por unidad de tiempo, respectivamente. En [18] se puede obtener los valores de los parámetros que definen los comportamientos posibles del modelo. En

Fig.5.17 se representan dos patrones distintos, el primero con dos puntos fijos estables y uno inestable y el segundo con un ciclo límite.

Wong, Wang & Deco

El modelo se describe mediante unas ecuaciones diferenciales acopladas que representan la tasa de disparo de población y la variable de compuerta sináptica promedio en una región cortical local, que se pueden encontrar en [18]. Presentan términos de acoplamiento a larga distancia y retardo temporal. Este modelo presenta tres regímenes de actividad que vienen determinados por el valor de los parámetros *w* (conexiones recurrentes locales) e I_0 (entrada externa). El primer régimen incluye únicamente un punto fijo estable que corresponde a una baja actividad. El segundo régimen consta de dos puntos fijos estables, representando uno actividad baja y otro actividad alta, junto con un punto fijo inestable. Por último, el tercer régimen contiene exclusivamente un punto fijo estable que refleja una alta actividad **(Fig. 5.18)**.

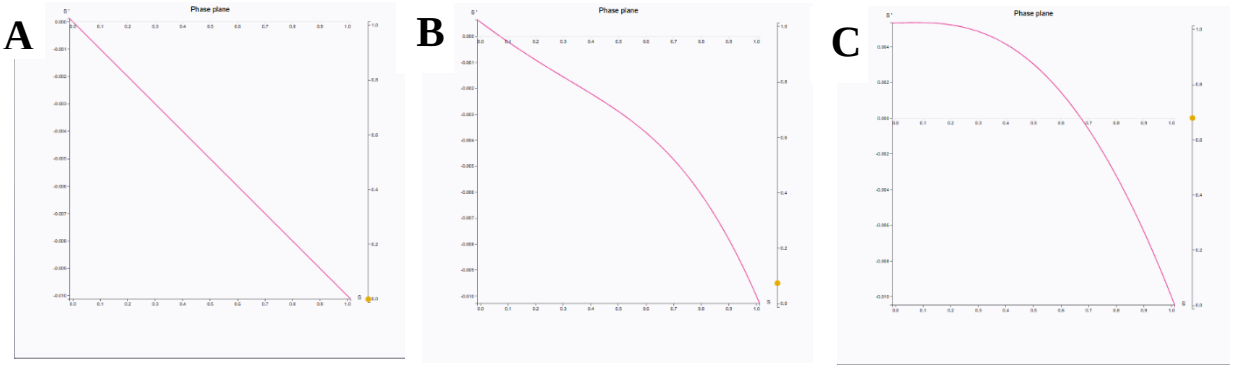


Figura 5.18. Modelo Wong, Wang & Deco para w=0,6. **(A)** Primer régimen $I_{=0}$. **(B)** Segundo régimen para I_0=0,33. **(C)** Tercer régimen para I_0= 0,42.

5.7.Exploración del espacio de parámetros

Como se vio en el apartado 4.3. d), para determinar qué valores de los parámetros del modelo simulado se van a acercar más a la realidad se debe realizar una etapa de ajuste de parámetros. Para ello, se realizaba varias simulaciones, en las que cada parámetro de interés iba a variar un cierto paso en cada simulación. Por último, se realizaba una correlación entre la sFC y la eFC y la matriz simulada que obtenía mejor ajuste implicaba que ese parámetro era el más adecuado.

TVB permite realizar esta exploración del espacio de parámetros (PSE, *Parameter Space Exploration*) de forma directa para realizar las simulaciones. Para realizar la PSE, cuando se termine de realizar la configuración de la simulación de la misma forma que se hizo en 5.3. a), se pulsa al botón PSE. Entonces aparecerán nuevos campos para rellenar. Primero habrá que elegir los, como máximo, dos parámetros que quieren ser estudiados. Por último, para cada uno hay que especificar el intervalo de estudio y el tamaño de paso. Se realizará una simulación por cada posible combinación de parámetros.

Para comparar los resultados obtenidos en la simulación se van a calcular métricas para cada una de las simulaciones (metaestabilidad, varianza global, sincronía, y varianza de nodo). Estas métricas van a poder ser visualizadas de par en par mediante dos *visualizers*.

En el primero *Discrete Parameter Space Exploration* (**Fig. 5.19.**), cada simulación está representada por un punto de cierto tamaño y color. El tamaño de estos representa el valor de una métrica y la escala de colores el valor de la otra métrica. Estos se disponen espacialmente, de forma que cada uno de los ejes representa uno de los parámetros que han ido variando con cada simulación.

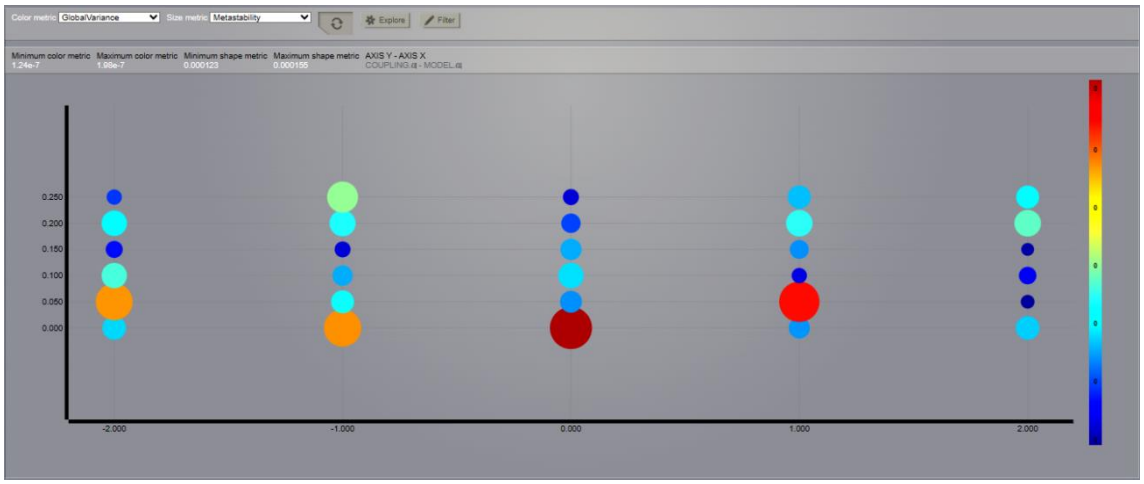


Figura 5.19. *Discrete Parameter Space Exploration*

En el segundo *visualizer*, *Isocline Parameter Space Exploration* (**Fig. 5.20)**, se muestra cómo varía de forma continua el valor de una métrica seleccionada para los distintos valores de los dos parámetros seleccionados, siendo estos representados por los ejes X e Y.

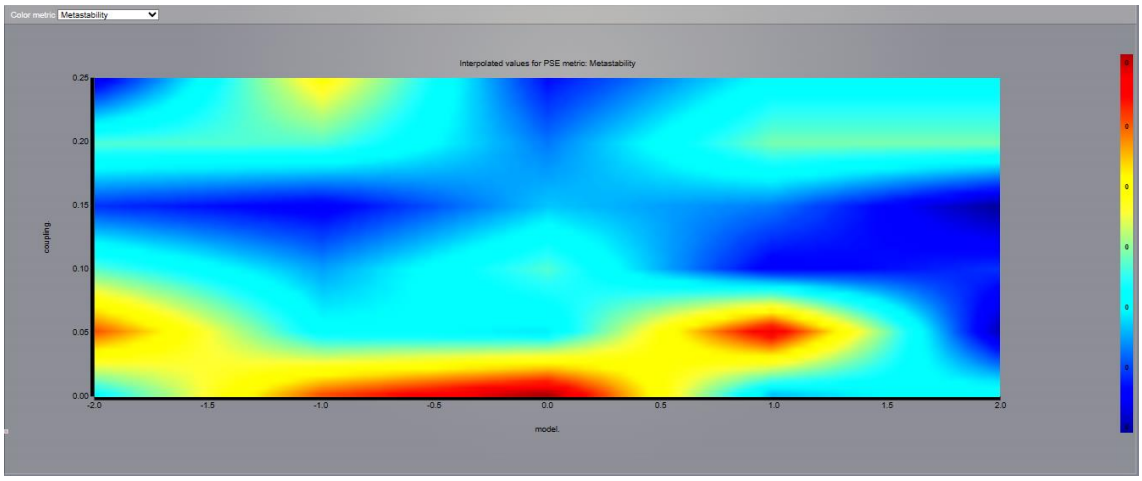


Figura 5.20. *Isocline Parameter Space Exploration*

5.8.Simulaciones con estímulos

TVB permite introducir estímulos en la simulación del cerebro. Se distinguen dos tipos de estímulos, los regionales y los de superficie. Para diseñar estos se debe acceder a la sección "*Stimulus*".

a) Regiones

Los estímulos a nivel regional consisten en aplicar un cierto estímulo a una o varias regiones por completo. Para diseñar el estímulo deseado se debe acceder a la correspondiente sección dentro del panel "*Stimulus*". Una vez allí aparecerá la pantalla como en **Fig. 5.21 (A)**. Aquí se puede nombrar el estímulo que se va a crear y asignarle un conectoma con las regiones del cerebro. Por último, se tiene que definir la ecuación temporal que va a definir cómo evoluciona el estímulo y cada cuanto debe aplicarse a lo largo de toda la simulación. Para ello, se debe elegir el tipo de función de las distintas opciones ofertadas y completar los parámetros con los valores adecuados. A la derecha de la pantalla se puede observar la aplicación de los estímulos a lo largo de cierto tiempo. La configuración utilizada para el diseño se puede ver en la **tabla 5.4.**

Para elegir sobre qué regiones este va a ser aplicado, se debe pulsar en *Set Regions Scaling* (**Fig. 5.21 (B)**). Ahí a cada región se le puede aplicar también un factor de escala o fuerza del estímulo para que este sea más o menos débil. Para guardar se debe pulsar en Save *New Stimulus on Region*. Una vez de vuelta a la configuración del estimulador (se ha utilizado **tabla 5.2.)**, en el apartado *Spatiotemporal stimulus* se selecciona el estímulo creado.

Los resultados muestran como al utilizar un modelo de dinámica local *Generic 2D Oscillator*, con *a=-2,* el sistema va a llegar al equilibrio. A partir del 1000 ms se empieza a aplicar los estímulos y se observa cómo se aplica a la región en su totalidad. Para la región rPFCPOL , el factor de escala era positivo, por eso se produce una hiperactividad en dicha región, mientras que en lTCC, el factor de escala es negativo, por lo que se produce una hipoactividad (**Fig. 5.21(C)**).

Tabla 5.4. Configuración de estímulo a nivel regional

Display name		Stimulus_region_1	
Connectivity		Connectivity [76]-John Doe	
Temporal Equation		Pulsetrain	
T		100	
Tau		10	
Amp		1	
onset		1000	
Región 1	rPFCPOL	***Scale factor***	20
Región 2	lTCC	***Scale factor***	-20

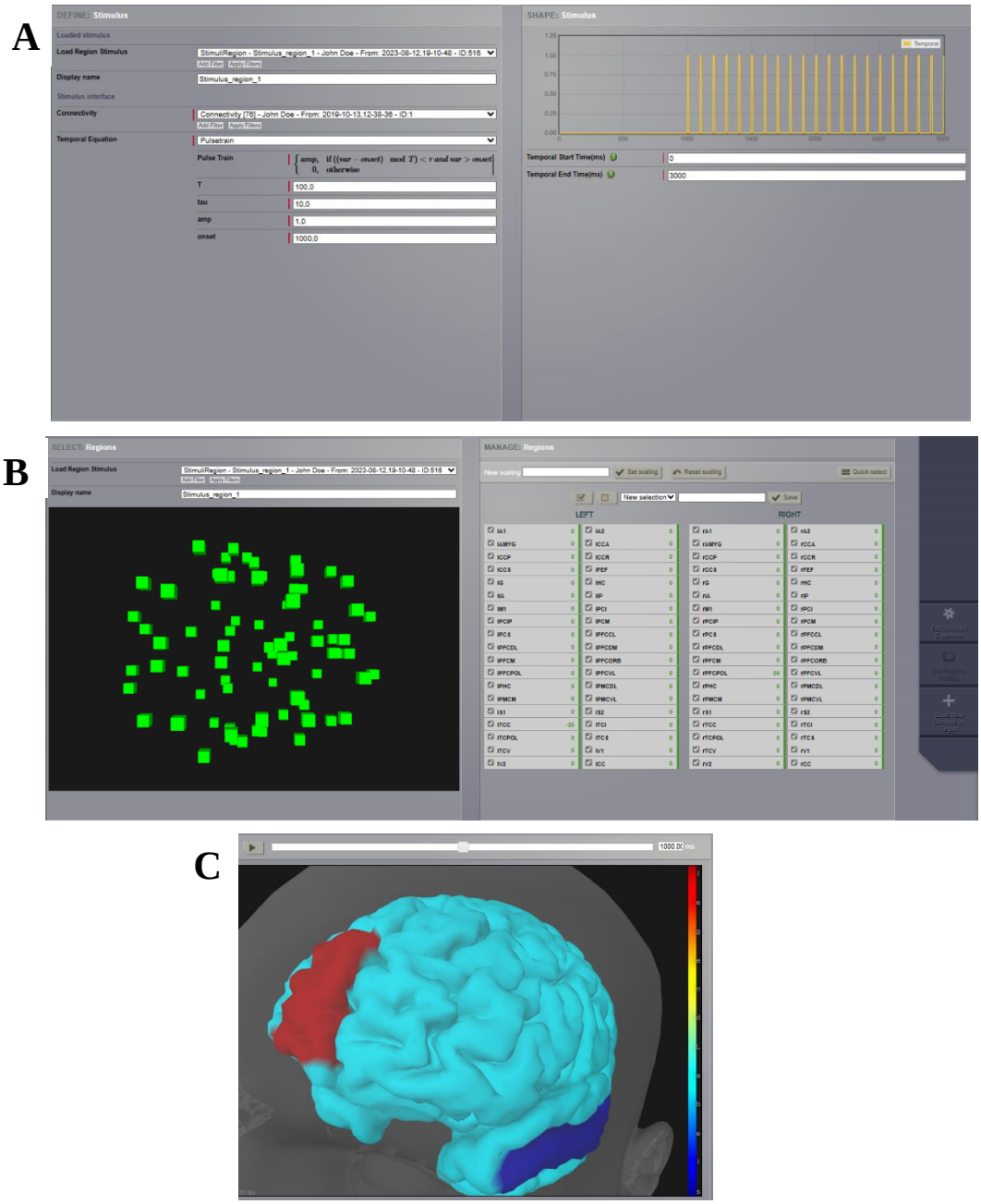


Figura 5.21. Simulaciones con estímulos. **(A)** Diseño estímulo sobre región. **(B)** Selección de la región sobre la que va a ser aplicado el estímulo. **(C)** Simulación en t=1001 ms, en rojo rPFCPOL con factor de escala positivo y en azul lTCC con factor de escala negativo.

b) Superficie

Los estímulos de superficie consisten en aplicar un cierto estímulo a uno o varios nodos que están representados como vértices en la malla triangular, durante una simulación basada en superficie. Para su diseño se debe acceder, dentro de la sección "*Stimulus*" a "Surface *Stimulus*". De la misma forma que a nivel regional, se debe indicar la ecuación temporal, pero esta vez también se deberá completar la ecuación espacial que describe cómo se va a expandir el estímulo en el espacio. Una vez definidas estas ecuaciones, se

pulsa en "*Edit Focal Points and View"*, y se mostrará una nueva pantalla en la que se podrá seleccionar directamente sobre la superficie cerebral en qué nodo se quiere aplicar la simulación. Además se puede observar una previsualización de cómo evolucionaría el estímulo en el tiempo y en el espacio. Para guardar el estímulo se pulsa "*Save new stimulus on Surface*".

Tabla 5.5. Configuración de estímulo a nivel de superficie

Display name	Stimulus_surface
Surface	Surface - Cortical Surface - John Doe
Spatial Equation	Sigmoid
Amp	1,0
Radius	10,0
Sigma	1,0
Offset	0,0
Temporal Equation	Pulsetrain
T	100
Tau	10
Amp	1
onset	1000

Para su aplicación se configura una simulación basada en superficie de la misma forma que se hizo en 5.3. b)., pero esta vez en el apartado *Spatiotemporal stimulus* se selecciona el que se haya guardado. De nuevo, en la simulación definida en la tabla 5.5, el sistema llega a un punto de equilibrio. A los 1000 ms va a introducirse el primer estímulo en los nodos seleccionados. Este estímulo se propagará siguiendo la ecuación espacial descrita anteriormente.

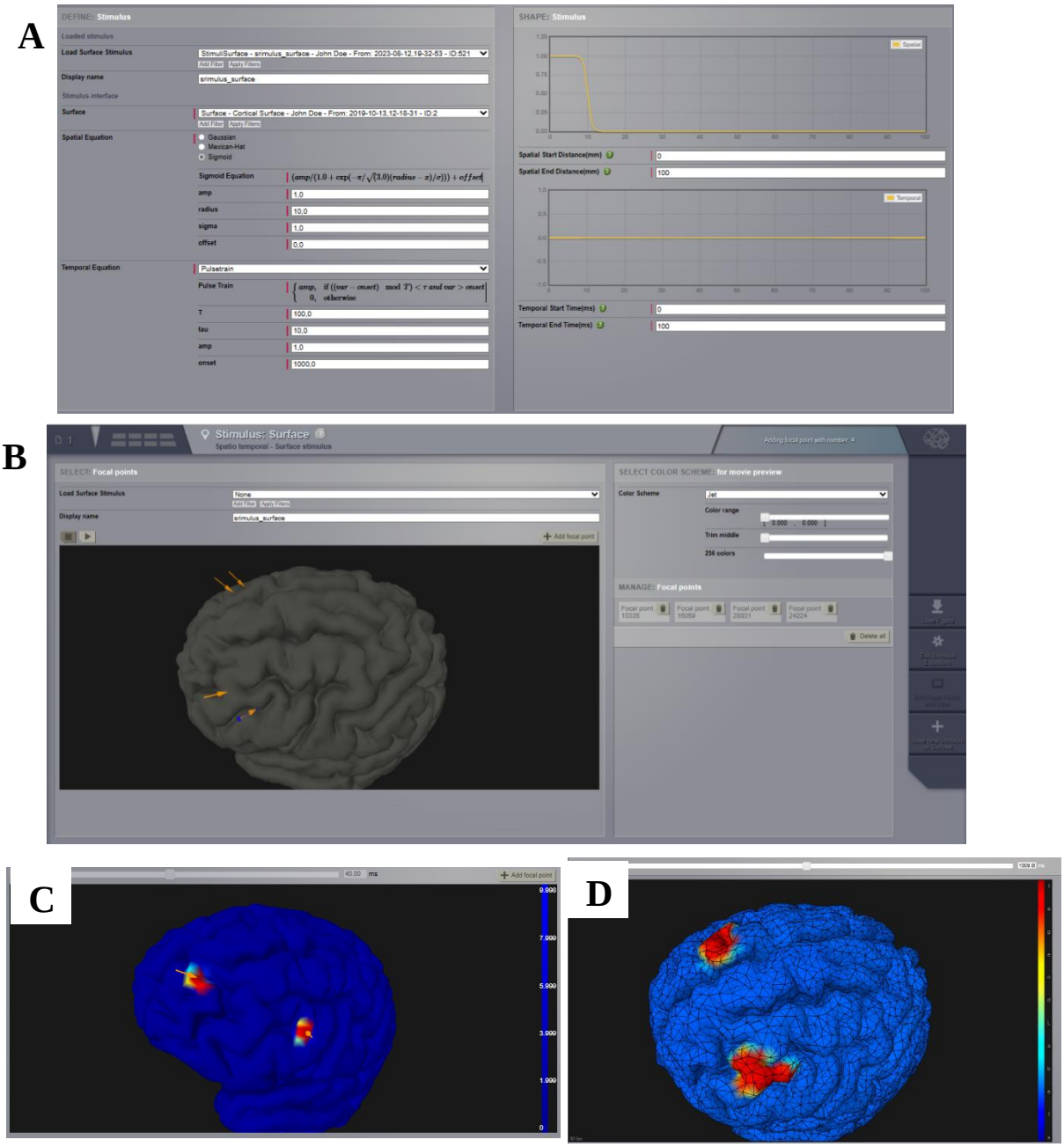


Figura 5.22. Simulaciones con estímulos en superficie. **(A)** Diseño estímulo sobre superficie. **(B)** Selección del nodo sobre el que va a ser aplicado el estímulo. **(C)** Previsualización del estímulo. **(D)** Resultados de la simulación, en las zonas que se aplica el estímulo se produce una hiperactividad.

5.9.Analyzers

TVB ofrece una serie de herramientas que permiten realizar un análisis de los resultados arrojados por las simulaciones. Aunque este no es su objetivo y se recomienda el uso de otros programas, pueden resultar útiles para realizar una primera aproximación de los resultados.

a) Transformada wavelet continua

La transformada wavelet continua (CWT) es utilizada para realizar un análisis tiempo-frecuencia. Este tipo de transformadas utilizan wavelets madre que van a ser escaladas y trasladadas a lo largo de la señal para ver cómo se adecuan a la señal, permitiendo obtener información simultanea sobre el tiempo y la frecuencia.

En la imagen se puede observar el espectrograma de la simulación basada en superficie a la que se le aplico un estímulo pulsátil. Se puede observar que al principio existen varias componentes a distintas frecuencias que desaparecen al llegar al punto de equilibrio. El tipo de estímulo que se aplica a partir de los 1000 ms, es un estímulo pulsátil que se repite cada 100 ms, es decir tiene un frecuencia de 0.01 kHz. Esta componente frecuencial se ve reflejada en el espectrograma a partir de los 1000 ms y se extiende durante el resto de la simulación.

Para el cálculo de CWT se debe indicar el tipo wavelet madre, el periodo de muestreo, la normalización (como se ajusta la amplitud de la wavelet madre a diferentes escalas), el ratio Q (relación entre frecuencia central y ancho de banda) y el rango de frecuencias.

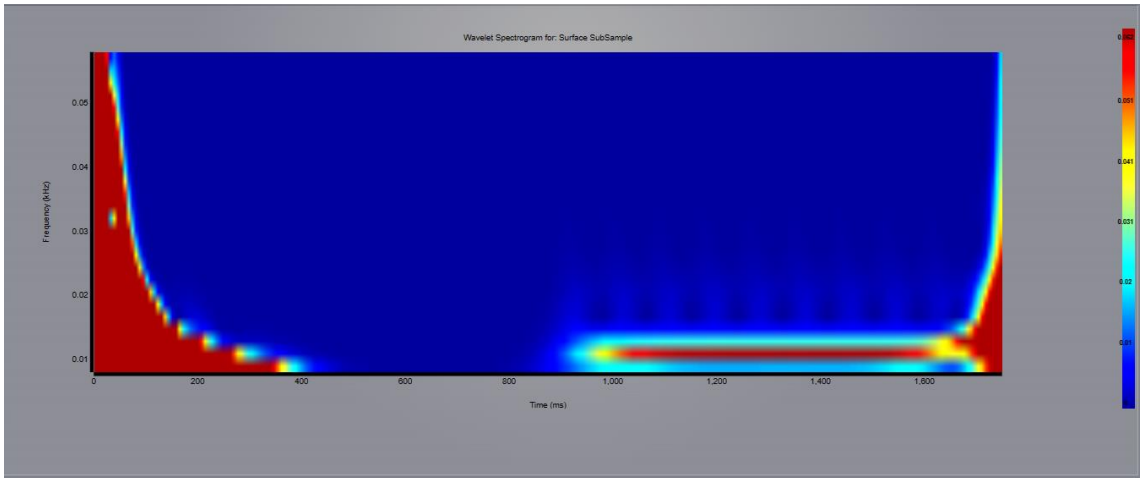


Figura 5.23. Transformada wavelet continua (CWT)

b) Análisis espectral de Fourier

Cálculo de la transformada discreta de Fourier, con el que se obtiene una representación de la señal en el dominio de las frecuencia. En la imagen se representa la transformada de Fourier de la simulación basada en superficie con aplicación de estímulos, donde se puede observar que la componente a baja frecuencias que representa los estímulos pulsátiles. Para su configuración se debe elegir el tamaño de la ventana y la función de esta.

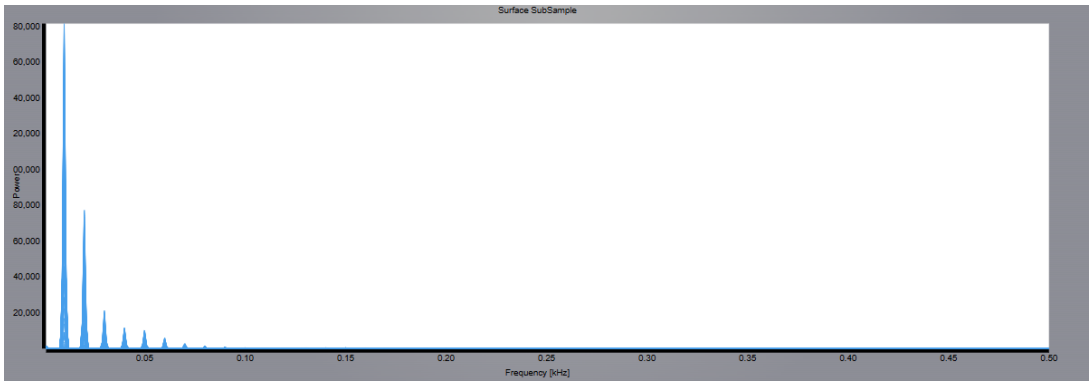


Figura 5.24. Transformada discreta de Fourier

c) Coeficientes de correlación de Pearson

El cálculo de los coeficientes de la correlación de Pearson entre las distintas señales generadas por cada nodo, sirve para obtener la matriz funcional simulada. En esta cada fila/columna representa un nodo y cada celda representa la correlación entre cada uno de estos.

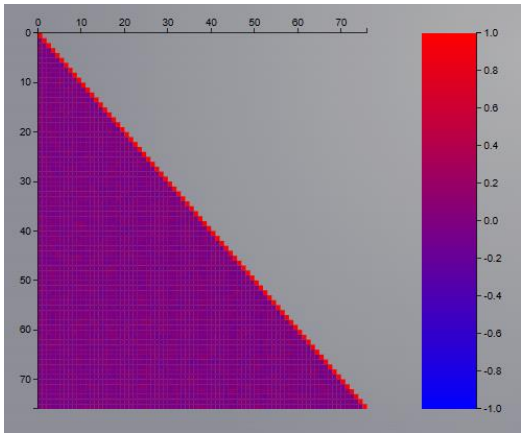


Figura 5.25. Coeficientes de correlación de Pearson

d) Matriz FCD

La conectividad funcional dinámica estudia como cambia la conectividad funcional a lo largo del tiempo. Se calcula una matriz de conectividad funcional por cada época utilizando la correlación de Pearson. Cada elemento de la matriz FCD es calculado como la correlación de Pearson entre dos matrices funcionales. Por ejemplo, el elemento ij de la matriz FCD es la correlación entre FC cuando t=i y FC cuando t=j. para su cálculo se debe determinar la longitud de la ventana, que será el tiempo de la simulación por el que va a calcular cada una de las FC y el tiempo entre el que se va a aplicar cada ventana con el que se va a calcular el solapamiento.

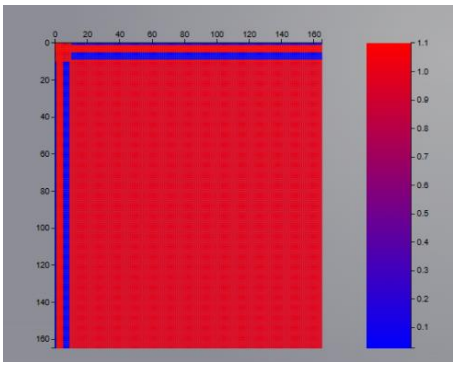


Figura 5.26. Matriz FCD.

e) Análisis de componentes independientes

Se puede obtener también un análisis de componentes independientes, el cual sirve para separar una señal multivariada en subcomponentes aditivos maximizando la independencia estadística mutua de las señales fuente.

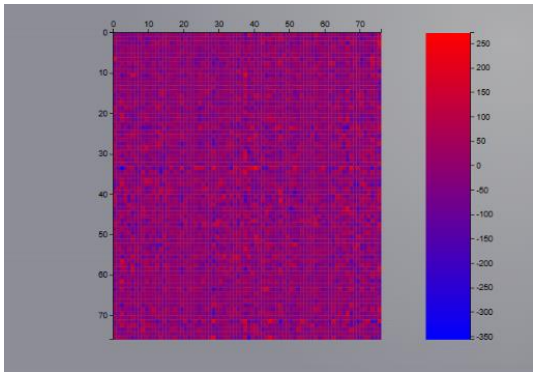


Figura 5.27. Análisis de componentes independientes

f) Análisis de componentes principales

El análisis de componentes principales es un método computacional para el análisis de datos multivariados que utiliza una transformación ortogonal para convertir un conjunto de variables (posiblemente correlacionadas) en un conjunto de variables linealmente no correlacionadas llamadas componentes principales. En el visor, a la izquierda se muestra el porcentaje de cada componente y a la derecha se muestra las diez primeras componentes contra cada nodo.

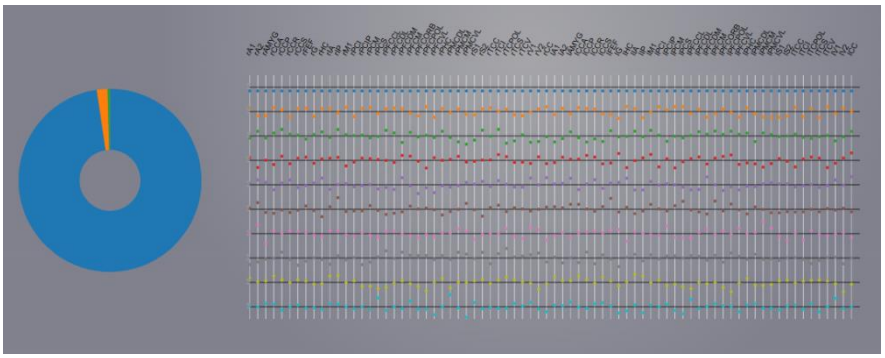


Figura 5.28. Análisis de componentes principales

g) Correlación cruzada de nodos

La correlación cruzada es una medida que sirve para cuantificar el grado de dependencia lineal entre dos series temporales. En este caso se calcula el coeficiente de correlación para todos los pares de nodos posibles. Se puede visualizar en una matriz de correlación.

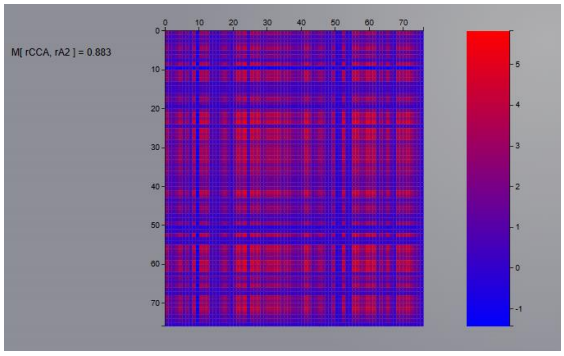


Figura 5.29. Correlación cruzada de nodos.

h) Covarianza temporal de nodos

La covarianza temporal mide cuánto cambian dos series de tiempo juntas. Se calcula los coeficientes entre todos los pares de nodos posibles. También puede ser visualizado en una matriz.

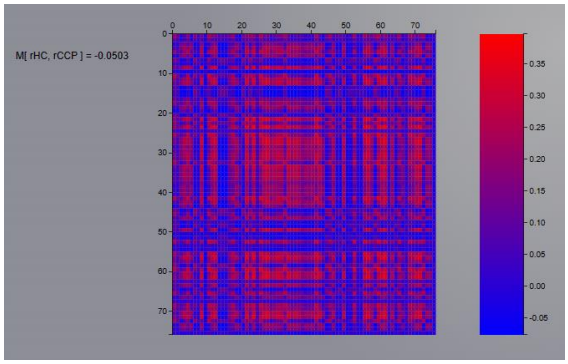


Figura 5.30. Covarianza temporal de nodos

i) Coherencia cruzada de nodos

La coherencia cruzada sirve para estimar como se relacionan dos señales temporales en el dominio espectral. Indica el grado en que la amplitud y la fase entre dos señales se relacionan entre sí en función de la frecuencia.

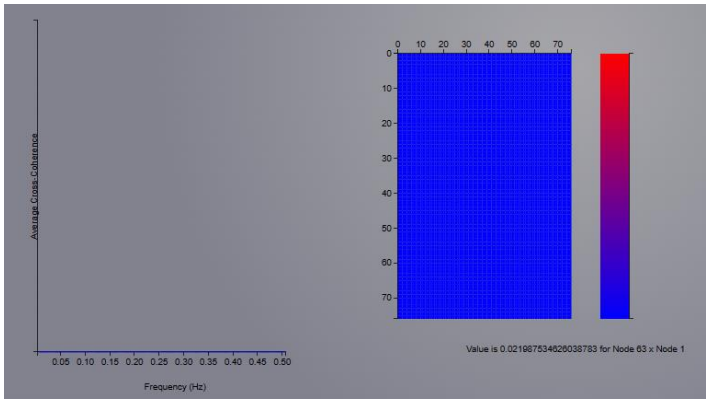


Figura 5.31. Coherencia cruzada de nodos

j) Brain Connectivity Toolbox

La *toolbox brain connectivity* está disponible en TVB. Esta ofrece una serie de algoritmos con los que se pueden calcular distintas medidas que caracterizan la red cerebral. Estos algoritmos pueden ser seleccionados directamente en el apartado de análisis.

- Algoritmos de centralidad: miden la importancia o la influencia de los nodos en una red basándose en diferentes criterios. Se pueden usar para identificar nodos centrales, periféricos o intermediarios en una red.
- Algoritmos de grado y similaridad: miden el número, la suma o la similitud de las conexiones de cada nodo en una red. Se pueden usar para identificar nodos importantes, similares o aislados en la red.
- Algoritmos de *clustering*: miden la tendencia de los nodos a formar grupos o subredes con muchas conexiones internas y pocas conexiones externas.
- Algoritmos de densidad: miden la proporción de conexiones presentes en relación con las posibles conexiones en una red. También miden cómo se distribuyen las conexiones en el espacio físico.

5.10. Aplicación clínica

a) El paciente virtual epiléptico

En el apartado 4.4. a), se menciona la investigación desarrollada en [62], donde se explica cómo se puede reproducir en TVB la propagación a partir del foco epileptogénico hacia otras área de una hiperexcitabilidad anormal. En este apartado se va a explicar cómo se puede implementar este modelo en TVB.

El modelo de masa neuronal utilizado en el artículo es el ***'Epileptor'.*** Este tiene un parámetro que denomina x_0, el cual controla la excitabilidad del tejido. Si es mayor que el valor crítico $x_0 = -2{,}05$ esa región será capaz de crear estímulos epilépticos de forma

autónoma y si es menor será un tejido sano. Se van a definir tres tipos de regiones: epileptogénicas con x_0= *-1* (IPFCDL), zonas de propagación con x_0= *-2* (IPFCPOL, IPFCCL, IPFCDM) y zonas sanas con x_0= *-2,8* (por debajo del valor crítico). Para ello se establecen las tres configuraciones de parámetros para el mismo modelo de población neuronal, de la forma que es explicada en el apartado 5.3. b).

Como se puede observar (**Fig. 5.32**), inicialmente se produce una hiperactividad en la región epileptogénica IPFCDL que se va a transmitir posteriormente a las zonas de propagación mientras que el resto de las regiones van a mantener una dinámica normal.

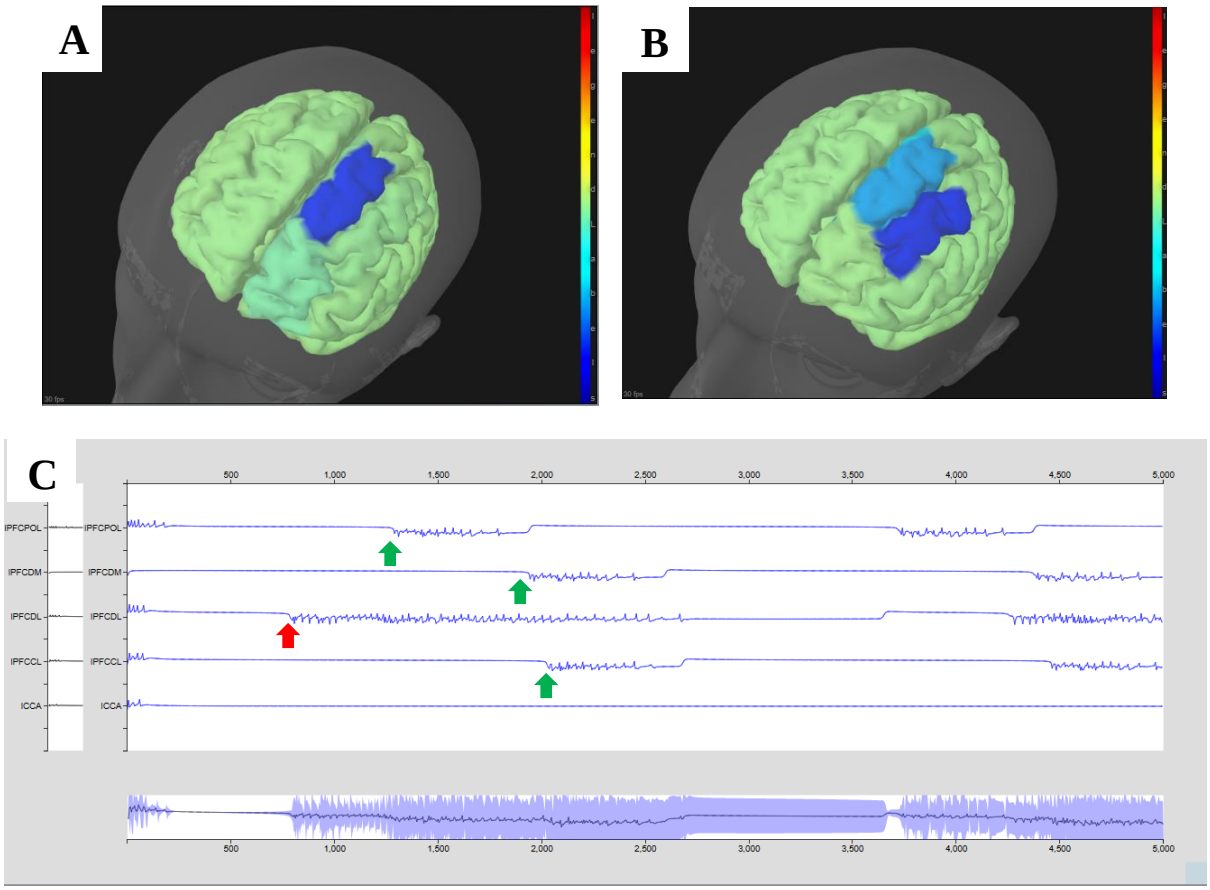


Figura 5.32. The Virtual Epileptic Patient. **(A)** Hiperactividad del foco epileptogénico. **(B)** Activación de las zonas de propagación. **(C)** Actividad de las distintas regiones. La flecha roja indica la activación del foco epileptogénico primero y las flechas verdes la transmisión de la actividad a las zonas de propagación. La región ICCA es una región sana que no se ve afectada por la dinámica de las demás regiones.

b) Infarto cerebral

De la misma forma que se hizo en [58], en TVB se puede reproducir la actividad de un cerebro que ha sufrido un infarto. Este tipo de lesiones suelen ser localizadas, es decir, se puede especificar en qué región se ha producido el daño. Se puede suponer que el daño causado es tan grave que elimina todas las conexiones de las regiones afectadas. Esto se puede modelar en TVB mediante el editor de matrices de conectividad. Para ello se seleccionan las regiones afectadas y se selecciona *"cut in"* y *"cut out"*, con lo que se reduce el valor de las conexiones, tanto las que llegan como las que parten de cierto nodo a cero. En nuestro caso, como se puede observar en **Fig. 5.33.** ¸las regiones afectadas son PCI, S1, TCC.

La simulación se realiza usando el modelo Stefanescu-Jirsa 3D, con los valores de los parámetros que se indican en el artículo. Este modelo es utilizado debido a que no depende demasiado de los retrasos sinápticos, siendo ideal para la señal BOLD debido a la baja resolución temporal de esta.

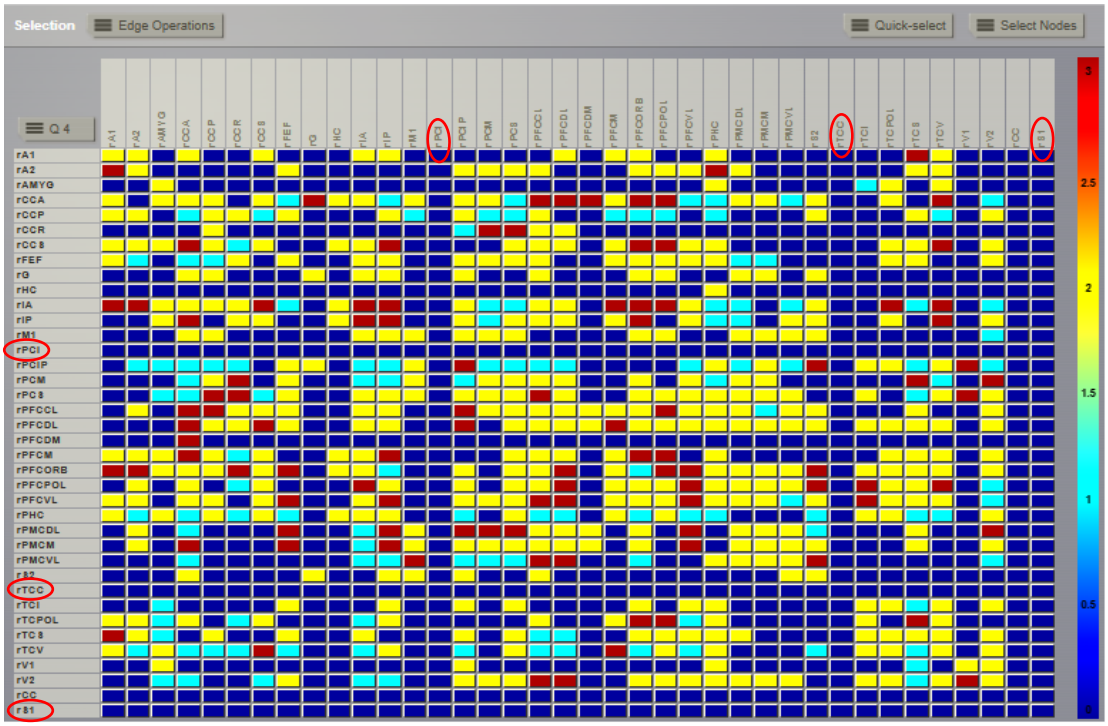


Figura 5.33. Matiz de conexiones de un sujeto que ha sufrido un ictus, afectando a las regiones PCI, TCC, S1.

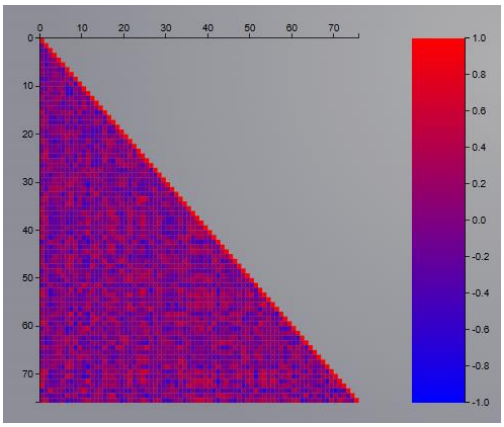


Figura 5.34. Matriz funcional de paciente que ha sufrido un infarto cerebral

6. Líneas futuras

El campo de la simulación cerebral todavía se encuentra dando sus primeros pasos. A pesar de que ha habido grandes avances en estos últimos años, todavía se está lejos de llegar a reproducir completamente la dinámica cerebral y entender cómo surge. El primer aspecto que debe ser mejorado, y que ha sido planteado a lo largo de todo el trabajo, es la integración multiescalar. El cerebro es un órgano que opera en diferentes niveles y los simuladores deben ser capaces de explicar cómo cambios desde el nivel subcelular pueden afectar al comportamiento de la totalidad del cerebro. Esto es algo en lo que se está investigando y, como se ha visto, se han obtenido resultados prometedores; pero todavía queda mucho trabajo hasta poder crear una teoría unificada de esta interacción multiescalar.

Si bien, para conseguir este objetivo la reproducción del conectoma cerebral neurona a neurona parece el planteamiento más obvio, quizás no sea el óptimo debido a los desafíos que plantea. La construcción de una red en la que se incluyan millones de neuronas junto con sus respectivas conexiones, necesitaría de unos requisitos computacionales elevados, además de su difícil implementación. La otra vía que se está desarrollando es el uso de campos neuronales que asuman la actividad de un conjunto de neuronas. Sin embargo, la mayoría de estos modelos son fenomenológicos y carecen de la posibilidad de explicar fenómenos a escalas inferiores. Por eso, una línea de investigación que parece razonable es el desarrollo de modelos mesoescalares a partir del comportamiento que surgiría en un circuito de cierto número de neuronas conectadas. Es decir, se debería de investigar la posibilidad de diseñar un microcircuito cuya actividad se vea modulada por sucesos a nivel subcelular o celular y que a partir de este se puedan crear un modelo mesoescalar, pero que a su vez incluya aspectos provenientes de niveles inferiores.

La inmensa mayoría de estos simuladores tan solo tienen en cuenta a las neuronas. Sin embargo, se debe recordar que estas no son el único elemento que compone el sistema nervioso, sino que también participan las células de la glía. Aunque su intervención en procesos neuronales todavía no está del todo clara, sí se ha visto que pueden ser partícipes de estos [39]. Por esta razón, estos componentes deben de ser tenidos en cuenta en futuros desarrollos de la simulación.

Muchos de los simuladores a gran escala tienden a considerar el cerebro aislado de su entorno. Sin embargo, no se debe perder la perspectiva de que el cerebro es el principal órgano de la función de relación de los seres vivos, que integra la información proveniente tanto del medio interno como externo y elabora una respuesta adecuada. Por esto, es necesario modelar estímulos más complejos, que respondan a ciertas situaciones, ya sean provenientes del entorno externo o del propio cuerpo. Sólo se podrá entender completamente el funcionamiento del cerebro cuando se comprenda cómo este interactúa con el ambiente.

7. Conclusiones

7.1. Grado de consecución de los objetivos

Se considera que los objetivos planteados han sido alcanzados en este trabajo. Se ha realizado una profunda descripción de diferentes simuladores cerebrales, realizando una comparación de estos, tanto teórica como experimentalmente, pudiendo extraer cuales son las ventajas y limitaciones de cada uno de ellos. Además, se han discutido sus aplicaciones actuales en la investigación y en la clínica mediante la realización de una revisión del estado del arte de los diferentes trabajos en los que estos simuladores son empleados, demostrando su utilidad. Por último, también se ha debatido sobre como existen diferentes métodos para abordar el problema de la integración multiescalar y hasta qué punto cada uno de estos simuladores consiguen realizarla.

7.2. Conclusiones extraídas

Los simuladores cerebrales son una herramienta de la neurociencia computacional que resultan útiles a la hora de comprobar hipótesis que tratan de explicar los sistemas complejos que existen en el cerebro, tanto sano como patológico, y que van a dar lugar al comportamiento que este presenta. Uno de los principales objetivos de estos softwares es convertirse en un puente entre las diferentes escalas y establecer como estás interactúan entre sí, encontrando una explicación a cómo la actividad de un gran número de neuronas van a dar a lugar a la dinámica de un gran circuito o del cerebro en su conjunto. Además, se ha podido comprobar que no solo resultan útiles en la investigación, sino también en la clínica, donde pueden tener diversas aplicaciones como el testeo de ciertos tratamientos antes de ser aplicados directamente al paciente y se presentan como un elemento que puede resultar de gran ayuda en la medicina personalizada.

A lo largo de este trabajo se han evaluado distintos simuladores neuronales y cerebrales. Cómo se ha podido comprobar, cada uno de ellos presenta una serie de ventajas e inconvenientes. Por esta razón, antes de elegir con qué tipo de software trabajar, es necesario definir previamente el problema que se quiere tratar y tener claros los requisitos necesarios para llegar a conclusiones acertadas. Por ejemplo, se debe especificar si se requieren modelos biofísicos detallados, si se quiere realizar una simulación a gran o a pequeña escala, etc. Además, se debe valorar las capacidades computacionales que se disponen para realizar la investigación, lo que determinará el nivel de complejidad del diseño de la red que va a ser emulada (por ejemplo, el número de neuronas o regiones cerebrales).

Respecto a los simuladores microescalares, se ha visto que existen diferentes opciones, entre las que se han valorado NEST, NEURON y Brian. NEST y NEURON ofrecen GUIs que pueden resultar útiles para los investigadores que posean pocos conocimientos de programación, aunque resulte recomendable tener conocimientos básicos. Brian ofrece una gran flexibilidad a la hora de definir los modelos, frente a los otros dos en los que es necesario utilizar modelos predefinidos. Por su parte NEURON, permite al usuario desarrollar modelos muy detallados biofísicamente. Por otro lado, NEST resulta más útil para aquellas redes que contengan modelos más sencillos, pero con un mayor número de elementos.

En cuanto a los simuladores macroescalares, TVB se presenta como un software con diversas funcionalidades y que permite reproducir diferentes estados cerebrales, así como diversas condiciones patológicas. Con el uso de masas y campos neuronales medios, se demuestra que no es necesario especificar las conexiones neurona a neurona para obtener resultados consistentes de la dinámica cerebral. Se posiciona como una herramienta de gran utilidad en la clínica debido a su capacidad de simular diferentes patologías que puede ayudar a determinar las causas de esta y como pueden afectar distintos tratamientos. Además, la posibilidad de realizar una exploración de parámetros y comparar los resultados obtenidos con datos de pacientes reales, permite ajustar adecuadamente los modelos que van a describir cómo se produce la dinámica cerebral. Está inferencia de parámetros va a ayudar a comprender qué procesos ocurren en el cerebro.

La gran cuestión a tratar es cómo estos simuladores son capaces de realizar una integración multiescalar y hasta que nivel de detalle son capaces de llegar. Si bien es cierto que ninguno de ellos tiene la capacidad de explicar todas las escalas en conjunto, algunos si son capaces de relacionar el comportamiento con otros niveles. Por ejemplo, en NetPyNe se demostró cómo cambios en parámetros del módulo RxD, que representa el nivel subcelular, afectaba a las señales de potencial de campo medio que se producían. Por su parte, el simulador NEST posee modelos compartimentales que incluyen canales iónicos que se pueden relacionar con la actividad conjunta de la red en su totalidad. Sin embargo, al igual que en Brian, los modelos compartimentales no son la especialidad de NEST. Por otro lado, TVB es capaz de conectar muy bien la nivel mesoscópico con el macroscópico, sin embargo es incapaz de abordar niveles inferiores. Por ello, se debe promover las co-simulaciones entre simuladores que abordan distintas escalas y que permitirá obtener una visión más general de todos los niveles que participan en la generación de la dinámica cerebral.

Sin embargo, todavía se presentan una serie de problemas y retos que deben ser resueltos. Lo primero que se debe destacar es que este es un campo que se encuentra en una etapa de desarrollo muy temprana, por lo que ningún simulador es capaz de obtener resultados completamente fieles a la realidad. Actualmente existe el debate de si para lograr este objetivo el gran desafío es aumentar el realismo biofísico de los modelos, lo que a priori puede parecer más obvio. Sin embargo, para algunos autores [2] el desarrollo de modelos más realistas no es tan esencial para conseguir resultados que permitan explicar los diferentes mecanismos del cerebro. Sería más importante comprender la disposición adecuada de los componentes neurocomputacionales en un cerebro artificial para reproducir los fenómenos de cognición biológica.

7.3. Aportaciones

En este trabajo se ha conseguido analizar abundante información, no solo sobre los diferentes simuladores, tanto micro como macroescalares, sino también sobre los distintos problemas a los que se enfrenta la simulación del cerebro y cómo pueden ser afrontados. Además los experimentos realizados con las GUI de NEST Desktop y NetPyNe-UI y las comparaciones realizadas, no se han visto descritas en la literatura y pueden ser útiles para aquellos investigadores que no posean altos conocimientos de

programación y que quieran trabajar con ellas, a la hora de decidir cual se puede ajustar más a su experimento y necesidades. Por último, el trabajo experimental realizado con TVB explora las diferentes posibilidades que este dispone y se describe un análisis detallado de las simulaciones disponibles y cómo ejecutarlas.

8. Bibliografía

[1] C. Eliasmith and O. Trujillo, “The use and abuse of large-scale brain models,” *Curr. Opin. Neurobiol.*, vol. 25, pp. 1–6, 2014.

[2] M. Colombo, “Why build a virtual brain? Large-scale neural simulations as jump start for cognitive computing,” *J. Exp. Theor. Artif. Intell.*, vol. 29, no. 2, pp. 361–370, 2017.

[3] G. T. Einevoll *et al.*, “The Scientific Case for Brain Simulations,” *Neuron*, vol. 102, no. 4, pp. 735–744, 2019.

[4] E. D’Angelo and V. Jirsa, “The quest for multiscale brain modeling,” *Trends Neurosci.*, vol. 45, no. 10, pp. 777–790, 2022.

[5] D. S. Bassett and O. Sporns, “Network neuroscience,” *Nat. Neurosci.*, vol. 20, no. 3, pp. 353–364, 2017.

[6] L. S. Costanzo, *Fisiología*, 5th ed. Elsevier, 2014.

[7] R. F. Betzel, J. D. Medaglia, and D. S. Bassett, “Diversity of meso-scale architecture in human and non-human connectomes,” *Nat. Commun.*, vol. 9, no. 1, 2018.

[8] B. Tunç and R. Verma, “Unifying inference of meso-scale structures in networks,” *PLoS One*, vol. 10, no. 11, pp. 1–14, 2015.

[9] A. Škoch *et al.*, “Human brain structural connectivity matrices–ready for modelling,” *Sci. Data*, vol. 9, no. 1, pp. 1–9, 2022.

[10] E. C. A. Hansen, D. Battaglia, A. Spiegler, G. Deco, and V. K. Jirsa, “Functional connectivity dynamics: Modeling the switching behavior of the resting state,” *Neuroimage*, vol. 105, pp. 525–535, 2015.

[11] T. Kirschstein and R. Köhling, “What is the source of the EEG?,” *Clin. EEG Neurosci.*, vol. 40, no. 3, pp. 146–149, 2009.

[12] F. Lopes da Silva, “EEG and MEG: Relevance to neuroscience,” *Neuron*, vol. 80, no. 5, pp. 1112–1128, 2013.

[13] M. Ramezanian-Panahi, G. Abrevaya, J. C. Gagnon-Audet, V. Voleti, I. Rish, and G. Dumas, “Generative Models of Brain Dynamics,” *Front. Artif. Intell.*, vol. 5, no. July, 2022.

[14] H. Paugam-Moisy and S. Bohte, “Computing with spiking neuron networks,” *Handb. Nat. Comput.*, vol. 1–4, pp. 335–376, 2012.

[15] W. GERSTNER, W. M. KISTLER, R. NAUD, and L. PANINSKI, *NEURONAL DYNAMICS: From Single Neurons to Networks and Models of Cognition*, no. Mi. 2014.

[16] A. N. Burkitt, “A review of the integrate-and-fire neuron model: I. Homogeneous synaptic input,” *Biol. Cybern.*, vol. 95, no. 1, pp. 1–19, 2006.

[17] G. Deco, V. K. Jirsa, P. A. Robinson, M. Breakspear, and K. Friston, “The dynamic brain: From spiking neurons to neural masses and cortical fields,” *PLoS Comput. Biol.*, vol. 4, no. 8, 2008.

[18] P. Sanz-Leon, S. A. Knock, A. Spiegler, and V. K. Jirsa, “Mathematical framework for large-scale brain network modeling in The Virtual Brain,” *Neuroimage*, vol. 111, pp. 385–430, 2015.

[19] P. Qu, L. Yang, W. Zheng, and Y. Zhang, “A review of basic software for brain - inspired computing,” *CCF Trans. High Perform. Comput.*, vol. 4, no. 1, pp. 34–42, 2022.

[20] “NEST - Tools - EBRAINS.” [Online]. Available: https://www.ebrains.eu/tools/nest.

[21] G. Marc-Oliver and D. Markus, “NEST (NEural Simulation Tool),” 2007. [Online]. Available: http://www.scholarpedia.org/article/NEST_%28NEural_Simulation_Tool%29. [Accessed: 10-Jul-2023].

[22] “Understand how NEST works.” [Online]. Available: https://nest-simulator.readthedocs.io/en/stable/understand_index.html. [Accessed: 10-Jul-2023].

[23] J. M. Eppler, M. Helias, E. Muller, M. Diesmann, and M. O. Gewaltig, “PyNEST: A convenient interface to the NEST simulator,” *Front. Neuroinform.*, vol. 2, no. JAN, pp. 1–12, 2009.

[24] F. Galluppi, A. Rast, D. Sergio, and S. Furber, “A General-Purpose Model Translation System for a Universal Neural Chip,” in *Conference: Neural Information Processing. Theory and Algorithms - 17th International Conference, ICONIP 2010, Sydney, Australia*, 2010, vol. 6444 LNCS, no. PART 2.

[25] “PyNN- tools - EBRAINS.” [Online]. Available: https://www.ebrains.eu/tools/pynn. [Accessed: 10-Jul-2023].

[26] “NEST Desktop - tools -EBRAINS.” [Online]. Available: https://www.ebrains.eu/tools/nest-desktop. [Accessed: 10-Jul-2023].

[27] “NEST Instrumentation App.” [Online]. Available: https://www.ebrains.eu/tools/nest-instrumentation-app. [Accessed: 10-Jul-2023].

[28] “The NESTML modeling language.” [Online]. Available: https://nestml.readthedocs.io/en/latest/index.html. [Accessed: 10-Jul-2023].

[29] T. C. Potjans and M. Diesmann, “The cell-type specific cortical microcircuit: Relating structure and activity in a full-scale spiking network model,” *Cereb. Cortex*, vol. 24, no. 3, pp. 785–806, 2014.

[30] M. Schmidt, R. Bakker, K. Shen, G. Bezgin, M. Diesmann, and S. J. van Albada, *A multi-scale layer-resolved spiking network model of resting-state dynamics in macaque visual cortical areas*, vol. 14, no. 10. 2018.

[31] B. Feldotto *et al.*, “Deploying and Optimizing Embodied Simulations of Large-Scale Spiking Neural Networks on HPC Infrastructure,” *Front. Neuroinform.*, vol. 16, no. May, pp. 1–23, 2022.

[32] “What is Neuron.” [Online]. Available: https://nrn.readthedocs.io/en/8.2.2/guide/what_is_neuron.html. [Accessed: 28-Jul-2023].

[33] “The NEURON Simulator.” [Online]. Available: https://nrn.readthedocs.io/en/8.2.2/. [Accessed: 28-Jul-2023].

[34] S. Dura-Bernal *et al.*, “NetpyNE, a tool for data-driven multiscale modeling of brain circuits,” *Elife*, vol. 8, pp. 1–26, 2019.

[35] R. A. McDougal, M. L. Hines, and W. W. Lytton, “Reaction-diffusion in the NEURON simulator,” *Front. Neuroinform.*, vol. 7, no. November, pp. 1–13, 2013.

[36] W. W. Lytton, A. H. Seidenstein, S. Dura-Bernal, R. A. McDougal, F. Schürmann, and M.

L. Hines, “Simulation Neurotechnologies for Advancing Brain Research: Parallelizing Large Networks in NEURON,” *Neural Comput.*, vol. 2733, no. March, pp. 2063–2090, 2018.

[37] M. Stimberg, R. Brette, and D. F. M. Goodman, “Brian 2, an intuitive and efficient neural simulator,” *Elife*, vol. 8, pp. 1–41, 2019.

[38] “Brian 2 documentation.” [Online]. Available: https://brian2.readthedocs.io/en/stable/. [Accessed: 28-Jul-2023].

[39] M. De Pitt and H. Berry, *Computational Glioscience*. 2022.

[40] R. A. Tikidji-Hamburyan, V. Narayana, Z. Bozkus, and T. A. El-Ghazawi, “Software for brain network simulations: A comparative study,” *Front. Neuroinform.*, vol. 11, no. July, pp. 1–16, 2017.

[41] P. Sanzleon *et al.*, “The virtual brain: A simulator of primate brain network dynamics,” *Front. Neuroinform.*, vol. 7, no. MAY, 2013.

[42] M. Marmaduke Woodman *et al.*, “Integrating neuroinformatics tools in TheVirtualBrain,” *Front. Neuroinform.*, vol. 8, no. APR, pp. 1–9, 2014.

[43] “The Virtual Brain: Scalable Brain Simulation.” [Online]. Available: https://www.thevirtualbrain.org/tvb/zwei/neuroscience-simulation. [Accessed: 16-Apr-2023].

[44] A. Solodkin, J. Zimmermann, A. R. McIntosh, L. Stefanovski, and P. Ritter, *Neurological biomarkers and neuroinformatics: The role of the virtual brain*. Elsevier Inc., 2018.

[45] V. K. Jirsa, O. Sporns, M. Breakspear, G. Deco, and A. R. Mcintosh, “Towards the virtual brain: Network modeling of the intact and the damaged brain,” *Arch. Ital. Biol.*, vol. 148, no. 3, pp. 189–205, 2010.

[46] A. Ghosh, Y. Rho, A. R. McIntosh, R. Kötter, and V. K. Jirsa, “Cortical network dynamics with time delays reveals functional connectivity in the resting brain,” *Cogn. Neurodyn.*, vol. 2, no. 2, pp. 115–120, 2008.

[47] C. J. Honey, R. Kötter, M. Breakspear, and O. Sporns, “Network structure of cerebral cortex shapes functional connectivity on multiple time scales,” *Proc. Natl. Acad. Sci. U. S. A.*, vol. 104, no. 24, pp. 10240–10245, 2007.

[48] S. A. Knock, A. R. McIntosh, O. Sporns, R. Kötter, P. Hagmann, and V. K. Jirsa, “The effects of physiologically plausible connectivity structure on local and global dynamics in large scale brain models,” *J. Neurosci. Methods*, vol. 183, no. 1, pp. 86–94, 2009.

[49] V. K. Jirsa, “Neural field dynamics with local and global connectivity and time delay,” *Philos. Trans. R. Soc. A Math. Phys. Eng. Sci.*, vol. 367, no. 1891, pp. 1131–1143, 2009.

[50] T. Proix, A. Spiegler, M. Schirner, S. Rothmeier, P. Ritter, and V. K. Jirsa, “How do parcellation size and short-range connectivity affect dynamics in large-scale brain network models?,” *Neuroimage*, vol. 142, pp. 135–149, 2016.

[51] P. Ritter, M. Schirner, A. R. Mcintosh, and V. K. Jirsa, “The Virtual Brain Integrates Computational Modeling and Multimodal Neuroimaging,” *Brain Connect.*, vol. 3, no. 2, pp. 121–145, 2013.

[52] C. J. Gauthier and A. P. Fan, “BOLD signal physiology: Models and applications,” *Neuroimage*, vol. 187, no. March 2018, pp. 116–127, 2019.

[53] G. Deco, V. Jirsa, and K. J. Friston, “The Dynamical and Structural Basis of Brain Activity,” in *Principles of Brain Dynamics: Global State Interactions*, M. I. Rabinovich, K. J. Friston, and P. Varona, Eds. 2012.

[54] K. J. Friston, A. Mechelli, R. Turner, and C. J. Price, “Nonlinear responses in fMRI: The balloon model, Volterra kernels, and other hemodynamics,” *Neuroimage*, vol. 12, no. 4, pp. 466–477, 2000.

[55] K. Shen, G. Bezgin, M. Schirner, P. Ritter, S. Everling, and A. R. McIntosh, “A macaque connectome for large-scale network simulations in TheVirtualBrain,” *Sci. Data*, vol. 6, no. 1, pp. 1–12, 2019.

[56] J. Zimmermann *et al.*, “Differentiation of Alzheimer’s disease based on local and global parameters in personalized Virtual Brain models,” *NeuroImage Clin.*, vol. 19, no. April, pp. 240–251, 2018.

[57] M. I. Falcon *et al.*, “The virtual brain: Modeling biological correlates of recovery after chronic stroke,” *Front. Neurol.*, vol. 6, no. NOV, pp. 1–13, 2015.

[58] M. I. Falcon, J. D. Riley, V. Jirsa, A. R. McIntosh, E. E. Chen, and A. Solodkin, “Functional mechanisms of recovery after chronic stroke: Modeling with the virtual brain,” *eNeuro*, vol. 3, no. 2, pp. 202–208, 2016.

[59] M. I. Falcon, V. Jirsa, and A. Solodkin, “A new neuroinformatics approach to personalized medicine in neurology: The Virtual Brain,” *Curr. Opin. Neurol.*, vol. 29, no. 4, pp. 429–436, 2016.

[60] K. Bansal, J. Nakuci, and S. F. Muldoon, “Personalized brain network models for assessing structure–function relationships,” *Curr. Opin. Neurobiol.*, vol. 52, pp. 42–47, 2018.

[61] S. Olmi, S. Petkoski, M. Guye, F. Bartolomei, and V. Jirsa, “Controlling seizure propagation in large-scale brain networks,” *PLoS Comput. Biol.*, vol. 15, no. 2, pp. 1–23, 2019.

[62] V. K. Jirsa *et al.*, “The Virtual Epileptic Patient: Individualized whole-brain models of epilepsy spread,” *Neuroimage*, vol. 145, pp. 377–388, 2017.

[63] L. Stefanovski *et al.*, “Linking Molecular Pathways and Large-Scale Computational Modeling to Assess Candidate Disease Mechanisms and Pharmacodynamics in Alzheimer’s Disease,” *Front. Comput. Neurosci.*, vol. 13, no. August, pp. 1–27, 2019.

[64] J. M. Meier *et al.*, “Virtual deep brain stimulation: Multiscale co-simulation of a spiking basal ganglia model and a whole-brain mean-field model with The Virtual Brain,” *Exp. Neurol.*, vol. 354, no. April 2021, 2022.

[65] T. Kunze, A. Hunold, J. Haueisen, V. Jirsa, and A. Spiegler, “Transcranial direct current stimulation changes resting state functional connectivity: A large-scale brain network modeling study,” *Neuroimage*, vol. 140, no. October, pp. 174–187, 2016.

[66] J. S. Goldman *et al.*, “Brain-scale emergence of slow-wave synchrony and highly responsive asynchronous states based on biologically realistic population models simulated in The Virtual Brain,” *bioRxiv*, p. 2020.12.28.424574, 2020.

[67] T. T. Nakagawa, V. K. Jirsa, A. Spiegler, A. R. McIntosh, and G. Deco, “Bottom up modeling of the connectome: Linking structure and function in the resting brain and their changes in aging,” *Neuroimage*, vol. 80, pp. 318–329, 2013.

[68] H. Aerts *et al.*, “Modeling brain dynamics in brain tumor patients using the virtual brain,” *eNeuro*, vol. 5, no. 3, 2018.

[69] H. Aerts *et al.*, “Modeling brain dynamics after tumor resection using The Virtual Brain,” *Neuroimage*, vol. 213, no. September 2019, p. 116738, 2020.

[70] M. Schirner *et al.*, “Brain simulation as a cloud service: The Virtual Brain on EBRAINS,” *Neuroimage*, vol. 251, no. November 2021, p. 118973, 2022.

[71] J. S. Goldman *et al.*, “A comprehensive neural simulation of slow-wave sleep and highly responsive wakefulness dynamics,” *Front. Comput. Neurosci.*, vol. 16, 2023.

[72] L. Kusch, S. Diaz, W. Klijn, C. Bernard, A. Morrison, and V. Jirsa, “Multiscale cosimulation design template for neuroscience applications,” *bioRxiv*, 2021.

[73] S. Khatoniar, “Blue Brain Technology: An Overview,” *J. Interdiscipl. Cycle Res.*, no. July 2020, 2020.

[74] B. V. Benjamin, N. A. Steinmetz, N. N. Oza, and J. J. Aguayo, “Neurogrid simulates cortical cell-types , active dendrites , and top-down attention,” *Neuromorphic Comput. Eng.*, 2021.

[75] V. Sharma, R. B. Vilarrubias, and P. F. M. J. Verschure, “BrainX3: A Neuroinformatic Tool for Interactive Exploration of Multimodal Brain Datasets,” in *Biomimetic and Biohybrid Systems. Living Machines*, 2023, pp. 157–177.