Universidad Nacional Autónoma de México

Facultad de Estudios Superiores Acatlán

Neurociencia computacional: Una configuración neuronal, Simulación, Algoritmos y Procesamiento de datos

T E S I S

que para obtener el título de:

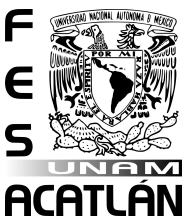
Licenciado en MAC

presenta:

Aaron Rodríguez Santiago

asesores:

Dra. Ingrid Chantal Torres Ramos
Dr. Javier Andres Orduz Ducuara



Naucalpan de Juárez, Estado de México, 2023

Agradecimientos

A la UNAM y a la Facultad de Estudios Superiores Plantel Acatlán

Quiero expresar mi agradecimiento a la UNAM, desde mi entrada al CCH Azcapotzalco y posteriormente mi ingreso a la licenciatura en la Facultad de Estudios Superiores plantel Acatlán, me siento orgulloso de formar parte de la UNAM, y agradezco el el trabajo de sus docentes, las instalaciones para el estudio y los recursos brindados para mi desarrollo profesional.

Al Dr. Andres Javier Orduz Ducara

Gracias al Dr. Andres Javier Orduz Ducara, por apoyarme durante una buena parte del proceso de concepción de este escrito, desde la elección del tema y el desarrollo del mismo, en especial proporcionándome ese impulso necesario para continuar durante el periodo de pandemia.

A la Dra. Ingrid Chantal Torres Ramos

Le agradezco a la doctora por su comprensión, apoyo y confianza cuando más lo necesite, brindándome la guía para dar conclusión a este trabajo, posteriormente dándome guía durante el proceso consecuente para la titulación, sus comentarios y observaciones han sido de gran ayuda y mucho valor.

A mis sinodales

Agradezco a mis sinodales, por su apoyo e interés en mi trabajo, gracias a sus comentarios y retroalimentación he podido mejorar la escritura y desarrollo del mismo, fue una grata experiencia contar con los puntos de vista de tantos profesionales y espero que sus expectativas hayan sido correspondidas.

A a mis padres y hermanas

Como pilar de mi vida le quiero agradecer a mi familia, a mis padres, mis guías desde el día que llegué a este mundo, dándome todo su apoyo y la oportunidad de concluir mis estudios en el área de estudio que más me gusta, también agradezco a mis hermanas, siempre presentes y brindándome su cariño y confianza, convirtiéndose en una razón para ser mejor cada día.

Índice general

Índice de figuras

Introducción

El sistema nervioso es el principal responsable de la comunicación de los órganos del ser humano, responsable de nuestro razonamiento, percepción e interacción con el mundo, la rama de la ciencia encargada de estudiar su funcionamiento es la neurociencia y es de aquí de donde parte el desarrollo del presente trabajo.

En el documento se parte de una breve introducción a este campo, pasando por el nacimiento de la neurociencia computacional y su desarrollo en la rama de la simulación neuronal y el procesamiento de bancos de datos.

El primer capítulo desarrolla algunos conceptos sobre cómo es que la neurona funciona, su trabajo como la unidad básica de comunicación en el sistema nervioso, los medios por los cuales lleva a cabo este proceso, terminando en un tipo de interacción particular (Ottoson, 1983).

Con la neurona como objetivo, el segundo capítulo describe el proceso por el cúal Alan Lloyd Hodgkin y Andrew Fielding Huxley llegaron a representar matemáticamente el estímulo de una neurona gigante de un axón, dando origen al modelo Hodgkin-Huxley (Koslow y Subramaniam, 2005) y haciéndolos acreedores de un premio nobel en fisiología (NobelPrize.org, 2009).

Durante el capítulo 3, se introduce el simulador NEST (Gewaltig y Diesmann, 2007), herramienta que permite interactuar con diferentes modelos, retomando el comportamiento experimental y permitiendo la interacción entre neuronas por medio de scripts en Python.

Finalmente, en el capítulo 4, se trabajó con el Allen Brain Map (for Brain Science, 2004), un proyecto perteneciente al Instituto Allen y que pone diferentes recursos a disposición de los interesados en el campo de la neurociencia computacional,

dividiendo sus recursos en atlas de datos enfocados en diferentes áreas, organismos y metodologías.

El trabajo parte con una introducción a la neurociencia computacional, con el objetivo de llevar a cabo una simulación neuronal y contrastar estos resultados con los obtenidos en un laboratorio como los del Instituto Allen, así un entusiasta en temas relacionados con neurociencia es capaz de simular una experiencia de laboratorio, sin las limitaciones que conlleva no contar con el equipo necesario.

Capítulo 1

La neurona sencilla

El sistema nervioso es el principal encargado en la regulación, el control y la coordinación de todas las funciones del cuerpo humano, este sistema es de acción rápida y se controla por medio de tejidos nerviosos (Melo Florián, 2011). Los tejidos nerviosos se comunican por medio de impulsos y se conectan a todas las partes del cuerpo humano partiendo del sistema nervioso central. Estos conceptos se ilustran en la figura 1.1.

Para definir la neurociencia es importante introducir algunos conceptos previos. Los humanos tienen sensores de entrada que llevan la información al sistema nervioso central y, posteriormente, a los órganos efectores, e.g. los músculos. La neurociencia es una campo interdisciplinario que provee una interpretación desde diferentes perspectivas sobre el funcionamiento del sistema nervioso. Estos enfoques van desde el punto de vista biológico, explicando como funciona una neurona; el psicológico, trabajando con la respuesta del cerebro al estrés; o por medio de la computación, interpretando y simulando los procesos cerebrales. La neurociencia se centra principalmente en el cerebro y su impacto en los comportamientos y funciones cognitivas (Brazier, 2018).

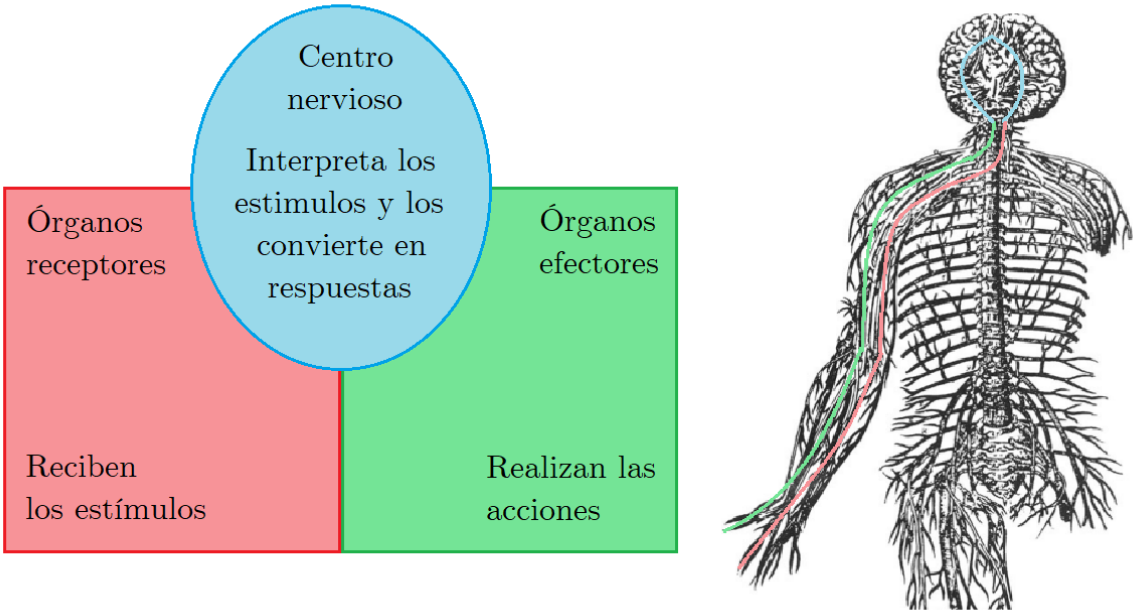


Figura 1.1: Un estimulo externo es captado por un órgano receptor, procesado por el centro nervioso y se expresa por medio de un órgano efector. *Nota.* Adaptado de *Séptima neurona figura* (p. 0), por D. Ottoson, 1983, Physiology of the nervous system

Es de esta doctrina multidisciplinaria de donde nace la neurociencia computacional, el estudio del cerebro humano tomando como base las matemáticas y aprovechando herramientas computacionales para trabajar con los datos. De esta manera se inicia este capítulo que retoma conocimientos necesarios sobre la fisiología del sistema nervioso y las neuronas para entender de donde surgen los modelos matemáticos.

1.1. Fisiología y Electrofisiología

La fisiología es la ciencia que trata las funciones orgánicas por las cuales se manifiesta y se mantiene la vida (Carratalá et al., 2000), estudia la estructura y el funcionamiento del cuerpo humano. Parte del objeto de estudio de la fisiología se encuentran los sistemas que interaccionan dentro del cuerpo humano, tales como, digestivo, locomotor, nervioso, entre

otros. En particular es de interés para este trabajo el sistema nervioso. Una manera de entender cómo es que la fisiología entiende al sistema nervioso es por medio del análisis del acto reflejo. El acto reflejo se define como una serie de pasos que lleva a cabo el cuerpo al recibir un estímulo externo, describe el proceso desde el estímulo hasta terminar en una reacción.

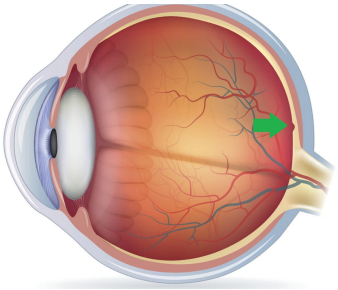


Figura 1.2: La luz recorre todo el ojo hasta llegar a la fóvea que es el punto con mayor sensibilidad a la luz. *Nota.* Adaptado de *Eye*, por Liu, 2020, Review of Myopia management (https://reviewofmm.com/can-the-al-cr-ratio-alone-be-used-to-determine-the-magnitude-and-or-progression-of-myopia/)

Por ejemplo, el acto reflejo comienza con un estímulo que llega a través del ojo (ver figura 1.2), por medio de fibras nerviosas aferentes, que llevan toda la información sensorial por el cuerpo, hasta llegar al centro nervioso. Cuando el centro nervioso recibe la información interpreta el impulso. Ahora la luz captada por el ojo tiene un nuevo significado, las longitudes de onda ahora tienen un significado dado por el cerebro, como lo puede ser una pelota moviéndose o una lampara encendida.

Con la información interpretada solo resta realizar la acción es por eso que el siguiente paso consiste en transferir este impulso a través del cuerpo, esto se logra por medio de neuronas eferentes, hasta llegar a los órganos efectores, que como lo indica el nombre lo indica, son los encargados de efectuar la acción, es así como la información de la pelota moviéndose puede provocar que el individuo lo esquive o la atrape.

El acto reflejo sucede en milésimas de segundo, es un proceso tan instantáneo que en 1844 (Ottoson, 1983) Johannes Müller dijo lo siguiente:

> *"El tiempo en que la sensación pasa del exterior del cerebro y a la espina dorsal de vuelta a los músculos para provocar una contracción es infinitesimalmente pequeña e inmedible"*

Esta cita sirve de contraste para ilustrar como era la perspectiva que se tenía de este

tipo de comunicación durante los inicios de la neurociencia.

El acto reflejo sirve de ejemplo de cómo es que se comunican las partes que conforman el sistema nervioso, y toda esta comunicación es posible gracias a las neuronas. Las neuronas son las encargadas de la parte funcional del tejido nervioso, se trata de células altamente especializadas (Puig et al., 2002), que son calificadas por su fisiología y morfología. La estructura básica de la neurona esta conformada por el núcleo y las dendritas, presentes en el cuerpo celular y nodos de Ranvier, células de Schwann, mielina y terminales como parte del axón ("Significado de Neurona", 2019, como se puede ver en la figura 1.3.

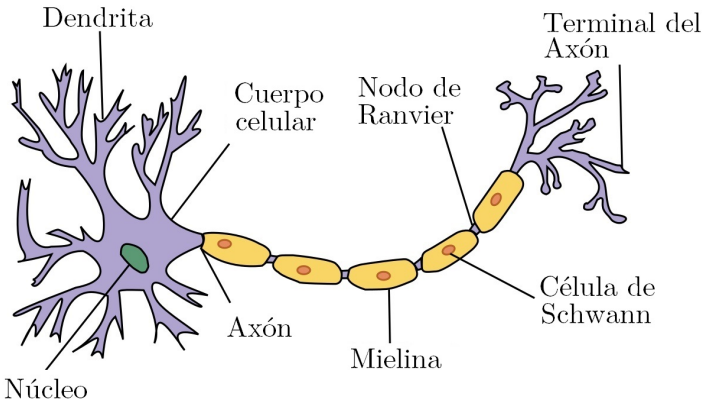


Figura 1.3: Cuerpo de una neurona. *Nota.* Adaptado de *Neuron*, por Quasar, 2011, Wikipedia (https://es.wikipedia.org/wiki/Neurona#/media/Archivo:Neuron_Hand-tuned.svg)

A pesar de su variedad las neuronas cuentan con una estructura en común que las identifica. Las neuronas cuentan con un soma o cuerpo neuronal donde se encuentran la mayoría de sus organelos, como la mitocondria, aparato de Golgi, retículo endoplásmico liso, rugoso y el núcleo de la neurona. Estos organelos cumplen su función, pero para el análisis que se va a llevar a cabo sirve destacar el núcleo que es el encargado de la producción de neurotransmisores, estos organelos pertenecen al soma de la neurona.

Pegadas a las neuronas se pueden localizar las dendritas, se trata de pequeños receptores conectados al soma de la neurona. Las dendritas reciben información proveniente de otras neuronas por medio de los neurotransmisores o señales eléctricas.

Las dendritas son las herramientas con las que cuenta la neurona para la recepción de información.

Además de las dendritas para la recepción de estímulos, la neurona cuenta con axones para la transmisión, se trata de uno o más brazos largos rodeados por vainas de mielina. La intención del recubrimiento es favorecer la conducción del impulso a través del axón, la mielina es producida en intervalos regulares por la célula de Shwann (López Lombana y Hurtado Giraldo, 1993), entre estos recubrimientos de mielina quedan secciones llamadas nodos de Ranvier, esta estructura intercalada facilita la transmisión y permite que los impulsos se extiendan por el sistema nervioso, estos axones pueden ser tan largos como lo requiera el sistema, es por medio de los axones que el mensaje es transmitido desde el núcleo de la neurona hasta que llega a la terminal del axón, donde los neurotransmisores son encapsulados por vesículas sinápticas y se expulsan por los botones sinápticos al final del axón.

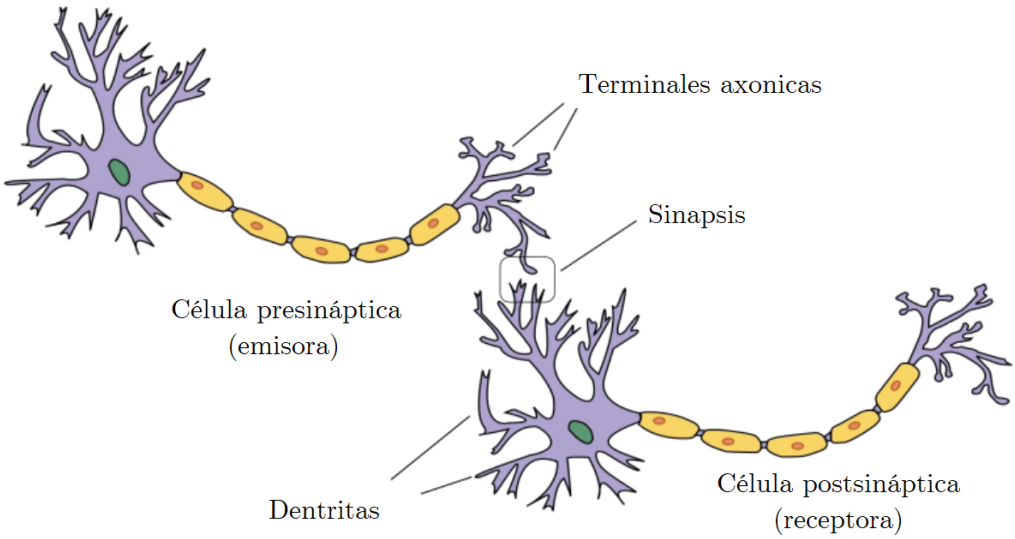


Figura 1.4: Los términos presinápticos y postsinápticos hacen referencia al proceso de comunicación entre neuronas por medio de las dendritas ("Significado de Neurona", 2019)

El punto de comunicación donde la neurona emisora y la neurona receptora se comunican, se llama sinapsis, es de aquí de donde surge el concepto de neurona presináptica para la emisora y postsináptica para la receptora. La sinapsis es un punto milimétrico donde se realiza este intercambio de información entre las terminales del

axón de la neurona presináptica y las dendritas de la neurona postsináptica como se muestra en la figura 1.4.

Toda la neurona se encuentra rodeada por una capa bilipídica de alrededor de 5nm de espesor, esta es la membrana de la neurona. La membrana sirve para aislar el interior de la neurona del exterior, la neurona cuenta con un gran número de iones en su interior y es gracias a la membrana que se puede mantener un desequilibrio iónico.

El desequilibrio iónico de la neurona con su exterior es aquello que da la vida a la neurona misma. La membrana cuenta con canales iónicos que permiten la entrada y salida de iones, los mensajes que se han mencionado reiteradamente en este documento pueden ser interpretados por medio de la lectura de la diferencia iónica del interior de la neurona con el exterior, es de este desequilibrio iónico y su intercambio que aparece el potencial de membrana.

Este potencial de membrana es el inicio de donde nace la electrofisiológica y la posibilidad de cuantificar la comunicación entre neuronas.

1.2. Comunicación neuronal y mediciones

La electrofisiología neuronal describe cómo se comporta eléctricamente una neurona y cuál es el proceso que da origen a estos comportamientos, nace de las teorías de Luigi Galvani en el siglo XVIII, que al manipular los nervios de las patas de una rana y ver su reacción concibe el concepto de electricidad, primer acercamiento a la electrofisiología que continua su desarrollo durante los siglos posteriores.

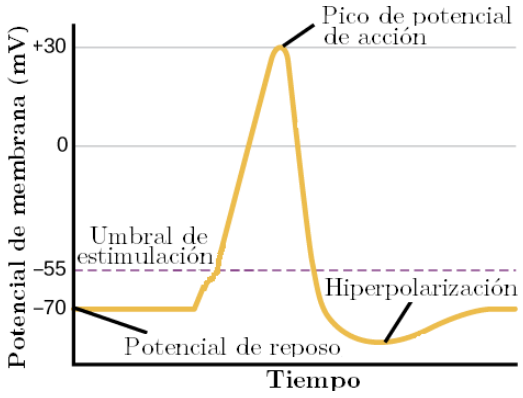


Figura 1.5: Potencial de membrana en función del tiempo . *Nota.* Adaptado de *Cómo se comunican las nuronas:figura3*, por OpenStax, 2016, OpenStax(https://openstax.org/)

Como se mencionó anteriormente las

neuronas se encuentran en un desbalance constante entre la cantidad de iones dentro y fuera de la membrana, formando un gradiente químico y eléctrico. Los iones encargados de provocar este gradiente son en su mayoría sodio, cloro y potasio. Esta gradiente varia a través del tiempo como se puede observar en la figura 1.5. El potencial de membrana cambia durante un estímulo que cambiará el voltaje de la neurona hasta llega al pico de potencial de acción, donde regresa a su estado de reposo por medio de una repolarización y una hiperpolarización, a todo este proceso se le llama potencial de acción ("How neurons communicate - biology", 2015).

Este gradiente es conectado por los canales iónicos. Los canales iónicos son principalmente proteínas presentes en la membrana neuronal que se vuelven permeables durante un periodo determinado de tiempo, esta activación es controlada por los neurotransmisores secretados por los botones sinápticos, la reacción de la neurona puede ser del tipo excitatorio o inhibitoria, dependiendo del canal iónico y el neurotransmisor, como lo puede ser el caso del glutamato, un neurotransmisor excitatorio y que les da instrucciones a los canales iónicos.

El glutamato es el principal neurotransmisor excitatorio del sistema nervioso por lo que es útil para describir el proceso de una manera general. El glutamato se concentra en las vesículas sinápticas de la neurona presináptica, donde es liberado en el espacio sináptico, aquí llega a los receptores AMPA[1] y NMDA[2], en esta parte del proceso el neurotransmisor ya cruzó de la parte presináptica a la postsináptica.

Con el estímulo llegando a los receptores AMPA y NMDA, suceden dos reacciones. El receptor AMPA comienza su reacción, se vuelve permeable a las moléculas de sodio permitiendo su acceso a la neurona postsináptica. A su vez el receptor NMDA se encuentra bloqueado por una molécula de magnesio, esta es desplazada por las cargas entrantes de los iones de sodio, en este momento el receptor comienza a ser permeable a las moléculas de sodio y calcio, con preferencia a estas últimas, así es como los receptores modifican tanto el gradiente químico como el gradiente eléctrico.

Tras el proceso de reacción de los receptores la neurona postsináptica cuenta con

[1]Receptor del ácido α-amino-3-hidroxi-5-metil-4-isoxazol
[2]Receptor N-metil-D-aspartato

un mayor número de iones positivos en su interior. La neurona que se encontraba en un estado de reposo con una carga de -70mV comienza a despolarizarse. El potencial de membrana comienza a variar llegando a un punto máximo y es entonces cuando la neurona necesita deshacerse de sus cargas y comienza el proceso de hiperpolarización.

Durante el estado de hiperpolarización de la neurona esta entra a un estado refractario. El estado refractario consulta si existe una señal de mayor intensidad, si la recibe repite el proceso de despolarización, pero en caso contrario se llega a un estado refractario absoluto, donde la neurona ya no puede recibir más estímulos durante un tiempo. Un ejemplo de esto lo que se observa cuando al recibir un estímulo externo de manera constante este va perdiendo efectividad.

A estos puntos de reacción se les conoce como potencial de acción. El potencial de acción cuenta con un límite superior, por lo que para poder dar la señal al cuerpo de que un estímulo es mayor no se puede aumentar el voltaje de la membrana, en lugar de esto lo que realizan la neurona es aumentar la frecuenta del potencial de acción durante el periodo de refracción. Estos potenciales de acción son los que van a describir el comportamiento de las neuronas y se trata de un concepto clave para el desarrollo de este documento.

Es así como se puede comenzar a esquematizar el proceso de reacción de una neurona de manera general. Este proceso parte de la neurona presináptica y concluye con la neurona postsináptica volviendo al estado de reposo, los pasos serían los siguientes:

1. La neurona presináptica con ayuda de calcio encapsula los neurotransmisores y los libera en el espacio sináptico.

2. Una parte de los neurotransmisores es captada por receptores especializados y los residuos vuelven a la neurona presináptica con ayuda de los astrocitos.

3. Los receptores comienzan a ser permeables a determinados iones despolarizando la neurona y al superar el umbral aumenta mucho más el potencial de membrana gracias al potencial de acción.

4. Al llegar al límite la neurona comienza a llevar a cabo procesos de hiperpolarización llegando a un estado de refractario donde espera una nueva señal.

   a) Si se recibe una señal de mayor intensidad se vuelve al paso 3

   b) Si no se recibe una señal de mayor intensidad pasa al estado refractario absoluto.

5. En el estado basal la neurona vuelve a organizarse para volver a ser viable para la recepción de un impulso.

Este es el proceso básico de excitación de una neurona, aunque existen algunas variantes. Una variante de esto es el proceso activado por el neurotransmisor GABA (*Gamma-Aminobutyric Acid*), que es el principal neurotransmisor inhibitorio del sistema nervioso donde en lugar de llevar a cabo todo el proceso de activación estimula procesos que evitan la excitación con propósitos diferentes. De manera general este es el esquema que se continuará explorando al hablar de la comunicación en neuronas particulares.

Por ejemplo, para que la luz entre al ojo se ve el proceso presente en la figura 1.6. ① La luz impacta con la córnea y ② atraviesa todo el ojo hasta impactar con la retina donde ③ los fotorreceptores conocidos como bastones y codos se comunican por medio de las células bipolares, hasta llegar a ④ las células ganglionares, conectando con el nervio óptico ⑤.

1.3. Visión y transmisión de la imagen

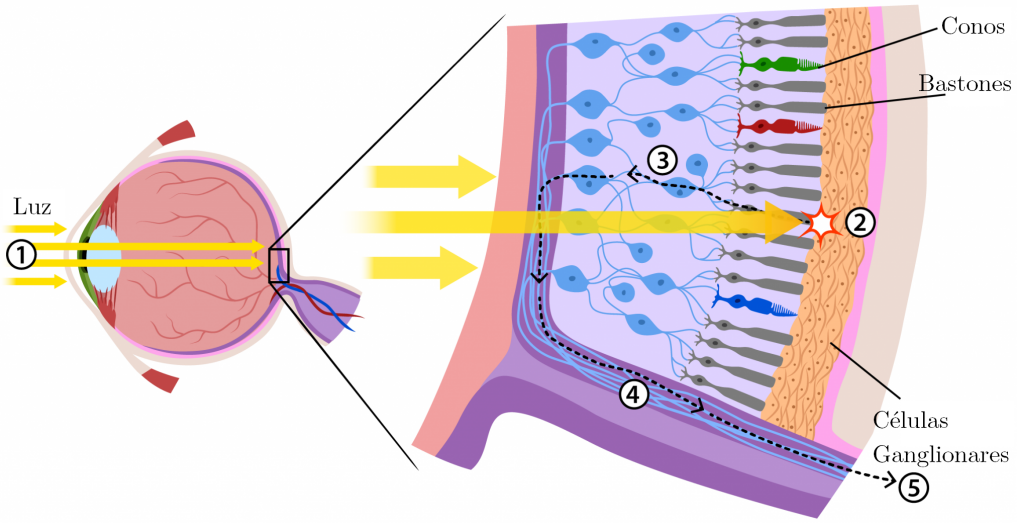


Figura 1.6: Neuronas presentes durante la visión. *Nota.* Adaptado de *Rods and cones*, por Kazilek y Cooper, 2010, Arizona State University (https://askabiologist.asu.edu/rods-and-cones)

El repaso al proceso de comunicación neuronal da las bases para entender el funcionamiento principal del sistema nervioso, un enfoque más particular se puede contemplar al revisar un proceso en singular como la visión. Debido a su relevancia y a la cantidad importante de recursos con los cuales se cuentan sobre la visión será uno de los principales enfoques que se revisarán en la tesis, es por eso que se hará un repaso de cómo es que el proceso de comunicación que involucra la visión para a través del sistema nervioso.

Dentro del esquema del acto reflejo visto al comienzo del capítulo, se menciona lo que sucede desde que el estímulo es recibido por el órgano receptor hasta que se llega por medio de las neuronas aferentes al centro nervioso, la visión sigue el mismo proceso para la visualización e interpretación de la imagen.

En los mamíferos el proceso de visualización comienza en el órgano receptor del ojo. El ojo cuenta con su morfología particular, cristalino, iris, pupila, córnea, entre otros, cada uno cumple su función, esto puede verse con mayor detalle en un curso especial de morfología. Para mantener el enfoque en la visión desde la perspectiva neuronal no se explicará toda la fisiología del ojo, pero si se mencionan algunas partes

que es útil conocer.

El ojo es un órgano receptor de luz, es por eso que es importante dar un repaso a las partes del ojo que tienen una función primordial durante la manipulación de la luz que dará origen a la visión. Para comenzar, el cristalino se encuentra al centro del ojo, se trata de una lente que por medio de ligamentos puede refractar la luz, dirigiendo la luz a la retina, al fondo del ojo y en un punto especifico de esta se encuentra la fóvea, donde se regresa el campo de las neuronas ya que es el punto donde se encuentra el mayor punto de neuronas fotorreceptoras.

De manera más localizada se puede hablar de la retina. Toda la retina del ojo cuenta con terminaciones nerviosas, estas son activadas tras la metabolización de dos proteínas que reaccionan con la luz solar, estas son las Rodpsina y la Iodopsina (Wheway et al., 2019), estas proteínas entran en contacto con los bastones y conos presentes en la retina, siendo los principales fotoreceptores.

Comenzando por los bastones, estas son neuronas especializadas que convierten el estímulo visual que llega como fotones en un impulso químico eléctrico procesado por el sistema nerviosos central, estas son las encargadas de percibir el tamaño, la forma y el brillo del entorno. La diferencia en concentración entre los bastones y conos es casi de 130 millones con respecto a 7 millones (“Rod”, 2007), los bastones tienen una función más general dentro de la visión.

La otra neurona fotorreceptora son los conos. Los conos están especializados en la captación de los colores y detalles, debido a su menor concentración están más especializadas en su labor siendo especialmente efectivas al reaccionar a niveles menores de luz (Rodgers, 2007).

Cuando la rodopsina y la iodopsina entran en contacto con los fotones a frecuencias específicas reaccionan, estas frecuencias se pueden observar en la gráfica 1.7, activando los fotorreceptoras que comienzan la liberación de Glutamato, dando inicio al proceso revisado durante el sistema anterior.

La iodopsina recibe las longitudes de los colores azul verde y rojo, por su parte la rodopsina reacciona como se ve en la línea punteada, estas proteínas son metabolizadas y capturadas por las neuronas (Terakita, 2005), esto se puede observar

en la figura 1.7.

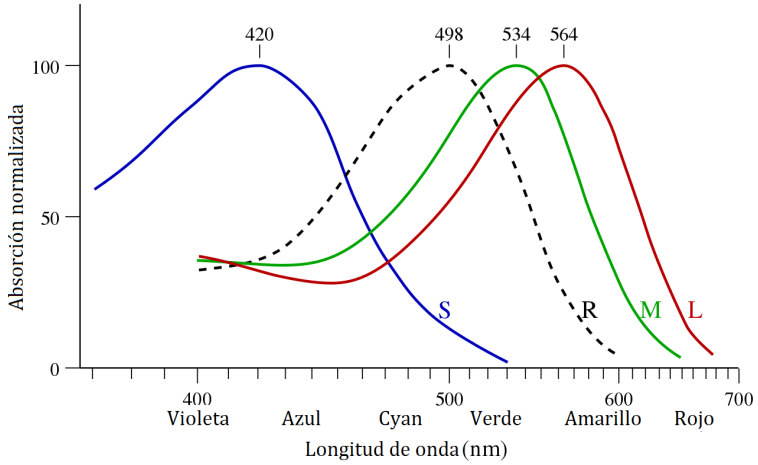


Figura 1.7: *Nota.* Adaptado de *Cone-response.svg*, por DrBob, 2011, Wikimedia (https://commons.wikimedia.org/wiki/File:Cone-response-de.svg)

El proceso de reacción de las neuronas comienza tras la metabolización de la iodopsina y rodopsina al contacto con la luz, esto hace que la neurona reaccione destruyendo el AMP cíclico, molécula energética que mantiene los canales iónicos permeables; cerrando los canales iónicos que permitían el acceso del sodio, provocando el potencial de acción y terminando con la hiperpolarización de las neuronas, así termina la reacción de las neuronas.

Posterior a la reacción de las neuronas fotorreceptoras el proceso continua influenciando a otras neuronas. Tras los fotorreceptores siguen las neuronas horizontales, encargas de regular la cantidad de neurotransmisores, las neuronas bipolares que transmiten el mensaje a las neuronas amacrinas, más variadas en sus funciones, como lo son los estímulos persistentes como la luz o la interacción directa con los bastones, finalizando con las neuronas ganglionares, llegando hasta el nervio óptico (Purve et al., 2012).

Todas estas neuronas juegan un papel primordial en la visión, por medio del intercambio de neurotransmisores las neuronas fotorreceptoras, bipolares y ganglionares se encargan de llevar a cabo la transmisión de información al nervio óptico (Purve et al., 2012) que se encargará de interpretar la información recopilada.

El estudio de la corteza visual como sección del cerebro encargada de la

interpretación de la información captada por los ojos, nace de los estudios de David Hubel y Torsten Wiesel en primates. Durante sus estudios fueron capaces de dimensionar la cantidad de neuronas involucradas llegando al orden de miles en cuestión de milésimas de segundo (Melo Florián, 2011).

Es por la cantidad de neuronas involucradas en un periodo tan corto de tiempo que es tan difícil obtener información particular sobre un proceso llevado a cabo por las neuronas. (Dayan y Abbott, 2001). Es de esta necesidad de control que surge la idea de observar las neuronas, modelarlas y mediante el uso de los mismo en simulación es, obtener resultados.

Capítulo 2

Matemáticas neuronales

Existe una gran diversidad de neuronas cada una con sus características y funciones diferentes, por ejemplo para la comunicación del impulso visual son necesarios cinco tipos de neuronas diferentes en la retina y un tipo más en el nervio óptico, sus variaciones van desde los neurotransmisores hasta la forma en que se comunican.

Ahora bien, a pesar de esta variedad, se puede abstraer el concepto de neurona hasta un punto donde se pueda dar explicación a los procesos que lleva a cabo. En este capítulo se hará un resumen de como se llevó a cabo una de las primeras interpretaciones matemáticas de la neurona, con el objetivo de realizar una primera digitalización del proceso algorítmicamente.

2.1. Modelación neuronal

El objetivo principal de este tema es exponer cómo se pasa de un proceso biológico natural a un equivalente que modelar por medio de ecuaciones. Es decir, se presenta un modelo como un circuito eléctrico, así mismo, esta sección está basada fuertemente en el curso impartido por el profesor Michael Fee en el 2018, **Introduction to Neural Computation** (Fee y Zysman, 2018).

Para empezar se observa a la neurona como un medio separado de líquido intra y extra celular; en otras palabras, se puede interpretar como un medio acuoso donde

iones de sodio y cloro se encuentran disueltos, de modo que estos iones pueden moverse de maneras distintas, este movimiento surge de la difusión generada por los gradientes eléctricos o químicos.

Partiendo de la idea de un ion de sodio como elemento libre, sin interacción con los gradientes que se desplaza en el medio acuoso, este se comporta como si se tratara de una caminata aleatoria moviéndose en una dirección u otra de manera imparcial, una caminata aleatoria unidimensional (Berg, 2018).

Para ilustrar este comportamiento se toma en cuenta el movimiento en dos direcciones, izquierda o derecha a una velocidad v_x por un tiempo τ antes de cualquier colisión y considerando la caminata aleatoria de la siguiente manera:

- La mitad de las partículas se mueven a la derecha con una distancia $\delta = v_x\tau$
- La otra mitad de las partículas se mueven a la izquierda con una distancia δ
- Ahora N partículas comienzan en la posición $x = 0$ en el tiempo $t = 0$
- La posición i-ésima partícula en el paso n. se describe como $x_i(n)$
- Ahora se asume que el movimiento de cada partícula es independiente

Entonces se puede escribir que la posición de cada partícula en el paso n es la función de posición en el paso anterior:

$$x_i(n) = x_i(n-1) \pm \delta \tag{2.1}$$

Esta es una relación de recurrencia, con la condición inicial $x_i(0) = 0$ y $n \geq 1$, que conforman la definición recursiva. Por otra parte, se puede definir la **posición promedio** de i partículas en el tiempo n como:

$$\Big\langle x_i(n) \Big\rangle_i = \frac{1}{N} \sum_i x_i(n)$$

Usando la ecuación 2.1:

$$\Big\langle x_i(n) \Big\rangle_i = \frac{1}{N} \sum_i [x_i(n-1) \pm \delta]$$

$$= \frac{1}{N} \sum_i [x_i(n-1)] + \frac{1}{N} \sum_i [\pm\delta]$$

Como se definió al comienzo la mitad de las partículas se desplazan una distancia $+\delta$ y la otra mitad a $-\delta$ por lo que la suma de estas distancias es tiende a 0 de manera asintótica o cuando N es par:

$$\Big\langle x_i(n) \Big\rangle_i = \frac{1}{N} \sum_i [x_i(n-1)] + \frac{1}{N} \cancelto{0}{\sum_i [\pm\delta]}$$

$$= \frac{1}{N} \sum_i [x_i(n-1)]$$

Que por definición es:

$$\Big\langle x_i(n) \Big\rangle_i = \Big\langle x_i(n-1) \Big\rangle_i$$

Por lo tanto, se conoce que la posición promedio de i partículas en el tiempo n es igual al promedio en el tiempo anterior.
Se define el **valor absoluto** de la distancia de una partícula con respecto al origen como:

$$\Big\langle \mid x_i(n) \mid \Big\rangle_i = \sqrt{\Big\langle x_i^2(n) \Big\rangle_i}$$

Y para llegar a esto se toma como antecedente, la posición promedio al cuadrado:

$$\left\langle x_i^2(n) \right\rangle_i = \frac{1}{N} \sum_i x_i^2(n) \tag{2.2}$$

De la ecuación anterior se trabaja con la suma trabajando con $x^2(n)$:

Usando la ecuación 2.1:

$$\begin{aligned} x_i^2(n) &= (x_i(n-1) \pm \delta)^2 \\ &= x_i^2(n-1) \pm 2\delta x_i(n-1) + \delta^2 \end{aligned}$$

Sustituyendo en la ecuación 2.2

$$\left\langle x_i^2(n) \right\rangle_i = \frac{1}{N} \sum_i \left[x_i^2(n-1) \pm 2\delta x_i(n-1) + \delta^2 \right]$$

Agrupando en promedios:

$$\left\langle x_i^2(n) \right\rangle_i = \left\langle x_i^2(n-1) \right\rangle_i + \left\langle \pm 2\delta x_i(n-1) \right\rangle_i + \left\langle \delta^2 \right\rangle_i$$

El segundo término es la suma y resta de las δ por lo que tiende a ∞ y el tercer término es independiente del promedio, por lo que sólo queda:

$$\begin{aligned} \left\langle x_i^2(n) \right\rangle_i &= \left\langle x_i^2(n-1) \right\rangle_i + \cancelto{0}{\left\langle \pm 2\delta x_i(n-1) \right\rangle_i} + \left\langle \delta^2 \right\rangle_i \\ &= \left\langle x_i^2(n-1) \right\rangle_i + \delta^2 \end{aligned}$$

Esta es la varianza del movimiento y en cada paso aumenta δ^2.

Iniciando desde el momento 0 igualado con $x_i(0) = 0$, a partir de $n \geq 1$, se obtiene:

$$\begin{aligned}\left\langle x_i^2(1)\right\rangle_i &= \delta^2 \\ \left\langle x_i^2(2)\right\rangle_i &= 2\delta^2 \\ &\vdots \\ \left\langle x_i^2(n)\right\rangle_i &= n\delta^2\end{aligned}$$

Entonces:

$$\left\langle x_i^2(n)\right\rangle_i = \frac{\delta^2 t}{\tau} \quad \text{con} \quad n = \frac{t}{\tau}$$

Ahora se define el coeficiente de difusión como:

$$D = \frac{\delta^2}{2\tau} \tag{2.3}$$

Dejando el promedio al cuadrado:

$$\left\langle x_i^2(t)\right\rangle_i = 2Dt$$

Y volviendo al valor absoluto se tiene que:

$$\left\langle \mid x_i(t) \mid \right\rangle_i = \sqrt{\left\langle x_i^2(t)\right\rangle_i} = \sqrt{2Dt} \tag{2.4}$$

Con esto se puede calcular cuanto tiempo tarda un ión en recorrer un a distancia por difusión, en el caso de los iones, la constante de difusión esta dado por $D = 10^{-5}\frac{\text{cm}^2}{s}$. El lado izquierdo de la ecuación 2.4 llamada distancia absoluta, suele ser etiquetada como L, además se resalta el hecho de que representa la distancia mínima promedio de cualquier partícula i respecto al origen y está dada por:

$$\begin{aligned}L &= \sqrt{2Dt} \\ L^2 &= 2Dt\end{aligned}$$

Luego, el tiempo:

$$t = \frac{L^2}{2D} \tag{2.5}$$

Entonces con el coeficiente de un ion a una distancia absoluta de $10\mu\,\mathrm{m} = 10^{-3}\,\mathrm{cm}$

$$t = \frac{10^{-6}\,\mathrm{cm}^2}{2(10^{-5})\frac{\mathrm{cm}^2}{\mathrm{s}}} = 0{,}5\,\mathrm{s} = 500\mathrm{m}\,\mathrm{s}$$

Otro ejemplo es el nervio isquiático que surge desde el último nervio lumbar y puede llegar a los muslos con una extensión de casi $1\mathrm{m} = 10^2\mathrm{cm}$. Al calcular el tiempo, t, se obtiene:

$$t = \frac{10^4\,\mathrm{cm}^2}{2(10^{-5})\frac{\mathrm{cm}^2}{\mathrm{s}}} = 5 * 10^8\,\mathrm{s} \approx 138888\,\mathrm{h}$$

Equivalente a decir que un estimulo neuronal tomaría más de 15 días para recorrer un metro de distancia.

En conclusión la comunicación neuronal, únicamente por medio de difusión es un proceso inviable para el sistema nervioso, y con esto se puede observar un caso sencillo que involucra el término de coeficiente de difusión.

Por otro lado, como se vio en el capítulo anterior, los iones están expuestos a un gradiente químico, por la diferencia de concentraciones entre el interior y el exterior de las neuronas.

Gracias a esta diferencia de concentraciones se puede rescatar el tema de las leyes de Fick, donde Adolf Fick enunciaba que el flujo de un material es proporcional al gradiente de su concentración (Philibert, 2006) y así poder incorporar el comportamiento al modelo matemático.

Para ilustrar mejor este comportamiento se define $N(x)$ es el número de partículas en la caja en la posición x, ver figura 2.1.

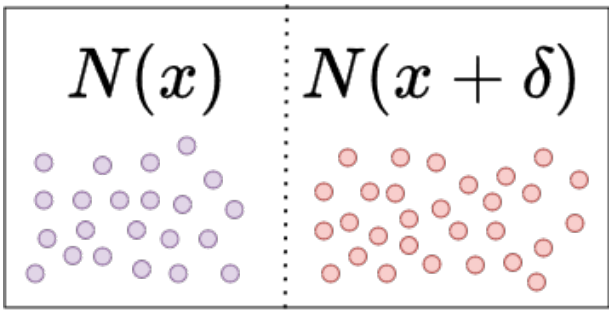


Figura 2.1: El diagrama representa la diferencia de concentraciones en un ensamble

De estas partículas existe una posibilidad del 50 % de que se desplacen de su lugar o 50 % de posibilidades de que se mantengan en su posición:

$$\frac{1}{2}\left[N(x) - N(x-\delta)\right]$$

Como el número de partículas que se mueven a la derecha en un intervalo de tiempo τ. Ahora se escribe la ecuación del flujo como el resultado del producto diferencia de concentraciones φ sobre la distancia:

$$J_x \quad = \quad -D\frac{1}{\delta}\left[\varphi(x+\delta) - \varphi(x)\right]$$

Fórmula que se puede reescribir como una ecuación diferencial por su estructura:

$$J_x \quad = \quad -D\frac{\partial \varphi}{\partial x} \tag{2.6}$$

Obteniendo la primera ley de Fick, donde el gradiente de concentración disminuye y este intercambio produce corrientes que obedecen las leyes de Ohm:

$$\Delta V = R \cdot I \tag{2.7}$$

Donde:

- I es la corriente (Amperes, A)

- ΔV es la diferencia de voltaje (Volts, V)
- R es la resistencia (Ohms, Ω)

Dicho brevemente, la diferencia de concentración entre el medio intra y extracelular da origen al intercambio iónico que será interpretado como un circuito eléctrico.

La representación de la neurona como circuito (ver figura 2.2) considera el líquido intra y extracelular de la neurona como el cable, la membrana como una capacitor y los canales iónicos como resistencias, dando como resultado la corriente $I = \frac{\Delta V}{R}$

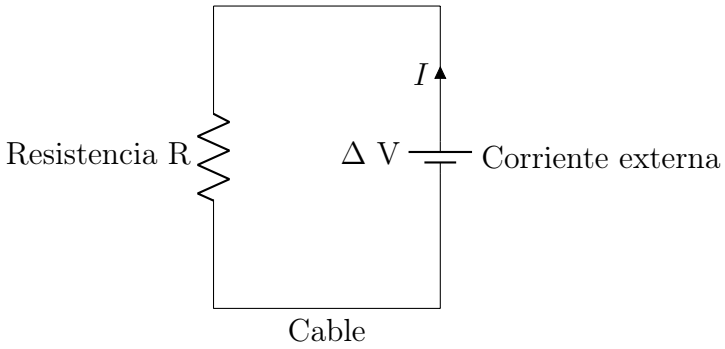


Figura 2.2: Representación de la neurona como circuito

La relación con V se da por la equivalencia de la carga eléctrica generada por la diferencia en la concentración de iones con la definición del potencial eléctrico como el trabajo generado por la fuerza eléctrica de una carga (Halliday et al., 2013). En síntesis la mayor concentración de los iones en una sección de la neurona termina por formar una fuerza electrica, interpretada como potencial eléctrico V, es decir voltaje.

Otra característica presente en las neuronas es el gradiente eléctrico. Este gradiente eléctrico produce una fuerza, que causa a un ión difundirse con una velocidad constante.

La fuerza de los iones está dada por,

$$\overrightarrow{F} = f \cdot \overrightarrow{Vd}. \tag{2.8}$$

Donde:

- f es el coeficiente de fricción
- $\overrightarrow{Vd}$ es la velocidad de difusión
- $\overrightarrow{F}$ es la fuerza

Y el coeficiente de fricción está dado por la relación Einstein-Smoluchowski:

$$f = k\frac{T}{D}$$

Con:

- k la constante de Boltzmann
- T es la temperatura absoluta del fluido

Despejando la velocidad de difusión de la ecuación 2.8, y considerando la definición del vector fuerza como $\vec{F} = q\vec{E}$, con q una carga de prueba y $\vec{E}$ el vector de campo eléctrico, se obtiene

$$\overrightarrow{Vd} = \frac{D}{kT}q\overrightarrow{E}$$

Se pueden establecer algunas relaciones entre las variables, tales como proporcionalidad entre corriente, la velocidad de difusión y el área de las secciones conductoras:

$$I \propto \overrightarrow{Vd}A \quad \text{donde } \propto \text{ significa } \textbf{es proporcional}$$

Y una relación entre la corriente, el área y el campo eléctrico,

$$I \propto EA = \frac{\Delta V}{L}A$$

Con:

- ΔV es el voltaje

- L es la distancia
- A el área de las secciones conductoras

Luego, la corriente es igual a:

$$I = \left(\frac{1}{\rho}\right)\frac{\Delta V}{L}A$$

Donde $\frac{1}{\rho}$ es la resistividad.

Siendo reescrito con base en la ley de Ohm:

$$I = \left(\frac{A}{\rho L}\right)\Delta V \tag{2.9}$$

Entonces sustituyendo la ecuación 2.9 en 2.7 se tiene que:

$$R = \frac{\rho L}{A}$$

Volviendo al contexto neuronal, la resistividad de la salina de los mamíferos es de $\rho \approx 60\Omega$ dificultando la transmisión energética.

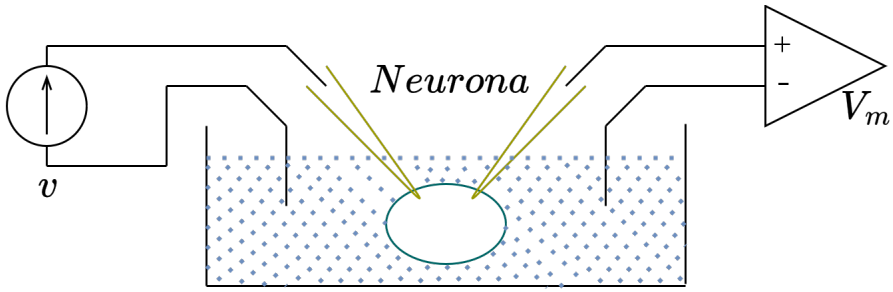


Figura 2.3: El ensamble de la neurona en un medio acuoso, se le inyecta un voltaje por medio de un electrodo y se mide la corriente del medio intra y extracelular con un voltímetro

El cuerpo humano tiene un medio intra y extracelular que regula el intercambio iónico, este intercambio iónico es interpretable como una corriente eléctrica que se transmite por el medio acuoso de la neurona, que de manera análoga ocupa el lugar de los cables en nuestro modelo, ver figura 2.3.

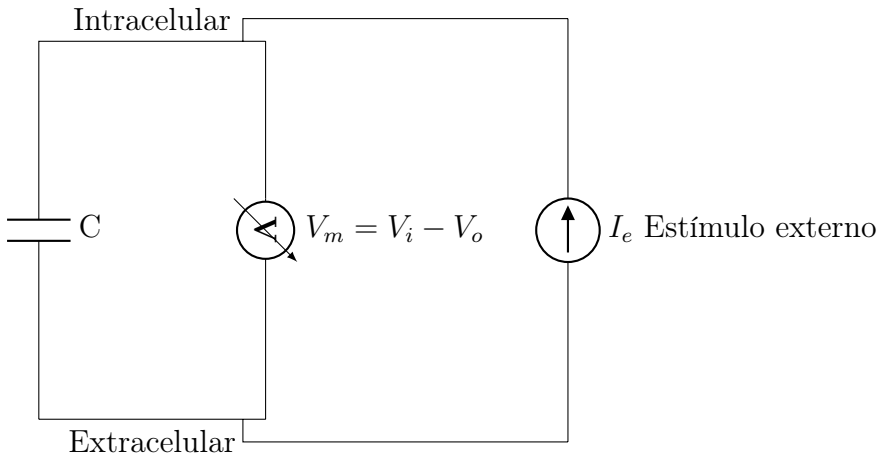


Figura 2.4: El segundo circuito equivalente es hace referencia al ensamble visto anteriormente con la neurona siendo "afectada" por una corriente externa, esta neurona no cuenta con agujeros por lo que se elimina la resistencia del circuito

Lo más importante es que el comportamiento que se había descrito fisiológicamente, ahora es incorporado al modelo y se interpreta mediante ecuaciones como la primera ley de Fick, que describe la difusión de los iones o como este intercambio puede ser expresado en términos de las leyes de Ohm.

De manera análoga al procedimiento de Alan Hodgkin y Andrew Huxley, solo que tomando una "neurona" en lugar de un axón, se sumerge la neurona en un medio salino simulando el líquido intra y extracelular, con una corriente inyectada dentro y fuera de la neurona con dos electrodos y otros dos electrodos para realizar las mediciones.

El propósito de este ensamble es abstraer el comportamiento de las neuronas durante la comunicación, ahora bien, el espacio intra y extracelular están separados por la membrana, la cual tiene un grosor de 23 Angstroms de grueso, que con el conductor, el medio acuoso, y todo separado por una membrana a manera de aislante, forma un capacitor.

Ahora, supongasé que el esquema no cuenta con la resistencia, esta se retiró momentáneamente con respecto al último, pero se rescatará un poco más adelante con su interpretación fisiológica.

Por otra parte, la diferencia de iones en cada lado de la membrana genera una

diferencia de carga y voltaje, ver figura 2.5.

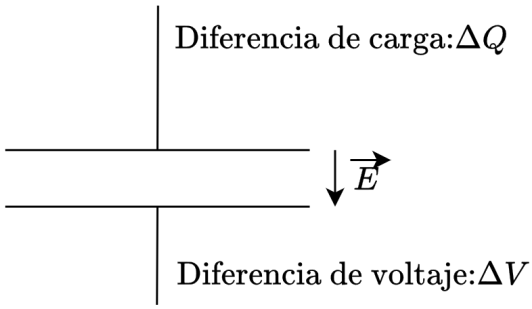


Figura 2.5: La diferencia de voltaje dentro y fuera de la neurona con la membrana sirviendo como aislante da origen a la representación de la membrana como un capacitor

Esta diferencia de carga es proporcional al cambio en el voltaje por la constante de capacitancia:

$$\Delta Q = C \cdot \Delta V \tag{2.10}$$

la capacitancia es proporcional al área de las superficies e inversamente proporcional a la distancia entre estas.

Para medir la corriente del capacitor es necesario observar como se comporta el desbalance presente en las cargas durante el tiempo, es decir:

$$I_c(t) = \frac{\Delta Q}{dt}$$

Esta diferencia en las cargas se vio en la ecuación 2.10 por lo que se puede reescribir como:

$$I_c(t) = \frac{\Delta Q}{dt} = C\frac{\Delta V_m}{dt} \tag{2.11}$$

donde V_m es el voltaje de la membrana.

Por otro lado, desde el punto de vista electromagnético, la primera ley de Kirchhoff enuncia que "la suma de las corrientes que entran a un nodo, es igual

a la suma de las corrientes que salen de dicho nodo" (Halliday et al., 2013) entonces, la corriente procedente del electrodo (I_e) se igualada a la de capacitancia (I_c)

$$I_e = I_c \tag{2.12}$$

por lo que igualando con la ecuación 2.11:

$$I_e(t) = C\frac{\Delta Vm}{\Delta t} \tag{2.13}$$

Integrando esta ecuación diferencial sobre el tiempo iniciando con un voltaje inicial en el tiempo 0.

$$V_m(t) = V_0 + \frac{1}{C}\int_0^t I_e(\tau)d\tau$$

La parte integral de la derecha es la corriente a través del tiempo por la ecuación 2.10

$$\int_0^t I_e(\tau)d\tau = \Delta Q$$

Entonces la diferencia de voltaje es

$$\Delta V_m = \frac{1}{C}\Delta Q$$

Para ejemplificar esto se observa lo que sucede con la función $V_m(t)$ cuando se inyecta una corriente constante a lo largo del tiempo.

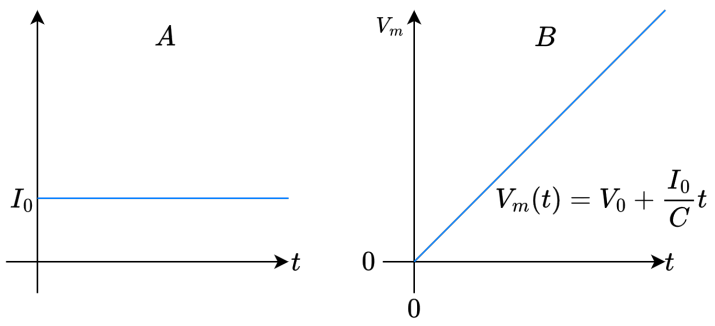


Figura 2.6: Con una corriente constante (A) el voltaje de la membrana aumenta de manera lineal (B)

En la figura 2.6 se muestra el comportamiento de la corriente, I_0, contra el tiempo. Este comportamiento es lineal constante. Por otro lado, se muestra V_m contra el tiempo. Esta función incrementa linealmente a través del tiempo.

Por otro lado, al inyectar la corriente durante un tiempo τ se tiene que el modelo está con un voltaje inicial hasta que llega el estímulo, posteriormente, aumenta hasta que dicho estímulo desaparece, ver figura 2.7

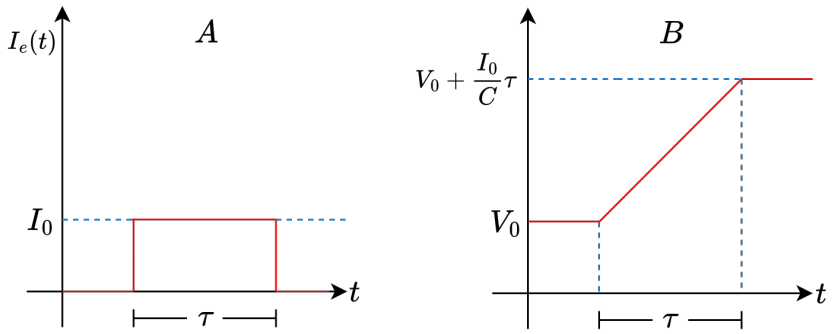


Figura 2.7: Con una corriente temporal (A) y un voltaje inicial diferente a 0, la neurona comienza a aumentar su voltaje hasta dejar de recibir el impulso (B)

Otro rasgo de la neurona son los canales iónicos que regulan la corriente por lo que tienen el papel de las resistencias:

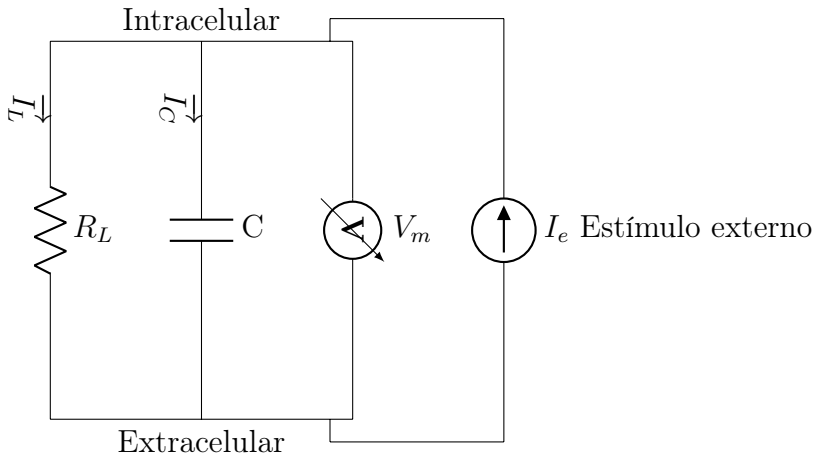


Figura 2.8: Este tercer circuito cuenta con el voltaje externo que se inyecta a las neuronas, los canales iónicos, la membrana y el líquido intra y extracelular

De la misma forma como en la ecuación 2.12, las corrientes de entrada y salida se igualan usando la ecuación 2.11:

$$I_L + C\frac{dV}{dt} = I_e \tag{2.14}$$

Por convención, en neurociencia la corriente que sale de la neurona es considerada positiva.

Con I_L como la corriente iónica que se forma al cruzar los iones por un canal iónico, se puede formular como la ley de Ohm:

$$I_L = \frac{V_m}{R_L}$$

Retomando R_L como la resistencia del canal iónico y sustituyendo en la ecuación 2.14:

$$\frac{V_m}{R_L} + C\frac{dV}{dt} = I_e$$

Multiplicando toda la expresión por R_L

$$V_m + R_L C\frac{dV_m}{dt} = R_L I_e \tag{2.15}$$

Ahora se inyecta una corriente constante dejando la diferencia del voltaje de la membrana igual a cero:

$$\frac{dV_m}{dt} = 0$$

Y se define $V_\infty = R_L I_e$ como el voltaje en el estado estable.

Donde V_∞ es el estado que se obtiene al inyectar la misma corriente durante un periodo indeterminado.

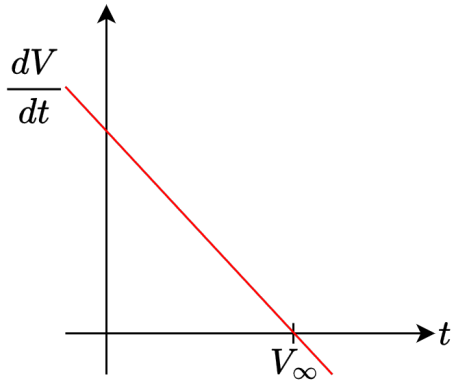


Figura 2.9: La diferencia de voltaje con respecto al tiempo llega a 0 en V_∞ por definición

Volviendo a la corriente de la membrana como en la ecuación 2.15:

$$V_m + \tau \frac{dV_m}{dt} = V_\infty$$

Donde $\tau = R_L C$, convirtiéndose en una ecuación diferencial de orden lineal despejando la diferencial:

$$\frac{dV}{dt} = -\frac{1}{t}(V - V_\infty) \tag{2.16}$$

Por lo que dependiendo de la diferencia del voltaje se tiene la representación que se muestra en la figura 2.9.

Comenzando con un V_0 inicial se observa como el voltaje disminuye a través del tiempo, tendiendo V_m disminuyendo $\frac{1}{e}$ cada periodo de tiempo τ:

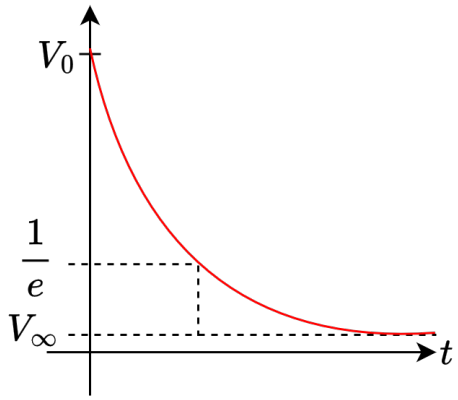


Figura 2.10: Y se ve un decremento del voltaje de la membrana en un tiempo $\frac{1}{e}$ hasta establecerse en V_∞

Cuando la corriente inyectada es constante la solución a la ecuación 2.16 es:

$$V(t) - V_\infty = (V_0 - V_\infty)e^{-\frac{t}{\tau}} \tag{2.17}$$

Ahora bien, se inyecta una corriente I_0 durante un periodo de tiempo (ver figura 2.11)

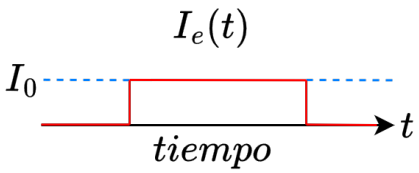


Figura 2.11: Corriente inyectada durante un periodo de tiempo

Por lo que $V_\infty(t)$ al ser por definición es proporcional a I_e aumenta a $R_L I_0$ por ese periodo de tiempo (ver figuras 2.12 y 2.13).

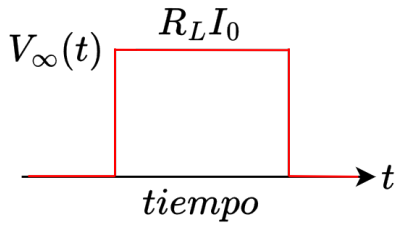


Figura 2.12: Por definición de V_{inf}, esta se define como el producto de la resistencia por la corriente

Así el voltaje va en aumento hasta llegar a igualar el estado estable y al desaparecer esta corriente vuelve al estado de reposo disminuyendo como en A en un tiempo τ:

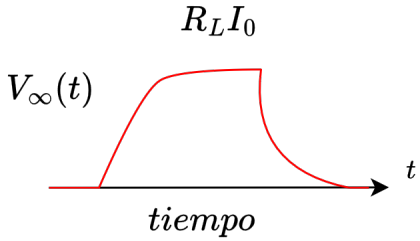


Figura 2.13: Y tras llegar a su punto más alto disminuye como una función logarítmica

Este es el modelo RC que surge de la resistencia y capacitancia actuando como un filtro.

Se observa que es necesario un impulso de una baja frecuencia y en casos con frecuencias altas y poco tiempo de estímulo el voltaje de la membrana no llega a igualar al inyectado.

Este tipo de resultados muestran la importancia de la ecuación 2.16.

$$\begin{aligned} V + \tau \frac{dV}{dt} &= V_\infty \\ V(t) &= V_\infty + (V_0 - V_\infty) e^{-\frac{t}{\tau}} \end{aligned}$$

Fundamental en física, química, biología y en nuestro caso, neurociencia.

Cuando se tienen circuitos con diferentes canales iónicos se interpretan como circuitos con resistencias y capacitores en paralelo respondiendo a las leyes de Kirchhoff (ver figura 2.14). Esta serie de circuitos modela el sistema nervioso humano.

Para aproximarnos a algo más real con un mayor número de canales iónicos se reescribe i en términos de conductancia:

$$I_L = G_L V_m$$

Y la corriente se puede describir de manera individual tomando g_L como conductancia específica y el área de la membrana:

$$I_L = G_L V_m = A g_L V_m$$

E igualmente la capacitancia depende del punto de la membrana a analizar, contando con su propia capacitancia C_m, esto equivale a los diferentes canales con los que cuenta la célula.

Y el área de la membrana es:

$$A = 4\pi r^2$$

Volviendo a $\tau = R_L C$ sustituyendo hasta obtener una expresión que únicamente involucre a la conductividad y capacitancia particular.

$$\begin{aligned} \tau_1 &= R_L C = \frac{C}{G_L} \\ &= \frac{c_m A}{g_L A} = \frac{c_m}{g_L} \end{aligned}$$

Haciendo que τ varíe dependiendo del canal iónico y sección de la neurona.

El modelo RC refleja el comportamiento del potencial de acción en un nivel básico, pero a diferencia de una neurona real, el modelo, al perder la corriente externa termina con un voltaje de 0, lo que para una neurona significa la muerte.

Para que el modelo adquiera este comportamiento será necesario añadirle una batería que al igual que el potencial de membrana, mantenga el voltaje de la neurona.

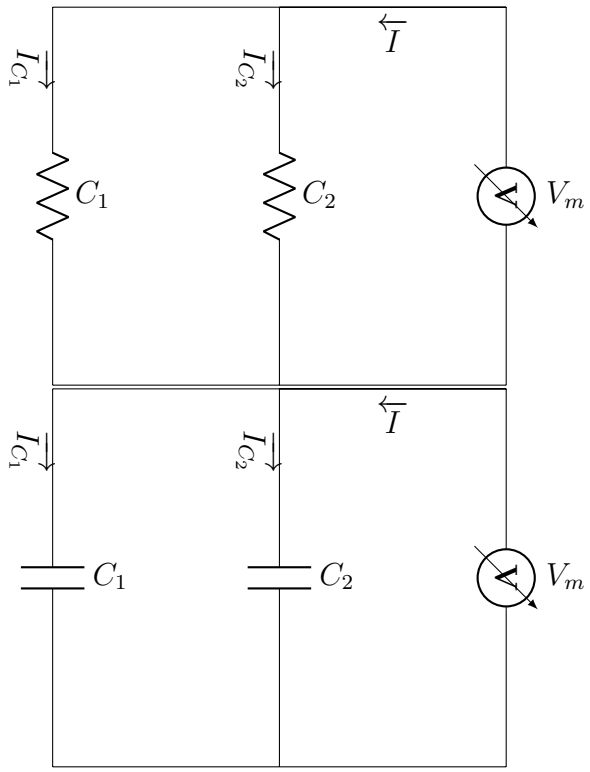


Figura 2.14: Circuito con resistencias y capacitancias en paralelo

Estas baterías surgen de los gradientes químicos y eléctricos presentes en la neurona, donde la ecuación de Boltzmann describe la relación entre la probabilidad de una partícula de cambiar de un estado a otro en un equilibrio térmico, en este modelo en particular estos estados son dentro y fuera de la neurona.

$$\frac{P_i}{P_o} = \exp\Big(-\frac{\mathrm{U_i} - \mathrm{U_o}}{\mathrm{kT}}\Big) = \exp\big(-\frac{\mathrm{q(V_i} - \mathrm{V_o)}}{\mathrm{kT}}\big)$$

Donde:

- kT Es la energía térmica

- $U = qB$ Es el potencial eléctrico

De esta ecuación se despeja la diferencia de voltajes de entrada y salida.

$$\begin{aligned} V_i - V_o &= -\frac{kT}{q} \ln\left(\frac{P_i}{P_o}\right) \\ V - V' &= -\frac{kT}{q} \ln\left(\frac{P_i}{P_o}\right) \end{aligned}$$

El valor de la constante es:

$$\frac{kT}{q} = 25 \text{ mV}$$

La diferencia de voltajes es algo que ya se ha tratado anteriormente y la probabilidad está dada por el ion en cuestión que nuestro canal iónico deje pasar, por lo que queda la siguiente expresión.

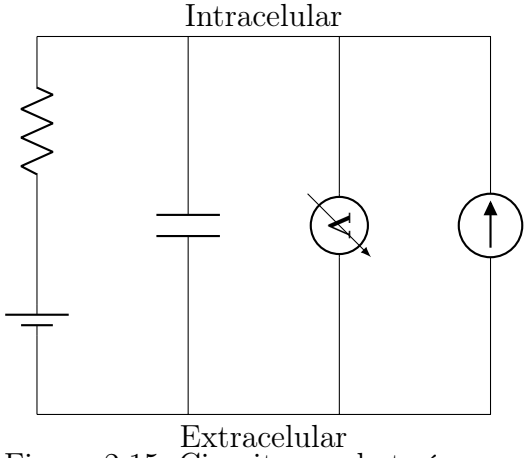


Figura 2.15: Circuito con baterías

$$\Delta V = V_i - V_o = 25\text{mV} \ln\left(\frac{[k]_o}{[k]_i}\right).$$

La elección del potasio está dada por las grandes diferencias de la concentración en el medio intra y extracelular, con el apoyo de la ecuación de Nernst se calcula el estado de reposo de este ión.

La figura 2.15 muestra un circuito con baterias, donde la batería es necesaria para que el modelo de la neurona conserve energía, su equivalente fisiológica es el gradiente químico y eléctrico de la neurona.

Con una concentración aproximada de 20 potasios en el citoplasma por cada potasio en el interior de la neurona se obtiene una diferencia de voltaje de :

$$\begin{aligned} \Delta V &= 25 \text{ mV} \ln\left(\frac{1}{20}\right) \\ &= -75 \text{ mV} \end{aligned}$$

Valor que da origen al voltaje visto en el subtema de electrofisiología.

Esta proporción varía entre los iones y se calcula como el caso del potasio variando

	Intracelular	Extracelular	Nernst
K^+	400	20	-75
Na^+	50	440	+55
Cl^-	52	560	-60
Ca^{++}	10^{-4}	2	+124

Cuadro 2.1: Tabla con los potenciales de equilibrio de Nernst con base en la diferencia de concentraciones entre el medio intra y extra célular

las concentraciones y en el caso del calcio $\frac{kT}{q}$ para las cargas.

Los resultados de la tabla 2.1 y los valores obtenidos por Hodgkin y Huxley en sus experimentos son congruentes, en el caso particular del potasio, el valor de -70mV tiene un error porcentual del 7 %.

En la siguiente sección se discute el modelo Hodgkin-Huxley (HH), los conceptos más relevantes y las matemáticas detrás del modelo HH.

2.2. Modelo Hodgkin-Huxley

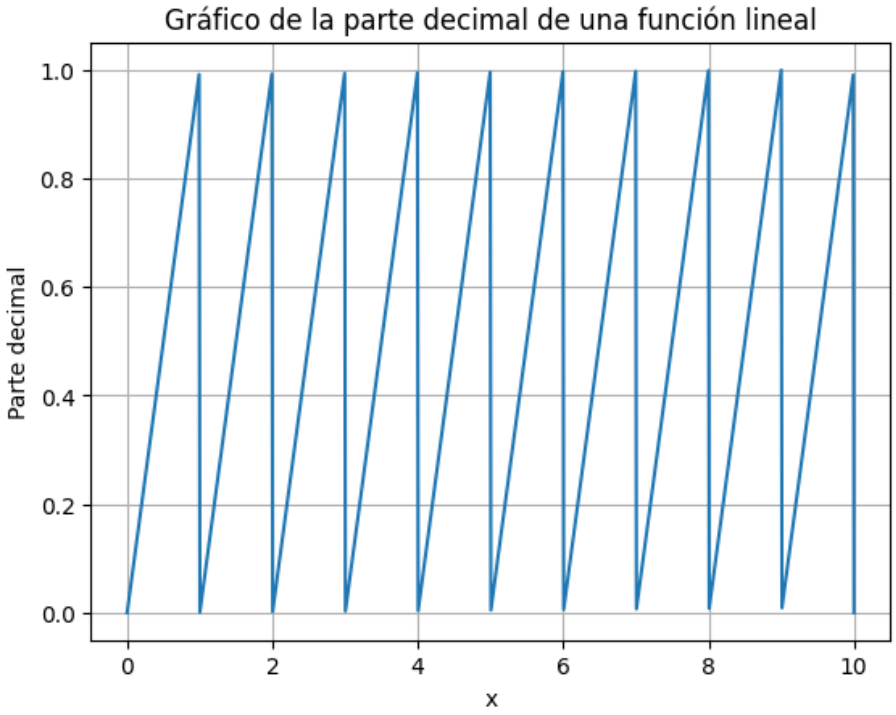


Figura 2.16: El generador de picos es un límite (*threshold*) que se añade a la función y permite obtener una gráfica periódica

Para conseguir este comportamiento, se añade al modelo un generador de picos y como se ha manejado hasta el momento, el estímulo es representado como la corriente que alimenta a la neurona durante cierto periodo de tiempo.

Cuando se retiran los canales iónicos de la ecuación y por consiguiente a las baterías, se observa como el potencial de membrana aumenta de manera lineal hasta encontrarse con el umbral del generador de picos, dando origen a una forma similar a la parte decimal de una función lineal, como en la figura 2.16.

La siguiente parte del modelo está fundamentada en cómo es que una neurona al llegar a un umbral comienza a tener picos de voltaje repetidamente a lo largo del tiempo, dependiendo de la entrada de corriente en la neurona. El símil de este comportamiento se da cuando la neurona, abre y cierra canales iónicos al llegar a cierta cantidad de iones en el espacio intracelular, como se vio en el capítulo 1.

El radio de activación está definido como 1 sobre el tiempo transcurrido entre un pico y otro:

$$f.r. = \frac{1}{\Delta t} \quad \text{Dónde } f.r. \text{ es el radio de activación}$$

Considerese un caso con un condensador/capacitor; se retoma la ecuación 2.11 de donde se despeja la diferencia en el tiempo

$$\Delta t \;\; = \;\; C\frac{\Delta V}{I_E}$$

Este es el modelo únicamente con las capacitancias, pero cuando se añaden los canales ionicos al modelo se obtiene algo similar a la mezcla del modelo RC con la gráfica con picos del modelo anterior (ver figura 2.17).

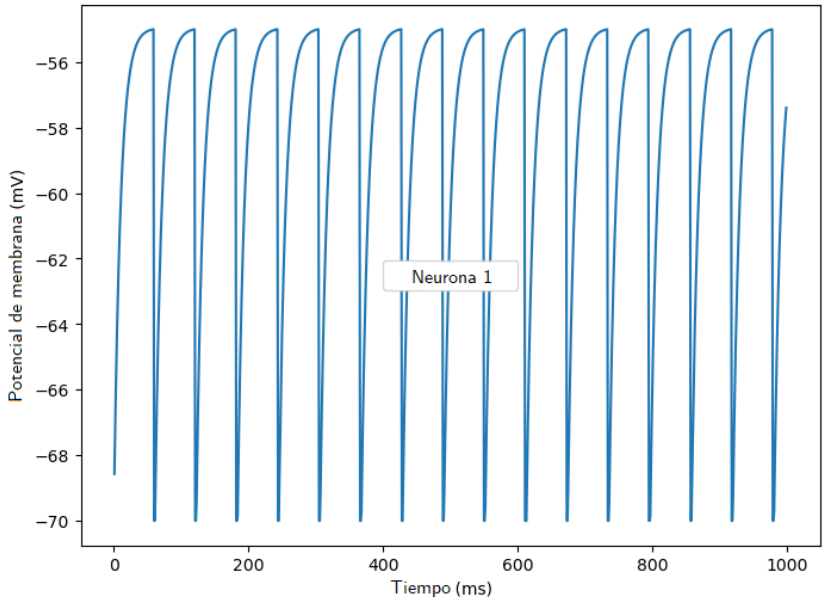


Figura 2.17: El modelo *integrate and fire* muestra un patrón repetido por la interacción con el *threshold*

Con este nuevo modelo vuelve a ser necesario calcular el radio de activación, pero en esta ocasión se requiere utilizar la ecuación 2.17, reemplazando $V(t)$ por V_{th}, t por Δt y V_0 por V_{res}:

$$V_{th} - V_\infty = (V_{res} - V_\infty)e^{-\frac{\Delta t}{\tau}}$$

De donde se despeja Δt, resultando:

$$\Delta t \quad = \quad -\tau \ln\left(\frac{V_\infty - V_{th}}{V_\infty - V_{res}}\right)$$

Dejando el radio de activación como:

$$f.r. = \Delta t^{-1} = \left[\tau \ln\left(\frac{V_\infty - V_{th}}{V_\infty - V_{res}}\right)\right]^{-1}$$

Cuando la corriente que ingresa en el ensamble es muy grande, aumenta de manera proporcional V_{inf} llegando a ser mucho mayor que V_{th} y V_{res}, por lo cual se puede reescribir el radio de activación como:

$$f.r. = \frac{1}{C\Delta V}(I_e - I_{tn}) \tag{2.18}$$

con:

$$I_{tn} = G_L(V_{tn} - E_L)$$

Comportándose similar a una función lineal con una corriente I_e inyectada.

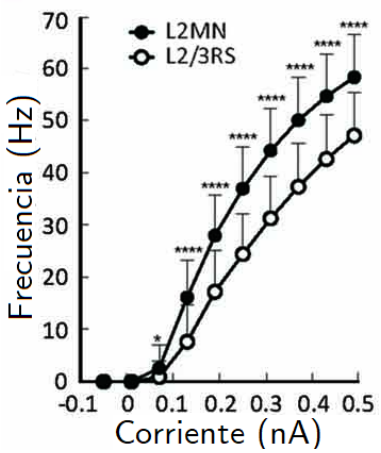


Figura 2.18: Resultados experimentales con la corteza temporal de ratones muestran el comportamiento del diagrama (circulos blancos). *Nota.* Adaptado de *Corteza temporal de ratones*, por Luo et al., 2017, Frontiers (https://www.frontiersin.org/article/10.3389/fnana.2017.00115)

Entre más corriente inyectada, mayor resulta el radio de activación, esto corresponde con una neurona real que como se mencionó en el capítulo de fisiología, la intensidad del impulso externo no es representado por la altura del pico durante el potencial de acción, esta intensidad se mide en la cantidad de picos que tenga la neurona.

Todo el desarrollo visto hasta el momento en este capítulo 2 lleva a las conclusiones obtenidas hace medio siglo por Sir Alan Hodgkin y Sir Andrew Huxley. El ensamblaje utilizado hasta el momento con una neurona sumergida en solución salina conectada a dos electrodos es un equivalente al experimento con el axón gigante de calamar que usaron estos científicos en su momento.

Una de las características más destacables del axón gigante de calamar es su distintivo tamaño, llegando aproximadamente a los 0.5 milímetros, en perspectiva cualquier otro axón es por lo menos 100 veces de menor tamaño, este tamaño permitió llevar a cabo las mediciones con relativa facilidad.

Hodgkin y Huxley aplican las equivalencias vistas hasta el momento, dando origen al modelo Hodgkin-Huxley, el poder real del modelo se hizo evidente cuando la integración numérica de esas ecuaciones diferenciales reproduce de manera bastante fiel todas las propiedades biofísicas del potencial de acción de una neurona, haciéndolos acreedores del premio de Fisiología y Medicina en 1963.

Con el modelo HH se busca obtener un comportamiento equivalente al potencial de acción de una neurona.

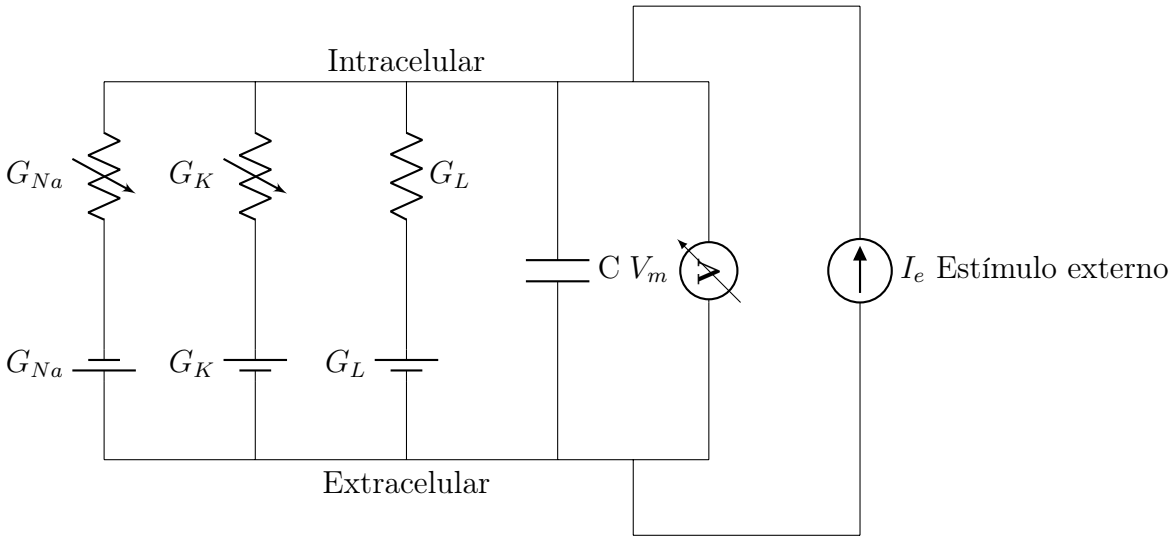


Figura 2.19: Esta es la configuración del modelo neuronal propuesta por Hodgkin-Huxley donde se puede simular el comportamiento del potencial de acción de una neurona

La ecuación del modelo HH es:

$$I_m(t) + C\frac{dV(t)}{dt} = I_e(t) \tag{2.19}$$

Con los valores trabajados hasta el momento, la corriente de la membrana se descompone en sus partes con la primera ley de Kirchkoff (Halliday et al., 2013).

Si

$$I_m = I_{Na} + I_K + I_L$$

Con

$$\begin{aligned} I_{Na} &= G_{Na}(V,t)(V - E_{Na}) \\ I_K &= G_K(V,t)(V - E_K) \\ I_L &= G_L(V - E_L) \end{aligned} \tag{2.20}$$

Será útil recordar que la tensión eléctrica para los diferentes iones fue obtenida al final del capitulo anterior al calcular el potencial de Nernst y los valores resultantes fueron los siguientes:

$$\begin{aligned} E_{Na} &= +55mV \\ E_K &= -75mV \\ E_L &= -50mV \end{aligned} \tag{2.21}$$

La conductancia es dependiente del voltaje, por lo que para el cálculo del potencial de acción de la neurona se toma el voltaje V_m de la misma y se realiza el siguiente algoritmo:

Con V_m:

- Se calculan los parámetros dependientes del voltaje usando V_m
- Se calculan las conductancias dependientes del voltaje
- Se calculan las corrientes de sodio y potasio de las conductancias.
- Se calcula la corriente total de la membrana.
- Se calculan V_∞ y τ

- Y finalmente se computa el potencial de membrana con:

$$V_m + \tau \frac{dV_m}{dt} = V_\infty$$

La pinza de voltaje es un dispositivo que mantiene el potencial de membrana en un voltaje V_c predefinido y mide la corriente necesaria para mantener ese determinado voltaje.

El componente clave es un amplificador operacional definido por la ecuación básica:

$$V_o = GA(V_+ - V_-)$$

Siendo GA la ganancia como un valor oscilante entre 10^5 a 10^6

Con la pinza de voltaje conectada a la neurona:

Si $V_m < V_c$ entonces:

- $V_o >> 0$
- Se introduce corriente dentro de la neurona
- Aumenta el potencial de membrana

$V_o >> 0$ Se introduce corriente dentro de la neurona aumenta el potencial de membrana Si $V_m > V_c$ entonces:

- $V_o << 0$
- Se extrae corriente de la membrana
- Se reduce el potencial de membrana

Por lo que a mayor ganancia $V_m \simeq V_c$.

Gracias a este dispositivo es posible obtener las mediciones suficientes para aproximar la conductancia con diferentes corrientes y dependencias de tiempo.

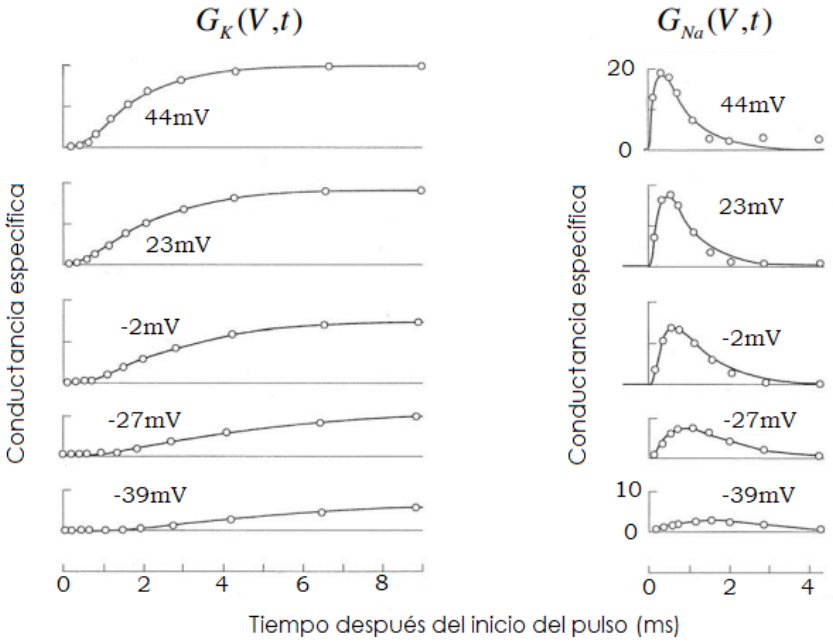


Figura 2.20: Tras las medidas experimentales se puede observar el comportamiento de la conductancia específica para cada ion. *Nota.* Adaptado de *Ionic Conductance*, por Fee y Zysman, 2018, MIT OCW(https://ocw.mit.edu/courses/9-40-introduction-to-neural-computation-spring-2018/resources/mit9_40s18_lec01/)

Esto es obtenido tras sustituir el sodio en el axón con cloruro de colina para detener la interacción única del potasio y tomando en cuenta todos los canales iónicos a la vez.

Otra manera de medir la conductancia total que pasa a través de la membrana está dada por le número total de canales permebles a sus correspondientes iónicos, en este caso compete definir:

- P_K = Probabilidad de que un canal de potasio este abierto.
- N_K = Número total de canales ionicos
- $\hat{g}_K$ = conductancia unitaria

Así se puede definir $P_K N_K$ como el número de canales iónicos abiertos.

Por lo que la conductancia total de K es:

$$G_K = P_K(V,t)N_K\hat{g}_K$$

Con los canales iónicos formados por 4 subunidades idénticas, a su vez, estas subunidades están en un estado abierto o cerrado con probabilidad n y las 4 subunidades deben estar abiertas para que el canal sea permeable, por lo que:

$$P_K = n^4 \tag{2.22}$$

Es la probabilidad de que las 4 subunidades están abiertas por lo que la conductancia es:

$$G_K = \overline{G}_K n^4$$

Con $\overline{G_K}$ como la conductancia máxima, de manera consecuente la corriente de potasio se puede escribir como:

$$I_K = \overline{G}_K n^4 (V - E_K) \tag{2.23}$$

Para ver como la diferencia termal pasa de un estado cerrado a uno abierto, se tienen estados de transición donde α_n es la probabilidad por unidad de tiempo de pasar de un estado cerrado a uno abierto y β_n para el cambio contrario, estos cambios se dan con la unidad de tiempo $\frac{1}{s}$.

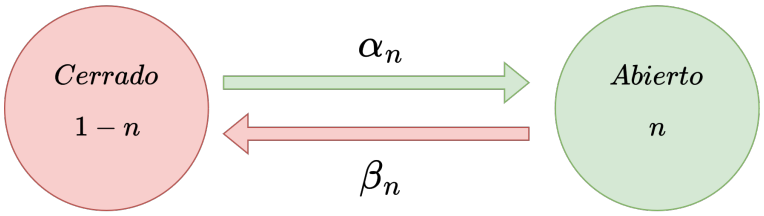


Figura 2.21: Esta es la configuración del modelo neuronal propuesta por Hodgkin-Huxley donde se puede simular el comportamiento del potencial de acción de una neurona

El cambio de las unidades abiertas por unidad de tiempo es:

$$\frac{dn}{dt} = (1-n)\alpha_n - n\beta_n \tag{2.24}$$

Esta ecuación describe el cambio de subunidades cerradas a abiertas, a su vez, desarrollando el lado izquierdo se puede llegar a:

$$\frac{1}{(\alpha_n+\beta_n)}\frac{dn}{dt} = \frac{\alpha_n}{\alpha_n+\beta_n} - n \tag{2.25}$$

De donde resulta útil definir n_∞ en el estado de reposo como:

$$n_\infty = \frac{\alpha_n}{(\alpha_n+\beta_n)} \tag{2.26}$$

Y τ_n como una constante de tiempo, definida como:

$$\tau_n = \frac{1}{(\alpha_n+\beta_n)} \tag{2.27}$$

Reescribiendo la ecuación 2.25 como:

$$\tau_n\frac{dn}{dt} = n_\infty - n \tag{2.28}$$

Con esta ecuación se puede integrar n, siendo posible calcular la conductancia.

De la ecuación 2.22 y la ecuación 2.23 de la corriente del potasio, solo falta obtener

las ecuaciones de los canales de sodio, al igual que con el potasio hay subunidades, que serán representadas con la letra m quedando escritas las ecuaciones 2.27 y 2.26 como:

$$\tau_n \frac{dm}{dt} = m_\infty - m \qquad (2.29) \qquad\qquad m_\infty = \frac{\alpha_m}{(\alpha_m + \beta_m)} \qquad (2.30)$$

Los coeficientes de transición son calculados de la siguiente manera:

$$\alpha_n(V) = \frac{1}{1 - e^{-0,1(V+55)}} \qquad (2.31)$$

$$\beta_n(V) = 0{,}125\ e^{-0,0125(V+65)} \qquad (2.32)$$

Donde:

- V está dado en mV.
- $\alpha_n(V), \beta_n(V)$ estan en ms^{-1}

A diferencia de los canales de potasio aquí solo son 3 subunidades para permitir el paso pero con una puerta de inactivación quedando la probabilidad de que el canal este abierto como:

$$P_{Na} = m^3 h$$

Dependiente de un canal de inactivación presente en el modelo y que se corresponde con:

$$\tau_h \frac{dh}{dt} = h_\infty - h$$

Los demás calculos son analogos al potasio por lo que finalmente la corriente del sodio queda de la siguiente manera:

$$I_{Na} = \overline{G}_{Na} m^3 h (V - E_{Na}) \qquad (2.33)$$

De esta manera obteniendo las partes necesarias para calcular el potencial de membrana de una neurona durante la comunicación, por lo que aprovechando las facilidades que brinda *Python* para estos cálculos iterativos se digitalizará directamente.

2.3. Digitalización de los modelos neuronales

Hodgkin y Huxley solo contaban con calculadoras mecánicas para los cálculos por lo que todo esto se tenía que hacer a mano, con la tecnología de la actualidad se puede facilitar este proceso por medio de sistemas de CAS (*Computer Algebra System*) como *MathLab* o apoyados por lenguajes de programación como *Python*.

Por medio de una clase en *Python* se puede describir de manera general el comportamiento del modelo HH. En la siguiente subsección se simula el comportamiento de una neurona. Esta implementación está inspirada en el *Open Source Brain Project* realizado por Padraig Gleeson (Gleeson, 2015).

Esta primera sección de código se llevó a cabo en una instalación de WLS 2 para Windows 10, por lo cual se trata de un subsistema de *Linux*, al cual se le instaló anaconda y *Jupyter Notebook* para los scripts.

Para la ejecución del proyecto se importa la librería *SciPy* [1] en su versión 1.8.0 que incluye una serie de métodos y procedimientos para trabajar con las ecuaciones necesarias, en particular, la función `odeint` resuelve las ecuaciones diferenciales del modelo. Para graficar las ecuaciones se utiliza *MatPlotLib*[2] en su versión 3.4.2.

[1] SciPy https://scipy.org
[2] MatPlotLib https://matplotlib.org

En el programa, las variables biológicas son:

- $C_m = 1 \frac{\mu F}{\text{cm}^2}$ Capacitancia de la membrana
- $g_{Na} = 120 \frac{\text{mS}}{\text{cm}^2}$ Conductancia máxima del sodio(Na)
- $g_K = 36 \frac{\text{mS}}{\text{cm}^2}$ Conductancia máxima del potasio(K)
- $g_L = 0{,}3$ Conductancia máxima del agujero
- $E_{Na} = 50$ mV Potencial de Nernst del sodio(Na)
- $E_K = -77$ mV Potencial de Nernst del potasio(K)
- $E_L = -54{,}387$ mV Potencial de Nernst del agujero

Para medir las dimensiones de la capacitancia se utilizan F, unidad estándar del Sistema Internacional de Unidades para cuantificar la capacitancia (Ramón Romero et al., 2015). A su vez, se mide la conductancia de cada canal en S, los siemens son una unidad de conductancia en el sistema internacional equivalente de manera inversa al ohmio, por lo que es la conductancia permitida en cada canal por centímetro cuadrado. Finalmente el potencial eléctrico de cada canal iónico se mide en Volts (V).

La codificación de las fórmulas es bastante literal con base en lo visto anteriormente en el documento, partiendo de la definición de los α para cada canal, esto se puede ver referenciado en las ecuaciones 2.31 y 2.32, donde se codifica de la siguiente manera, utilizando programación estructurada:

In [1]:
```
def alpha_m(self, V):
    return 0.1*(V+40.0)/(1.0 - sp.exp(-(V+40.0) / 10.0))


def beta_m(self, V):
    return 4.0*sp.exp(-(V+65.0) / 18.0)


def alpha_h(self, V):
    return 0.07*sp.exp(-(V+65.0) / 20.0)


def beta_h(self, V):
    return 1.0/(1.0 + sp.exp(-(V+35.0) / 10.0))


def alpha_n(self, V):
    return 0.01*(V+55.0)/(1.0 - sp.exp(-(V+55.0) / 10.0))


def beta_n(self, V):
    return 0.125*sp.exp(-(V+65) / 80.0)
```

Donde las principales variaciones son por los resultados obtenidos durante la experimentación de Hodgkin y Huxley.

La siguiente sección de la clase hace referencia a la fórmula 2.33 donde se calculan las diferentes corrientes dependientes del voltaje que pasan por cada canal:

In [1]:
```
def I_Na(self, V, m, h):
    return self.g_Na * m**3 * h * (V - self.E_Na)


def I_K(self, V, n):
    return self.g_K  * n**4 * (V - self.E_K)


def I_L(self, V):
    return self.g_L * (V - self.E_L)
```

Esta primera sección es un simple repaso de cómo se codifican las fórmulas y no es diferente a como se podría automatizar la fórmula utilizando una hoja de cálculo, pero a partir de ahora, utilizando *Python* para este tipo de cálculos, creando una función variable para la corriente inyectada, dependiente del tiempo y calculando los valores de las derivadas ecuación (2.24) de manera dinámica.

In [1]:
```
def I_inj(self, t):
    return 10*(t>100) - 10*(t>200) + 35*(t>300)
    - 35*(t>400)


@staticmethod
def dALLdt(X, t, self):
    V, m, h, n = X


    dVdt = (self.I_inj(t) - self.I_Na(V, m, h)
            - self.I_K(V, n) - self.I_L(V)) / self.C_m
    dmdt = self.alpha_m(V)*(1.0-m) - self.beta_m(V)*m
    dhdt = self.alpha_h(V)*(1.0-h) - self.beta_h(V)*h
    dndt = self.alpha_n(V)*(1.0-n) - self.beta_n(V)*n
    return dVdt, dmdt, dhdt, dndt
```

Finalmente, la función principal de la clase las secciones que componen esta función son: La obtención de los resultados de la integración para el voltaje y cada canal. La definición de los arreglos que contendrán la información de los voltajes y canales. Obtención de las corrientes. Graficación de los elementos.

Los valores de la integración son directos, utilizando la función `odeint` se envían los diferentes parámetros de nuestra neurona y consiguiendo un arreglo de valores con los resultados para el voltaje y cada canal:

In [2]:
```
X = odeint(self.dALLdt, [-65, 0.05, 0.6, 0.32],
    self.t, args=(self,))
```

Con la función `odeint` se resuelve un sistema ordinario de ecuaciones diferenciales con los parámetros `self.dALLdt` que retorna las ecuaciones diferenciales y las envía como funciones. [-65, 0.05, 0.6, 0.32] es un arreglo con las condiciones iniciales de los sistemas, es decir el voltaje inicial y las probabilidades de los canales. `self.t` es la variable independiente con la cual se calcula la ecuación, en este caso el tiempo. self estos son argumentos adicionales a la función por lo cual se envía a la ejecución misma.

El segundo paso es una asignación directa de los valores resultantes en los diferentes arreglos para cada valor.

In [3]:
```
V = X[:,0]
m = X[:,1]
h = X[:,2]
n = X[:,3]
ina = self.I_Na(V, m, h)
ik = self.I_K(V, n)
il = self.I_L(V)
```

Finalmente toca una sección un poco más genérica de código donde se conjuntan los resultados, cada sección de código inserta un resultado a la gráfica: Gráfica del potencial de acción:

In [4]:
```
plt.subplot(4,1,1)
plt.title('Hodgkin-Huxley Neuron')
plt.plot(self.t, V, 'k')
plt.ylabel('V (mV)')
```

Corriente que produce cada canal iónico:

In [5]:
```
plt.subplot(4,1,2)
plt.plot(self.t, ina, 'c', label='$I_{Na}$')
plt.plot(self.t, ik, 'y', label='$I_{K}$')
plt.plot(self.t, il, 'm', label='$I_{L}$')
plt.ylabel('Corriente')
plt.legend()
```

Valores de las puertas:

In [6]:
```
plt.subplot(4,1,3)
plt.plot(self.t, m, 'r', label='m')
plt.plot(self.t, h, 'g', label='h')
plt.plot(self.t, n, 'b', label='n')
plt.ylabel('Valor para los canales')
plt.legend()
```

Corriente inyectada:

In [7]:
```
plt.subplot(4,1,4)
i_inj_values = [self.I_inj(t) for t in self.t]
plt.plot(self.t, i_inj_values, 'k')
plt.xlabel('t (ms)')
plt.ylabel('$I_{inj}$ ($\\mu{A}/cm^2$)')
plt.ylim(-1, 40)
```

Y con todo listo, solo resta ejecutar el código:

In [8]:
```
if __name__ == '__main__':
    runner = HodgkinHuxley()
    runner.Main()
```

Con esto se obtiene la siguiente gráfica:

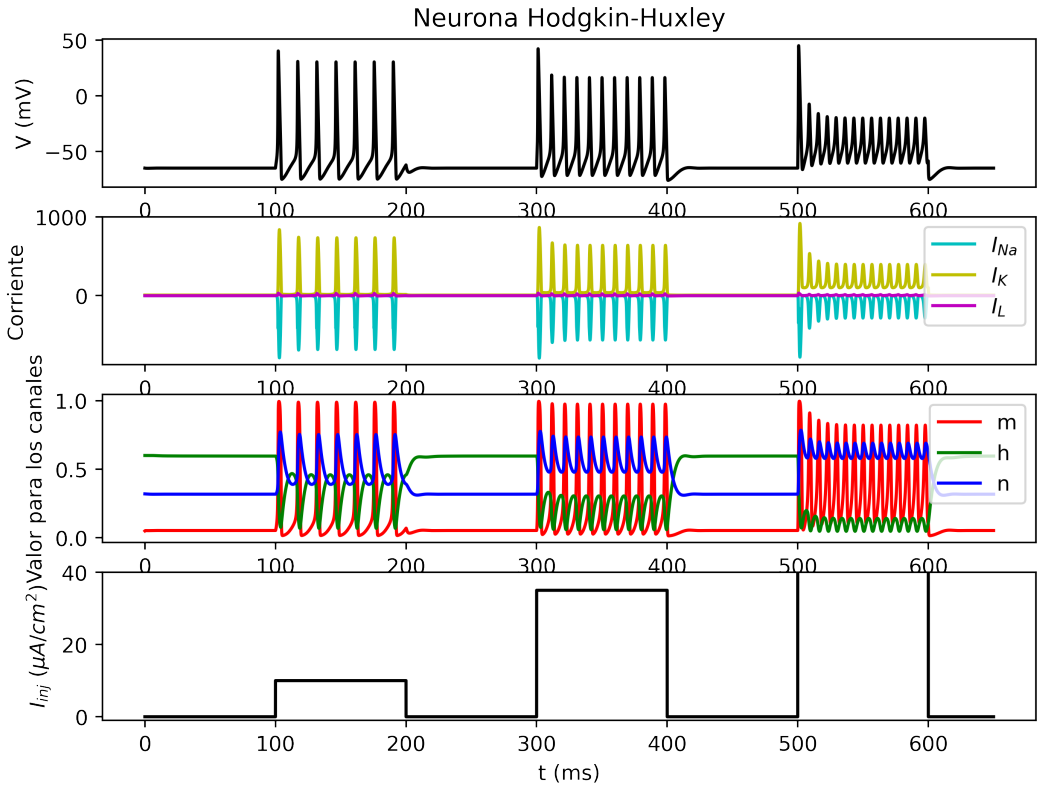


Figura 2.22: En la figura se puede observar en orden de abajo hacia arriba, la corriente inyectada en la neurona, el estado de los canales, la corriente generada y el voltaje de la membrana

Con el contexto necesario se puede observar cómo la neurona se comporta como es de esperarse, las puertas se abren cambiando su valor de 0 a 1 cuando la corriente es inyectada, la corriente varía dependiendo del tipo de ion que está entrando o saliendo de la neurona, y finalmente el voltaje, que sube y baja en picos con la entrada de la corriente y al inyectar una mayor cantidad de corriente manteniendo la amplitud, pero aumentando la frecuencia.

Se ha logrado el objetivo, la neurona, un ente biológico, ahora está convertido en valores almacenados dentro de variables de una clase de *Python*, el resultado de las investigaciones de Hodgkin y Huxley siguen siendo relevantes, convirtiéndose en un enlace o primer contacto con la neurociencia computacional, ahora con una neurona se tiene un primer avance, ahora resta explotar nuestros recursos y el desarrollo para llevar esta neurona al siguiente nivel y poder simular una configuración neuronal.

En la figura se puede observar en orden de abajo hacia arriba, la corriente

inyectada en la neurona, el estado de los canales, la corriente generada y el voltaje de la membrana.

Capítulo 3

Simulación neuronal

Como se ha mencionado en repetidas ocasiones en este documento, el desarrollo de la neurociencia computacional ha sido favorecido por las nuevas tecnologías que han surgido en estos años, ya sea desde la parte experimental con una mejora de los equipos de laboratorio, como desde la parte teórica con su implementación computacional.

Partiendo de la clase programada en *Python* al final del capítulo anterior se puede comenzar a variar los parámetros para poder experimentar cómo es que la neurona se transforma y reacciona de distintas maneras, claro, todo atado al tipo de neurona de la familia de la propuesta por Hodgkin-Huxley, por ejemplo, al aumentar los tiempos de corriente se aumenta y reduce mucho más la frecuencia de los picos de las neuronas.

3.1. La simulación neuronal

Las limitaciones a las que Hodgkin y Huxley se enfrentaron en su momento ahora están resueltas gracias al avance de la computación, la automatización de estas tareas, además del avance mismo de la fisiología neuronal da nacimiento a una nueva serie de herramientas de trabajo que generalizan el conocimiento previo y lo impulsan para seguir creciendo.

La historia actual de la computación ha demostrado que el paradigma abierto,

referente a temas de desarrollo ha impulsado tanto a desarrolladores como a sus herramientas a crecer con un buen ritmo y a su vez eliminando la barrera que significa la privatización de los conocimiento, facilitando a su vez que la gente se incluya en el proyecto.

Ejemplos de esto se pueden encontrar con el sistema operativo *GNU/Linux* o el gestor de versiones *Git* [1], ambos proyectos siempre presentes para la comunidad del desarrollo de software, es de esta ideología que surgen nuevos paradigmas en otras áreas, como la propuesta de *open science* de la Comisión Europea.

Open science es un nuevo paradigma que impulsa la colaboración y la difusión del conocimiento aprovechando las nuevas tecnologías digitales, con este nuevo paradigma cada usuario puede hacer una aportación al proyecto y así crecer la iniciativa más y más, estás ideas están desarrolladas con mayor detalle en el escrito de *Open Innovation*, *Open Science*, *Open to the World*, *a vision for Europe* (Commission, 2016).

A lo largo del documento se hacen presentes los beneficios e inclusive algunas aplicaciones que se llevan a cabo en la actualidad, la relevancia de este tipo de iniciativas se hace presente cuando se conocen proyectos que se apoyan de esta ideología, uno de estos proyectos, con una relevancia importantísima para este documento es la iniciativa NEST.

Proyectos como la iniciativa NEST impulsan constantemente en la labor de facilitar el acceso a la simulación neuronal, pero, ¿Qué es la iniciativa NEST? La iniciativa NEST es un proyecto que inicia en 2001 con una serie de publicaciones sobre neurociencia computacional para ganar relevancia, pero que en 2012 se convierte en una organización basada en miembros sin ánimo de lucro con sede en Suiza.

Con base en esta estructura apoyada en miembros la iniciativa NEST ha llevado a cabo una serie de proyectos y publicaciones que impulsan el desarrollo de la neurociencia computacional, pero el proyecto que es más relevante en este contexto es el simulador NEST.

El simulador NEST es una herramienta desarrollada en un inicio en *C++* pero

[1] Sitio del proyecto de versionamiento Git https://git-scm.com/about

que con el paso del tiempo ha transicionando a una interfaz en *Python* volviendo más accesible su uso pero conservando su dependencia a un compilador de *C++*. Actualmente el simulador NEST se encuentra en su versión 3.3 con una constante actualización en su documentación.

La instalación es variada y dependiente de las necesidades, se destaca la compatibilidad con el sistema operativo Linux, en particular con las distribuciones *Ubuntu*, *Debian* y *NeuroFedora*. Otras alternativas llamativas resultan en instalaciones enfocadas en contenedores como lo puede ser docker.

La instalación realizada para el desarrollo del presente documento es por medio de *WSL 2*, como un subsistema de *Linux* para *Windows* e instalando la distribución de *Ubuntu*, con la distribución instalada se configura el entorno para el trabajo con *Python* ya sea con *Conda* o directamente utilizando el repositorio PPA, todas las instrucciones necesarias para la instalación se encuentran presentes en la documentación del simulador NEST (de Schepper et al., 2022).

Con la instalación en *Conda*, se crea un ambiente que contendrá las dependencias y secciones del kernel requeridas para el uso del simulador NEST, ahora bien la codificación se realizó directamente en un *Jupyter Notebook* por temas de futura compatibilidad con el *Allen Brain Atlas*.

Una manera directa de poder interpretar el cambio de una codificación tradicional (como la vista al final del capítulo anterior y al inicio de este) al uso de un simulador es emulando el mismo tipo de modelos, **Integrate and Fire** y **Hodgkin-Huxley**, que debido a su relevancia también se encuentra disponible dentro del catálogo de modelos presentes en el simulador NEST.

Con los precedentes sobre el simulador NEST y un objetivo claro que es llevar a cabo la simulación de una neurona con estos modelos, en este capítulo se dará más detalle a nivel de código para el uso del simulador.

3.2. El simulador NEST

Así como el modelo *Integrate and Fire* fue de utilidad para visualizar el comportamiento que iba adquiriendo el modelo neuronal, será de utilidad para introducir los conceptos nuevos que trae consigo el simulador NEST.

Ahora que se cuenta con una representación digital de las neuronas, se pueden definir dos conceptos para su trabajo y manipulación, estos conceptos son: nodos y conexiones. Primero los nodos que forman una unidad de trabajo que se puede dividir en diferentes elementos:

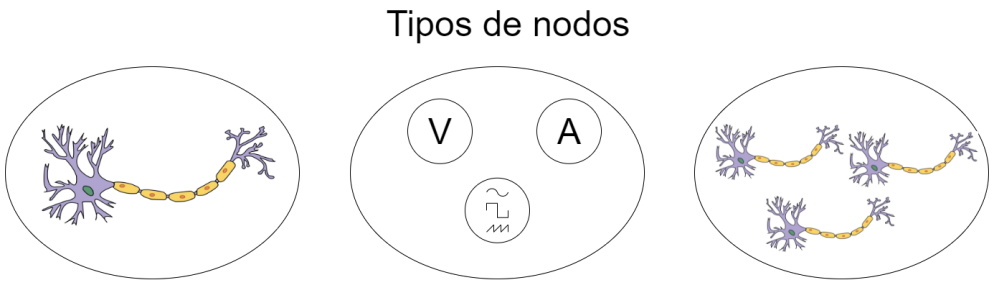


Figura 3.1: Un nodo representa una neurona, un dispositivo o a su vez una red neuronal en sí misma

Los dispositivos, al igual que en los ensambles vistos, son necesarios para estimular a las neuronas o realizar mediciones, los nodos son creados con el comando:

In [1]:
```
Create()
```

Y toman como argumentos el nombre del modelo y de manera opcional la cantidad de nodos a representar. Esta función nos retorna una `NodeCollection` que de manera breve, se trata de una colección de nodos con métodos muy similares a los arreglos de *Python*.

In [2]:
```
get()
```

Que permite acceder a las propiedades del nodo en una estructura similar a un diccionario, un nodo de neurona es inicializado de la siguiente manera:

In [3]:
```
neuron = nest.Create("iaf_psc_alpha")
```

Posterior a la inicialización del modelo, resulta útil consultar la documentación del modelo seleccionado y es grato saber que cada uno de estos modelos tiene su documentación específica, dentro de la documentación se puede encontrar más información a nivel de definición del modelo, las características detalladas del mismo y las fuentes que dieron origen o de donde se puede conseguir mayor información sobre la concepción de los modelos.

Por ejemplo, para el modelo *integrate and fire* de este tipo, se cuenta con la definición `iaf_psc_alpha`, en la documentación hay datos específicos sobre los cálculos, las variables y concepciones tomadas en cuenta durante la programación del modelo, estas variables van desde lo electrofisiológico, como el potencial de membrana a lo lógico como son ids de reconocimiento global para cada modelo neuronal en el caso del modelo `iaf_psc_alpha` se tienen los siguientes parámetros modificables:

V_m	mV	Potencial de membrana
E_L	mV	Potencial de membrana en estado de reposo
C_m	pF	Capacitancia de la membrana
tau_m	ms	Constante temporal de la membrana
t_ref	ms	Duración del periodo de refracción
V_th	mV	Límite de pico
V_reset	mV	Potencial de membrana en reseteo
tau_syn_ex	ms	Tiempo de excitación de la función de sináptica alfa
tau_syn_in	ms	Tiempo de inhibición de la función de la función sináptica alfa
I_e	pA	Corriente de entrada constante
V_min	mV	Valor mínimo absoluto para el voltaje de membrana

Muchos de estos parámetros están directamente relacionados con el modelo de *integrate and fire* descrito en el capítulo 2, de la idea original del *integrate and fire* se han añadido más parámetros durante el desarrollo del modelo, pero con el propósito

de homologar la simulación matemática con la informática se tomaran en cuenta principalmente aquellos parámetros que fueron previamente documentados.

Al utilizar el método `get()` de la neurona se obtiene un resultado similar al siguiente:

In [4]:
```
neuron.get()
```

Out [4]:

```
{'archiver_length': 0,
 'beta_Ca': 0.001,
 'C_m': 250.0,
 'Ca': 0.0,
 'E_L': -70.0,
 'element_type': 'neuron',
 'frozen': False,
 'global_id': 3,
 'I_e': 0.0,
 'local': True,
 'model': 'iaf_psc_alpha',
 'node_uses_wfr': False,
 'post_trace': 0.0,
 'recordables': ('I_syn_ex', 'I_syn_in', 'V_m'),
 'synaptic_elements': {},
 't_ref': 2.0,
 't_spike': -1.0,
 'tau_Ca': 10000.0,
 'tau_m': 10.0,
 'tau_minus': 20.0,
 'tau_minus_triplet': 110.0,
 'tau_syn_ex': 2.0,
 'tau_syn_in': 2.0,
 'thread': 0,
 'thread_local_id': -1,
 'V_m': -70.0,
 'V_min': -inf,
 'V_reset': -70.0,
 'V_th': -55.0,
 'vp': 0}
```

Se puede observar que muchas de las variables ya tienen un valor predefinido

y otras no cuentan como parte de los parámetros modificables, de aquí se puede destacar la propiedad `recordables`, en esta se encuentran definidos los elementos que pueden ser observados de la neurona, este ejemplo cuantifica la corriente de excitación de la función sináptica alfa (**I_syn_ex**), la corriente de inhibición de la función sináptica alfa (**I_syn_in**) y el voltaje de la membrana, que como se ha visto anteriormente es lo que se medía durante la simulación del modelo durante la formulación.

Con la neurona inicializada se puede comenzar a hacer lectura y escritura de los datos, por ejemplo, la lectura se puede realizar de la siguiente manera:

```
In [5]:  neuron.get("I_e")
         neuron.I_e

Out [5]: 0.0

In [6]:  neuron.get(["I_e","V_m","V_reset"])

Out [6]: {'I_e': 0.0, 'V_th': -55.0, 'V_reset': -70.0}
```

Y la lectura de manera análoga la escritura se puede realizar de la siguiente manera:

```
In [7]:  neuron.set(I_e=500.0)
         neuron.set({"I_e": 500})
         neuron.I_e = 500.0
```

Así se pueden variar todos los valores que se definieron en la documentación como parámetros.

Con la neurona lista se puede dar paso a otro elemento del tipo nodo, los dispositivos, el nombre es bastante claro sobre a lo que hace referencia, pero siempre resulta necesaria una aclaración sobre cómo es que se hace referencia a estos dispositivos dentro del simulador, un ejemplo de esto se puede observar con los

multímetros.

Los multímetros son dispositivos con los que se puede medir corriente, voltaje o resistencia (Halliday et al., 2013), en el ensamble de la neurona el funcionamiento debe de ser el mismo, dentro del simulador NEST el multímetro se encarga de grabar las variables dependientes del tiempo, en este caso, el voltaje de la neurona.

In [8]:
```
multimeter = nest.Create("multimeter")
multimeter.set(record_from=["V_m"])
```

Otro dispositivo importante es el `spike_recorder`, que se encarga de detectar y almacenar los picos de reacción de la neurona.

In [9]:
```
spikerecorder = nest.Create("spike_recorder")
```

Ahora se cuenta con 3 nodos independientes como lo muestra la figura 3.2, existiendo dentro de la simulación por lo que es necesario hablar sobre el otro tipo de elemento, las conexiones.

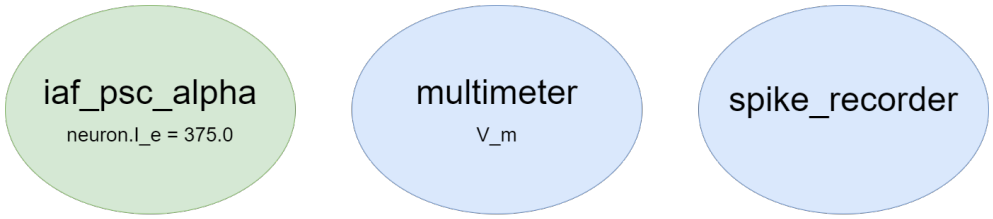


Figura 3.2: De manera aislada se cuenta con un nodo(verde) y dos dispositivos(azul)

Las conexiones se dan por medio de la función `connect()`, está conecta de manera predefinida todos los elementos siendo el primer parámetro el nodo presináptico y el segundo el nodo post sináptico, pero como se vio anteriormente, es de vital importancia definir cuando una neurona envía un mensaje a otra, con la relación pre-post sináptica.

Volviendo al ejemplo, la conexión se realiza de la siguiente manera.

In [10]:
```
nest.Connect(multimeter, neuron)
nest.Connect(neuron, spikerecorder)
```

Así es como el multimetro va a estar constantemente solicitando información a la neurona para ir grabando todo el evento, de manera inversa, el `spike_recorder` únicamente está interesado en almacenar información cuando la neurona tenga un pico este ensamble se comportaría como la figura 3.3.

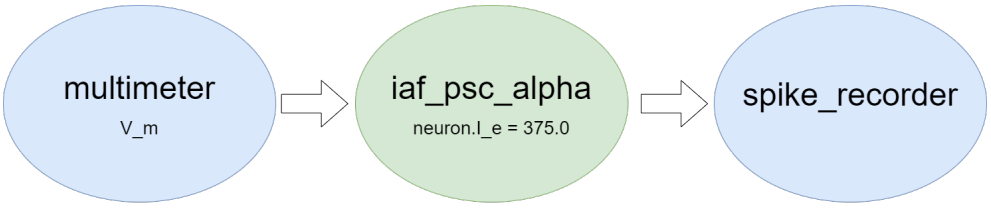


Figura 3.3: Con esta configuración el multímetro consulta constantemente información a la neurona y la neurona avisa al detector de picos cada que hay un potencial de acción

Así quedarían relacionados los nodos por lo que al ejecutar una simulación:

In [11]:
```
nest.Simulate(1000.0)
```

Todos los nodos dentro de la ejecución son simulados con base en los parámetros y a su vez los dispositivos conectados a estos obtienen la información definida durante su construcción.

De momento la información únicamente se encuentra encapsulada dentro de las variables, para poder graficar estos resultados es necesario obtener los datos almacenados en los dispositivos medidores.

Con la simulación realizada, tanto el multímetro como el `spike_recorder` tienen una gran cantidad de datos almacenados, pero solo se trata de variables dentro de un arreglo, que aunque relevantes, no son datos especialmente claros para la visualización del modelo, un ejemplo de esto se puede observar al solicitar los elementos del multímetro con el método `get()`.

In [12]:
```
multimeter.get()
```

In [12]:
```
{'element_type': 'recorder',
 'events': {'senders': array([4, 4, 4, ..., 4, 4, 4]),
  'times': array([1301., 1302., 1303., ..., 2398.,
  2399.]),
  'V_m': array([-70., -70., -68.27862371, ...,
  -55.08213919,
      -70., -70.])},
 'frozen': False,
 'global_id': 7,
 'interval': 1.0,
 'label': '',
 'local': True,
 'model': 'multimeter',
 'n_events': 1099,
 'node_uses_wfr': False,
 'offset': 0.0,
 'origin': 0.0,
 'record_from': ('V_m',),
 'record_to': 'memory',
 'start': 0.0,
 'stop': 1.7976931348623157e+308,
 'thread': 0,
 'thread_local_id': 6,
 'time_in_steps': False,
 'vp': 0}
```

Dentro contiene una serie de arreglos y variables en sí mismo, de las cuales se destaca "`events`" que contiene a su vez más arreglos en particular uno con el tiempo "`timesz` uno con el voltaje "`V_m`" para extraer esta información del diccionario se hace la referencia anidada:

In [13]:
```
dmm = multimeter.get()
Vms = dmm["events"]["V_m"]
ts = dmm["events"]["times"]
```

Almacenando primero el multímetro y haciendo referencia a los arreglos dentro del elemento "`events`", ahora con dos arreglos de variables del mismo tamaño la gráfica resulta directa, ubicando el tiempo en el eje x y el voltaje de la membrana en el eje y.

In [14]:
```
import matplotlib.pyplot as plt
plt.figure(figsize=(8, 6), dpi=80)
plt.plot(ts, Vms)
```

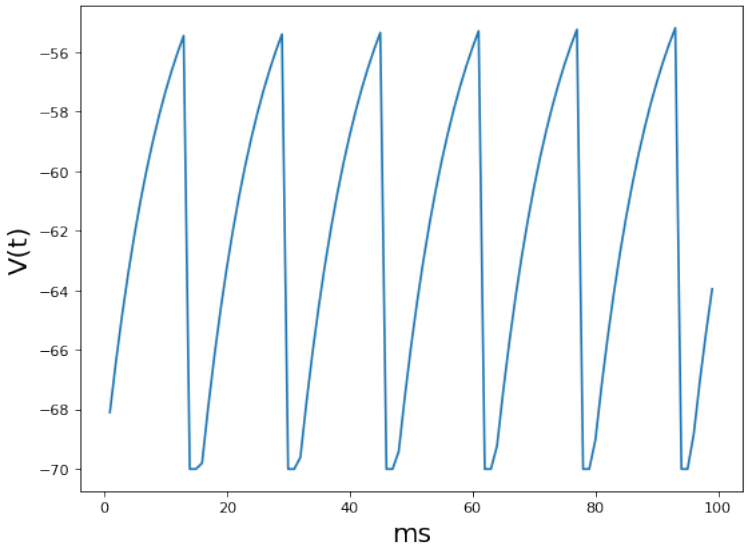


Figura 3.4: Con el input de 500 pA se pueden observar los diferentes picos a lo largo de la simulación

Como se ha visto anteriormente, se puede consultar internamente específicamente un elemento del diccionario de eventos y facilitando un poco la consulta de los elementos del diccionario.

In [15]:
```
dSD = spikerecorder.get("events")
evs = dSD["senders"]
ts = dSD["times"]
```

Por su parte la gráfica es homóloga.

In [16]:
```
plt.figure(figsize=(8, 6), dpi=80)
plt.plot(ts, evs, "+")
plt.show()
```

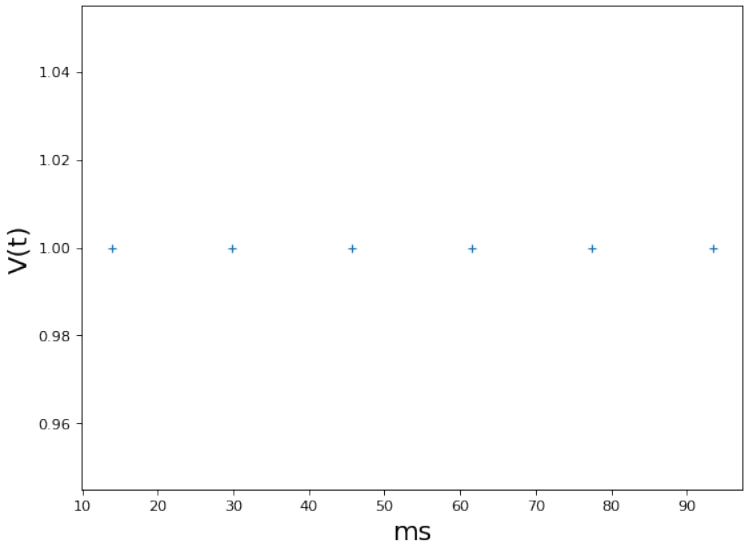


Figura 3.5: El detector de picos permite aislar los picos del potencial de acción de la demás información

Para la visualización el elemento que representará a los picos, en este ejemplo un símbolo de más (+).

En resumen así es como el simulador trabaja aprovecha el resultado de la implementación realizada por los colaboradores del simulador NEST y facilita el proceso permitiendo realizar experimentos con mucha mayor agilidad, se está más limitado en cuanto a la información particular que se puede obtener, pero a cambio se gana una mayor agilidad para trabajar.

Para dar fin a este subtema me gustaría mostrará en unas pocas líneas de código

como es que el simulador NEST puede trabajar con diferentes modelos, en esta ocasión el modelo Hodgkin-Huxley con un identificador **hh_psc_alpha**:

In [17]:
```
neuron_hh.I_e = 5000
multimeter_hh = nest.Create("multimeter")
multimeter_hh.set(record_from=["V_m"])
nest.Connect(multimeter_hh, neuron_hh)
nest.Simulate(100.0)
dmm = multimeter_hh.get()
Vms = dmm["events"]["V_m"]
ts = dmm["events"]["times"]
plt.figure(figsize=(8, 6), dpi=80)
plt.plot(ts, Vms)
```

Todo lo visto hasta el momento en una presentación compacta, dando como resultado la gráfica 3.6:

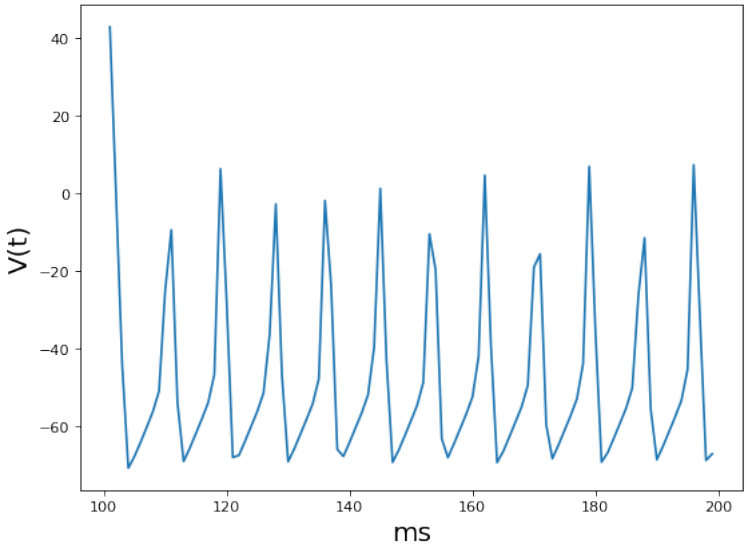


Figura 3.6: La entrada de una gran intensidad de corriente eléctrica se observan muchos picos en un periodo corto de tiempo

Y así es como se puede simular una neurona de manera individual, pero aún falta más por aprender sobre el simulador ya que una neurona no es suficiente para realizar la configuración neuronal, por lo que resulta necesario hablar sobre la conexión de

las neuronas y la influencia que están tendrán con las otras.

3.3. Simulando una configuración neuronal

Con una neurona individual no se puede definir una configuración neuronal por lo que resulta necesario retomar el otro tipo de elemento del simulador NEST, las conexiones. Anteriormente se habían realizado las conexiones entre las neuronas y los dispositivos, en ese momento se explicó que la función `conect()` parte de ciertos parámetros, pero de manera más precisa está definida de la siguiente manera:

```
Connect(pre, post, conn_spec=None, syn_spec=None,
                    return_synapsecollection=False)
```

Parámetros

- `pre(NodeCollection)` Nodos presinápticos o un arreglo con los ids de los nodos que conecta.
- `post(NodeCollection)` Nodos postsinápticos o un arreglo con los ids de los nodos que conecta.
- `conn_spec(str o dict, opcional)` Especifica reglas de conexión:
  - Estás reglas de conexión definen el orden de conexión de los nodos, por defecto la conexión es de todos con todos `(all_to_all)`, especialmente relevante cuando se tiene más de un nodo pre y post sináptico.
- `syn_spec(str o dict, opcional)` Especificación de la sinapsis:
  - Se trata de una especificación más detallados sobre el tipo de conexión, ya sea el peso(weigth), el modelo de `sinapsis(synapse_model)`, el retraso(`delay`) o tipo de receptor(`receptor_type`).
- `return_synapsecollection(bool)` Especifica si se retorna una SynapseCollection, tipo de objeto encargado de la representación de las conexiones.

Para ejemplificar este tipo de conexión primero se definen las neuronas a conectar, nuevamente del modelo `iaf_psc_alpha`:

In [1]:
```
import nest
neuron1 = nest.Create("iaf_psc_alpha")
neuron1.set(I_e=500.0)
neuron2 = nest.Create("iaf_psc_alpha")
multimeter = nest.Create("multimeter")
multimeter.set(record_from=["V_m"])
spikerecorder = nest.Create("spike_recorder")
nest.Connect(neuron2, spikerecorder)
```

Lo que se busca con la comunicación neuronal es que la neurona 2 obtenga información como resultado del estímulo compartido por la neurona 1 es por esto que únicamente se le inyecta corriente a la neurona 1 y al tratarse en este ejemplo únicamente de dos neuronas la conexión es bastante directa, utilizando el método `Connect()` se declara lo siguiente:

In [2]:
```
nest.Connect(neuron1, neuron2, syn_spec =
{"weight":20.0})
nest.Connect(multimeter, neuron2)
```

La intención de este código es conectar la neurona 1 con la neurona 2, especificando un peso de 20 unidades, este peso es una unidad interna dada dentro del simulador y que entra en juego al ser normalizada por definición en conjunto con las demás neuronas del ensamble, de momento sirve solo para ilustrar el concepto.

Al extraer la información del multímetro de la neurona 1 y graficarlo se obtiene:

In [3]:
```
dmm = multimeter_neuron_1.get("events")
Vms = dmm["V_m"]
ts = dmm["times"]
import matplotlib.pyplot as plt
plt.figure(figsize=(8, 6), dpi=80)
plt.plot(ts, Vms)
```

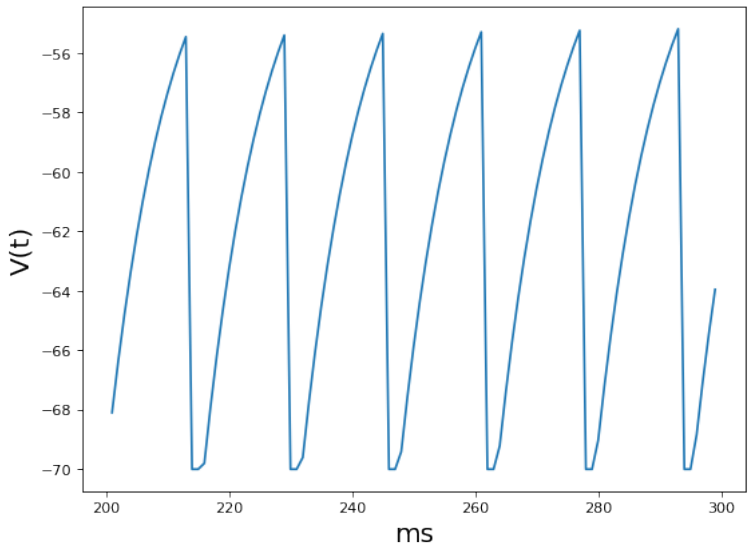


Figura 3.7: El comportamiento es bastante similar a lo visto anteriormente pero con una menor frecuencia

Y de la extracción de información de la neurona 2 se obtiene lo siguiente:

In [4]:
```
dmm = multimeter_neuron_2.get("events")
Vms = dmm["V_m"]
ts = dmm["times"]
import matplotlib.pyplot as plt
plt.figure(figsize=(8, 6), dpi=80)
plt.plot(ts, Vms)
```

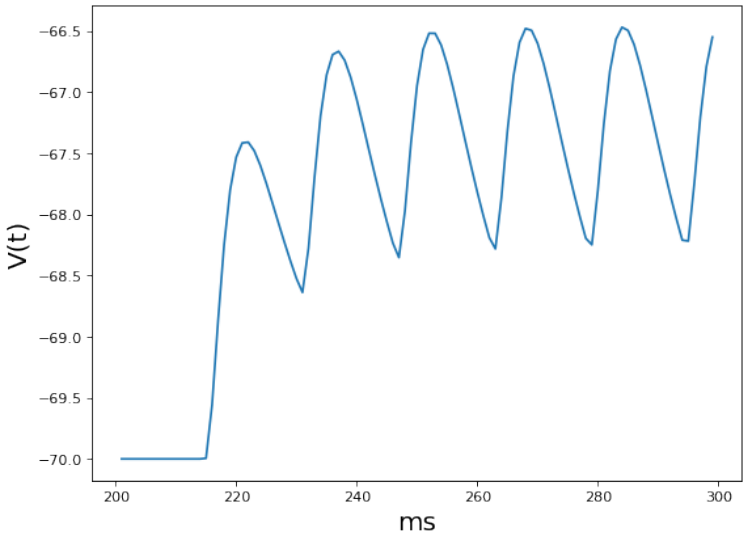


Figura 3.8: Se puede observar como los potenciales de acción provenientes de la neurona 1 producen una reacción continua en la neurona 2

El comportamiento es bastante diferente al de una neurona de *integrate and fire* normal, ya que los picos no llegan a alcanzar el límite predefinido pero a pesar de esto tiene un comportamiento en picos, además no se le inyectó ninguna corriente, lo que da la idea de que los picos de la otra neurona manda información de manera temporal, pero al no ser suficiente, esta segunda neurona no alcanza su potencial de acción y nunca llega a el `V_reset`.

Sin embargo, esta es una red bastante pequeña, apenas con dos neuronas, y la definición de cada una de ellas lleva su atención particular, para poder escalar la creación de las neuronas se puede desarrollar una función que automatice el proceso, pero esto no es necesario ya que el simulador cuenta con más elementos para facilitar la creación de las neuronas a través del método `Create()`.

Durante la creación de cada nodo se utiliza el método `Create()` que requiere la especificación del tipo de nodo que se va a generar, ya sea una neurona o un dispositivo, pero de manera interna, el método tiene otros parámetros y está conformado de la siguiente manera:

```
Create(model, n=1, params=None, position=None)
```

Parámetros

- `model(str)` Nombre del modelo a simular
- `n(int, opcional)` Número de nodos a crear
- `params(dict o list, opcional)` Parámetros para los nuevos nodos, pueden ser:
  - Un diccionario con cada valor individual o lista de tamaño n (Estos parámetros idénticos para cada modelo o puede especificarse de manera individual)
- `positions(spatial.grid o spatial.free objeto, opcional)` Objeto describiendo las posiciones espaciales de cada nodo, en caso de omitirse no se toma en cuenta la posición de las neuronas en el espacio.

Este último parámetro es necesario para algunas aplicaciones aunque no será necesario en este documento.

Así quedaría la creación de dos neuronas en la misma función de `create`, a esta asignación de colecciones se le llamará conjunto:

In [6]:
```
neuron = nest.Create("iaf_psc_alpha",2)
```

A su vez esto puede escalar tanto como se requiera, con 100 neuronas el proceso es similar:

In [7]:
```
neuron = nest.Create("iaf_psc_alpha",100)
```

La salida al hacer una consulta de las variables se resume en diccionarios con arreglos de longitud 100 conteniendo cada uno de los valores de cada neurona, aunque por la declaración realizada todas las neuronas tienen exactamente los mismo valores y al usar una asignación convencional:

In [8]:
```
neuron.set({"I_e": 200})
```

Todas las neuronas se modifican en paralelo, otra manera de poder hacer esta asignación durante la creación de las neuronas es enviando un diccionario como parte de los `params` del método `create`:

In [9]:
```
params = {"I_e": 200.0}
neuron = nest.Create("iaf_psc_alpha",100, params=params)
```

O definiendo un *default* para el modelo:

In [10]:
```
nest.SetDefaults("iaf_psc_alpha", params)
```

El problema que surge cuando se definen así los valores es que todas las neuronas contenidas en cada conjunto tendrán los mismo parámetros, cuando el conjunto es pequeño se puede asignar de manera directa con un diccionario y se manda un arreglo del mismo tamaño que la colección de nodos con la que se va a trabajar:

In [11]:
```
neuron.set({"I_e":[100,200,300]})
```

Esto deja claro que se pueden aprovechar las asignaciones de *Python* para dar valores de manera clara y dinámica, inclusive haciendo uso de distribuciones particulares.

Este es un simulador neuronal, existen otras alternativas de simuladores con sus diferentes métodos, fortalezas y debilidades, el simulador NEST tiene una rápida implementación y un lenguaje bastante asequible para alguien proveniente de un área computacional.

El simulador NEST cuenta con un apartado de publicaciones en su página principal donde se puede investigar más sobre el impacto y utilidad que ha aportado en otros trabajos de investigación.

Una manera de retratar la importancia de conocer como una serie de neuronas reaccionan, interaccionan y se influencian unas a otras es para entender que como el cerebro reacciona de manera generalizada a los estímulos externos.

Un equivalente médico a provocar estímulos y visualizar cómo los impulsos

eléctricos van modificando el comportamiento del cerebro es mediante un **EEG**, un electroencefalograma es un procedimiento donde el doctor obtiene el registro de la actividad bioeléctrica proveniente de la corteza cerebral.

Los resultados se obtienen ubicando electrodos en el cráneo del paciente en puntos determinados, como en la figura 3.9, así el doctor puede visualizar si en alguna zona del cerebro del paciente hay una reacción anormal y así poder diagnosticar una patología. Los resultados de un encefalograma se pueden observar en la figura 3.9:

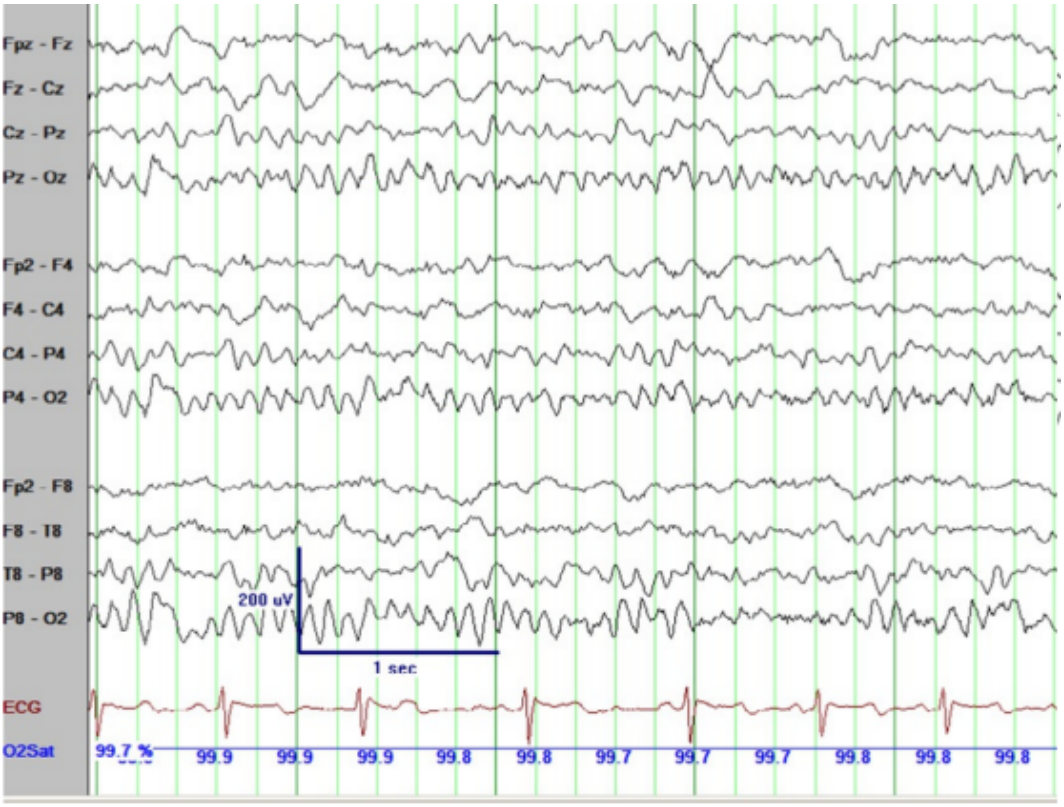


Figura 3.9: Por medio de la observación de un EEG se puede diagnosticar y monitorear desordenes neurologícos en los pacientes. *Nota.* Adaptado de *EGG*(p.108), por Feyissa y Tatum, 2019, Elsevier

Lo cual puede recordar a como el simulador NEST modifica el `V_m` de la neurona a través del tiempo con base en la corriente inyectada.

Es por esto que resulta necesario hacer una implementación que facilite el trabajo necesario para definir las neuronas, sus parámetros, los aparatos de medición y finalmente las gráficas resultantes.

Con datos dinámicos y con el precedente del uso del simulador, se comienzan a plantear nuevas posibilidades como lo es hacer funciones que automaticen un poco la prueba de modelos con distintos parámetros y distintos periodos de tiempo de manera más rápida.

Es por eso que se plantean dos funciones con comportamientos similares pero escalando la cantidad de neuronas implicadas.

Estas funciones son:

`multiple_experiments` y `multiple neurons`

`multiple_experiments( start, end, steps, time)`:

Parámetros

- `start( float )` - Número que indica la corriente inicial en pA que será inyectada a la neurona.

- `end( float )` - Número que indica la corriente en pA a la que se quiere llegar al inyectar corriente a la neurona.

- `steps ( float )` - Número que indica el aumento en pA que va a tener la corriente inyectada a la neurona en cada iteración.

- `time ( float )` - Número que indica el tiempo en ms que será simulado.

La intención de esta función es probar con distintas corrientes y visualizar los resultados que esta va teniendo con la variación en el parámetro de la corriente.

El algoritmo básico de la ejecución de la función es el siguiente:

Los resultados obtenidos de esta ejecución con los parámetros

`multiple_experiments(500,800,100,300)` se obtiene lo siguiente:

Algoritmo 1 multiple_experiments(inicio, final, pasos, tiempo)

```
arreglo_neuronas ← np.arange(inicio,final,pasos)
tamaño ← arreglo_neuronas.size
neuronas ← nest.Create(\quotes{hh_psc_alpha}, tamaño)
neuronas.set({"I_e":neurons_array})
multimetro ← nest.Create("multimeter", tamaño)
nest.set(record_from=["hh_psc_alpha"])
nest.Connect(multimeter,neurons,conn_spec="one_to_one")
nest.Simulate(time)
pyl.figure(figsize=(8,size*2),dpi=800)
neuronas ← pyl.title("Hodgkin-Huxley Neuron")
para cada x en tamaño hacer:
    pyl.subplot(size+1,1,x+1)
    pyl.plot(multimeter.get("events")[x]["times"],
multimeter.get("events")[x]["V_m"])
    pyl.ylabel("I_e:" + str(neurons.get("I_e")[x]))
final para
pyl.show()
```

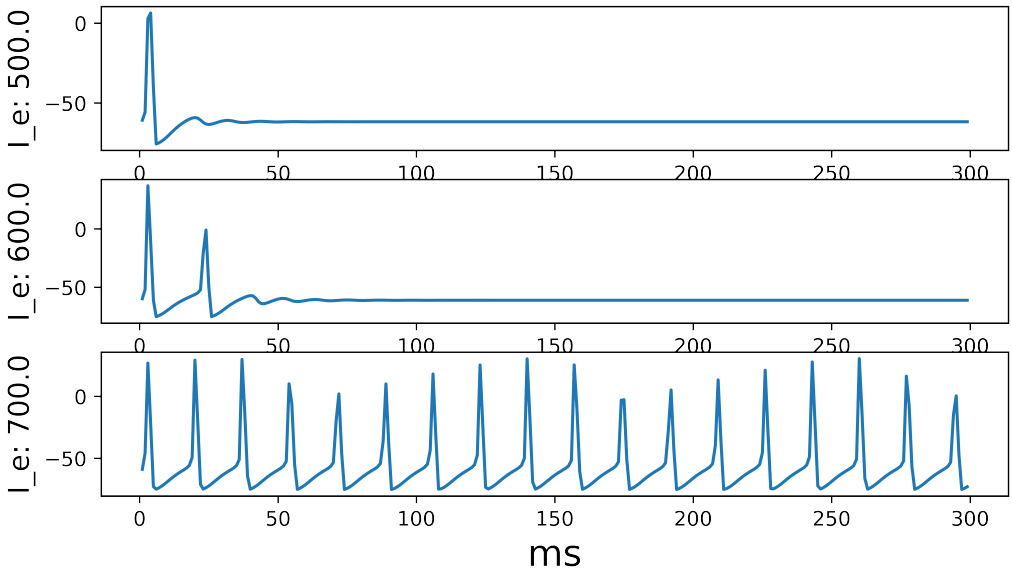


Figura 3.10: La función genera la cantidad de neuronas necesarias para llevar a cabo los pasos

Lo que la función muestra es la reacción que va teniendo una neurona con el modelo *Integrate and Fire* cuando se inyectan corrientes que van desde los 200 hasta los 400 pA avanzando 30 pA por paso hasta lograr su objetivo, se puede observar que las neuronas no tienen una reacción continúa hasta llegar a una corriente de 400

pA, y a partir de ahí las neuronas aumentan su frecuencia.

Ahora con una pequeña función es posible llevar a cabo una serie de experimentos variando la corriente inyectada a un modelo de neurona particular y observando su reacción.

Se tomó el primer tipo de implementación con neurona única y se escaló a otro nivel, por lo que la implementación de un modelo neuronal con la interacción de dos neuronas es inmediato por lo que se crea otro *script* que realiza lo siguiente:

Definición de las variables:

In [12]:
```
def multiple_neurons(start, end, steps, time, model,
con_type):
    size = np.arange(start,end,steps).size
    array = np.arange(start,end,steps)

    neurons_pre = nest.Create(model, size)
    neurons_pre.set({"I_e":array})

    neurons_post = nest.Create(model, size)

    multimeter_pre = nest.Create("multimeter", size)
    multimeter_pre.set(record_from=["V_m"])

    multimeter_post = nest.Create("multimeter", size)
    multimeter_post.set(record_from=["V_m"])
```

Conexión de las neuronas:

In [12]:
```
nest.Connect(neurons_pre,neurons_post,conn_spec=con_type)
nest.Connect(multimeter_pre,neurons_pre,conn_spec=
con_type)
nest.Connect(multimeter_post,neurons_post,conn_spec=
con_type)
```

Simulación y graficación:

In [12]:
```
nest.Simulate(time)
pyl.figure(figsize=(12, size),dpi=400)

pyl.title('Neurona pre-sinaptica')
for x in range( size ):
    pyl.subplot(size*2,1,x*2+1)
    pyl.plot(multimeter_pre.get("events")[x]["times"],
    multimeter_pre.get("events")[x]["V_m"])
    pyl.ylabel(str(neurons_pre.get("I_e")[x]))

    pyl.subplot(size*2,1,x*2+2)
    pyl.plot(multimeter_post.get("events")[x]["times"],

        multimeter_post.get("events")[x]["V_m"])
    pyl.ylabel(str(neurons_post.get("I_e")[x]))
pyl.show()
```

Esto da la oportunidad de aumentar de manera significativa la cantidad de experimentos que se pueden realizar sin tener que ingresar neurona a neurona y realizando la conexión manual de los multímetros y las colecciones de nodos.

En un primer experimento con este *script* se probó la función con los parámetros `multiple_neurons(500,800,100,300,"iaf_psc_alpha","one_to_one")`,

que define una corriente inicial de 500 pA una final de 1000 pA, con unos pasos de 100pA, además el modelo a utilizar es el *Integrate and Fire* y el tipo de conexión de las neuronas es de todos a todos, el resultado se muestra en la figura 3.11.

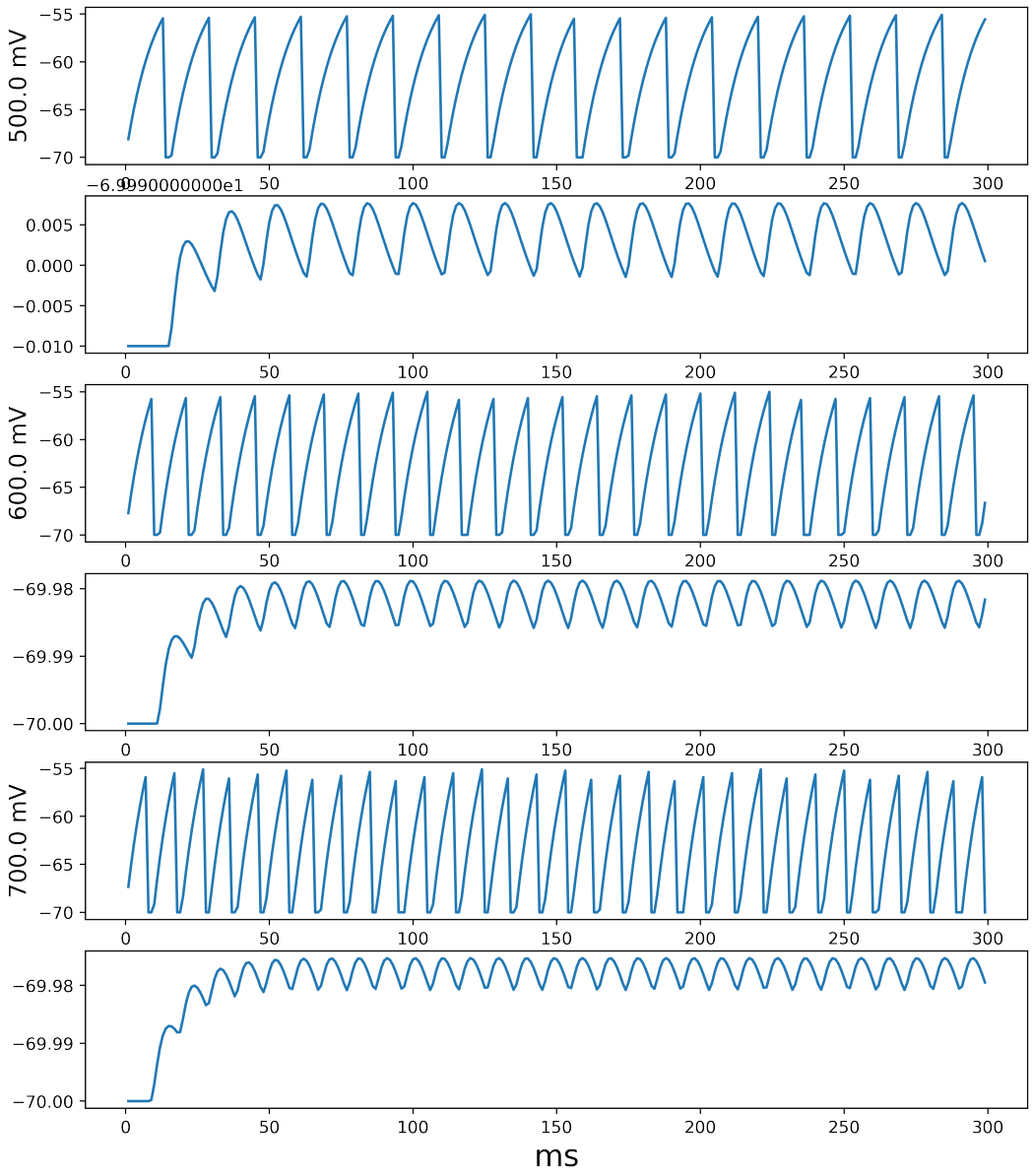


Figura 3.11: Esta función puede llegar a recordar al electroencefalograma, solo que es evidente que es algo más controlado

Es visible que cuando una neurona recibe una corriente desde otra neurona su comportamiento es equivalente, están teniendo una comunicación de uno a uno, pero qué pasa se conectan todas las neuronas con todas con los mismos parámetros iniciales dando la función, `multiple_neurons(500,800,100,300,"iaf_psc_alpha","all_to_all")`, pero en esta ocasión los resultados del lado de las neuronas postsinápticas son idénticos, esto

debido a que la interacción de las presinápticas es reconocida de manera idéntica para las neuronas postsinápticas y de hecho estos resultados son bastante similares a un EEG.

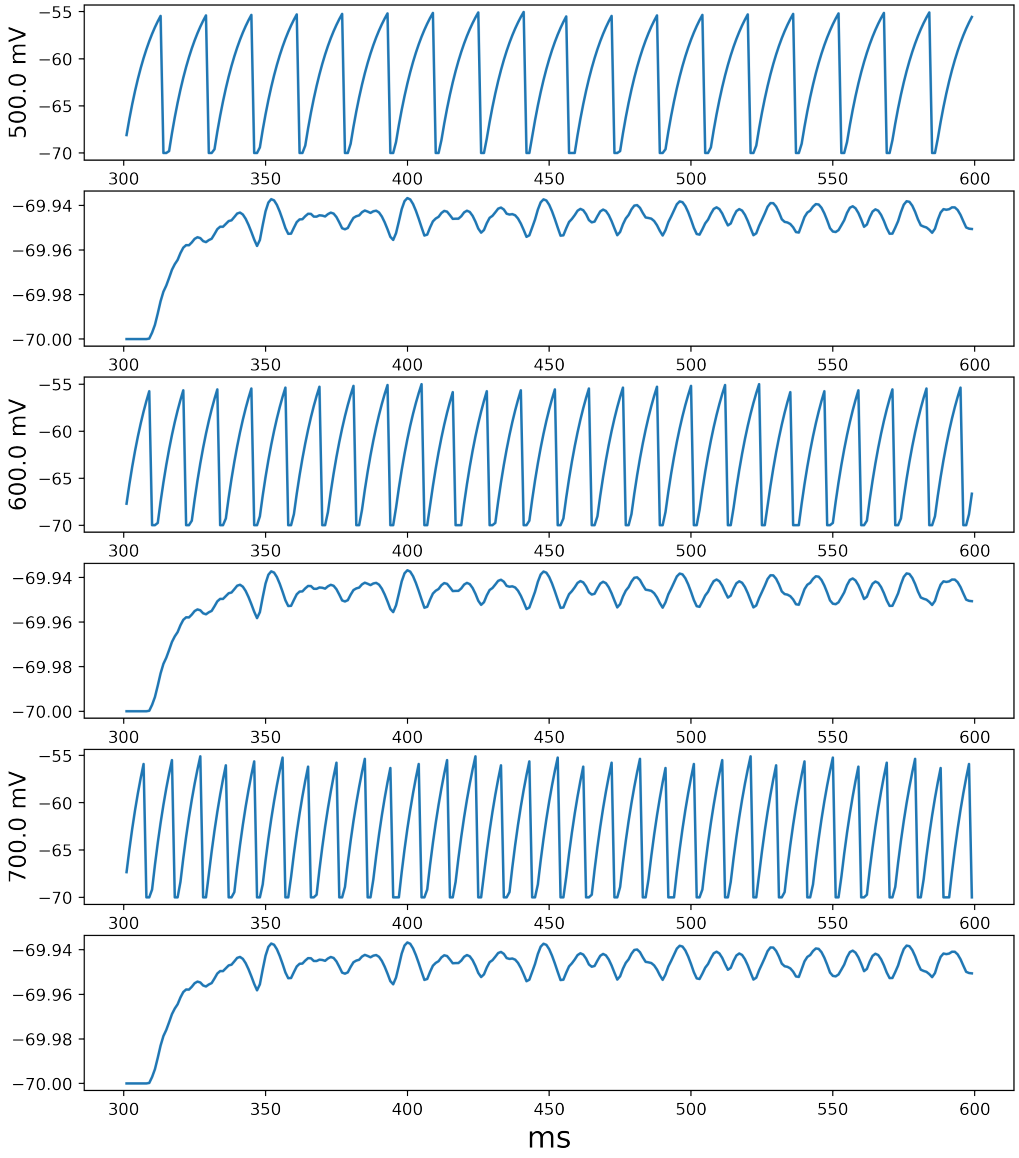


Figura 3.12: Esta función puede llegar a recordar al electroencefalograma, solo que es evidente que es algo más controlado

Esta última configuración resulta clave para el contraste entre los resultados experimentales como en la figura 3.14 y será aquella designada para servir de punto de referencia, por lo que para concluir con este capítulo resulta interesante ver cómo es que por medio de lo aprendido es posible replicar los resultados realizados

anteriormente con implementación realizada por Padraig Gleeson presente en la figura 3.13.

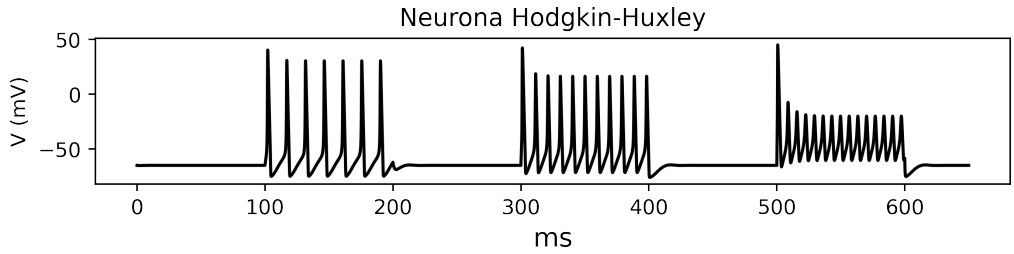


Figura 3.13: Esta primera gráfica es el resultado de aplicar el modelo HH de manera diferencial

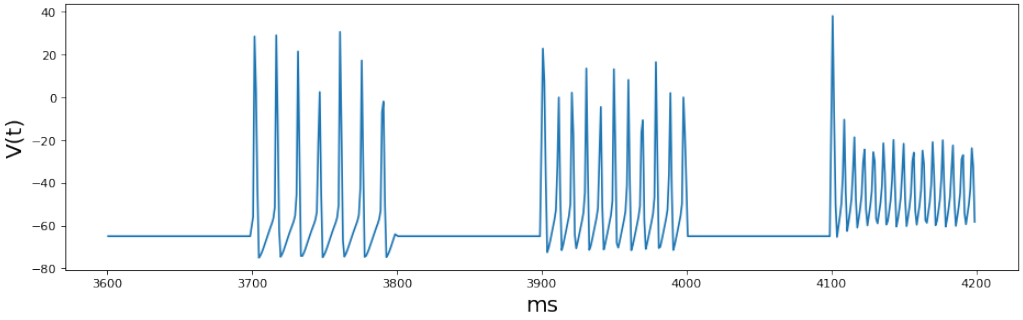


Figura 3.14: Esta segunda gráfica es el resultado de la aplicación del simulador NEST

Con esto se da por terminado el tercer capítulo de este documento, las simulaciones que se han llevado a cabo hasta el momento demuestra que con un conocimiento superficial sobre neurología computacional es posible llevar a cabo la práctica de simular configuraciones neuronales sencillas, pero es necesario indagar mucho más para poder realizar experimentos más adecuados, sin salir del mismo simulador NEST se cuenta con la determinación de pesos aleatorios basados en distribuciones o una ubicación espacial.

En el siguiente capítulo se hará una descripción general de cómo es que se puede acceder a resultados experimentales y como estos se corresponden a las simulaciones realizadas anteriormente.

Capítulo 4

Resultados experimentales

A lo largo del presente documento se ha ido recorriendo el proceso de la digitalización de la neurona desde el campo del análisis matemático y computacional, desde sus antecedentes biológicos, su interpretación matemática y la simulación de la neurona con el apoyo del simulador NEST, permitiendo al individuo simular el funcionamiento del cerebro sin la necesidad de contar con un laboratorio o con toda la indumentaria necesaria para hacer físicamente el experimento, únicamente con un equipo de computo.

Además de la simulación existen otros acercamientos al campo de la neurociencia computacional, por medio de *Allen Brain Map*, es posible acceder a una gran cantidad de recursos dígitales de esta disciplina.

El *Allen Brain Map*, es un proyecto auspiciado por el Instituto Allen, una institución líder en la investigación a larga escala de temas médicos y biológicos, fundada por Paul G. Allen y su hermana Jody Allen en 2003.

Durante este capítulo el proposito será explorar la infomación proveniente del área de neurociencias con la que cuenta el Instituto Allen, de donde es de particular interes la estructura y el funcionamiento del *Allen Brain Map*. Al igual que el simulador NEST el proyecto está influenciado por la *Open Science*, dando acceso público a datos de investigación en beneficio al trabajo colaborativo.

4.1. Introducción del *Allen Brain Map*

El *Allen Brain Map* es un proyecto que pone a disposición de los investigadores una serie de herramientas y bancos de datos experimentales obtenidos por el instituto Allen, estos recursos contienen información sobre diferentes partes del cerebro, por lo que hay una gran variedad de recursos, cada uno con su proposito en particular. De manera general los atlas son los siguientes:

El *Allen Developing Mouse Brain Atlas* se encarga de capturar patrones genéticos en las neuronas de ratones transgénicos, el propósito de usar de ratones transgénicos es eliminar la mayor cantidad posible de variables, así permitiendo el análisis del cerebro del ratón durante su desarrollo.

El instituto Allen ha llevado a cabo la tarea de agrupar los distintos tipos de neuronas, ya sea por sus formas, propiedades eléctricas y conjuntos de genes, la base de datos de este tipo más grande se puede encontrar en el *Allen Cell Types Database* que caracteriza células del cerebro de humanos adultos y la corteza cerebral del ratón.

En el capítulo 1, durante el subtema de la visión y transmisión de la imagen fue posible observar como es que diferentes neuronas interaccionan con la finalidad de transmitir un mensaje, el *Allen Mouse Brain Connectivity Database* cuenta con un mapa en alta resolución sobre el cerebro de un ratón transgénico permitiendo la visualización de este tipo de procesos de una manera más técnica.

Finalmente, el *Allen Brain Observatory*, es un análisis de la actividad fisiológica del cerebro del ratón, controlando las respuestas del ratón frente a un estímulo visual y revisando la actividad de la corteza visual, hipocampo y tálamo a través de docenas de sesiones de electrofisiología.

Herramientas como lo son el *Toolkit* que incluye el acceso a los datos del instituto Allen, por medio de su API (*Application Programming Interface*) o su SDK (*Software Development Kit*).

Las tres herramientas utilizadas en el presente trabajo son el *Allen Cell Types Database* con la intención de comparar los obtenido previamente con las neuronas individuales, el *Allen Mouse Brain Connectivity Database (AMBCD)* para conocer

la configuración de las neuronas durante la experimentación real con todos los datos existentes y *Synaptic Physiology* para conocer más datos de conectividad neuronal en el cerebro del ratón, con un acceso más directo.

Los datos serán consultados por medio de las siguientes plataformas:

- *Allen Cell Types Database* a través con el SDK, una serie de librerías accesibles con Python.
- AMBCD será consultado desde el portal web, permitiendo un acceso más sencillo a los recursos y su visualización.
- *Synaptic Physiology* por medio de *aisynphys una librería para python, que cuenta con herramientas de escritorio para consumir su información.*

La diversidad en el tipo de datos para los diferentes equipos, provocó un problema con el almacenamiento de la información y en especial en un entorno nuevo como lo es el desarrollo de una base de datos con información neurológica, se necesito llegar a un consenso, así es como se da origen al NWB (*Neurodata Without Borders*), actualmente en su versión 2.0 liberada en el 2019 (Albin, 2019).

Uno de los primeros acercamientos que tuvo la neurología a esta homologación de información fue *Neuroshare* [1], un sitio con la misión de dar soporte al desarrollo colaborativo de librerías abiertas y formatos de archivos de datos para neurofisiología y en general información *open source* para neurocientíficos.

Neuroshare sirvió como un primer acercamiento a esta estandarización y donde destacó una problemática nacida de la misma naturaleza de los experimentos neurológicos, la necesidad del contexto sobre los experimentos, NWB parte con el requerimiento de incluir todos los metadatos de cada experimentos.

Además de esto se plantean dos desafíos principales para permitir una mejor interpretación sobre los datos. El primer desafío es expresar las diferentes piezas de los datos y describir las relaciones entre ellas, como lo son el tiempo relativo entre el estímulo y la señal neuronal. El segundo desafío es desarrollar un esquema

[1] Sitio web: http://neuroshare.sourceforge.net/index.shtml.

de almacenamiento que permita a los usuarios acceder a información similar sobre elementos en común de una manera compatible, un ejemplo de esto puede ser obtener catálogo de técnicas de grabación.

Este proyecto parte con dos desarrolladores de tiempo completo y con diferentes colaboradores de medio tiempo entre los cuales están algunos neurocientíficos del instituto Allen, la primera reunión tuvo lugar en noviembre del 2014, la segunda en mayo del 2015 y el proyecto se dio por terminado en julio del 2015, más información se puede encontrar en su cuenta de github [2].

El equipo del NWB eligió HDF5 como el contenedor de la información, entre las ventajas de esta librería para el almacenamiento de datos están:

El almacenamiento de datos heterogéneos n-dimensionales donde cada elemento en sí puede ser un objeto complejo. La portabilidad de los archivos que gracias a su formato pueden ser transferidos sin muchas complicaciones. Compatibilidad con diversas plataformas, como lo son *C*, *C++*, *Fortran*, *Java* y *Python*. Acceso rápido a la información gracias a su estructura optimizada. La inexistencia de límites en el tamaño de los datos que pueden ser almacenados. La posibilidad de mantener los datos ligados a su *metadata* con *lifecycles* y *pipelines*. Estas características hicieron que HDF5 fuera seleccionado como el formato para almacenar, procesar y administrar los datos.

Con una primera versión de este recurso se inició el desarrollo de las API que permitió el uso del NWB desde otras plataformas como lo son *Python* y *MATLAB*, además el lenguaje de especificación, mapeado en XML permitía a algunas extensiones crearse de manera independiente y ser interpretadas a su vez por los distintos laboratorios (“HDF5”, 2019).

A pesar de que no es requisito conocer sobre el formato para el entendimiento de los atlas, resulta útil conocerlo ya que es el esquema en el que se puede realizar la lectura de los datos tanto para esta base de datos como para la consulta de cualquiera de los proyectos afiliados, además de que permite conocer más sobre la organización del instituto Allen como una parte importante en el desarrollo del

[2]Repositorio: https://github.com/NeurodataWithoutBorders.

proyecto colaborativo del *open science* especializado en neurociencia (Commission, 2016).

4.2. Simulación vs experimentación

Retomando la consulta de la información proveniente de los diferentes atlas, se inicia con el análisis e investigación correspondiente al estudio del *Allen Cell Types Database*, de manera general este atlas contiene un compilado de la información proveniente de células individuales, por su estructura electrofisiológica, morfológica y datos transcriptómicos provenientes del RNA de la células.

De este atlas es de especial interés la información electrofisiológica, estos datos fueron recabados usando diversos tipos de estímulos y protocolos, una de las metas de realizar este tipo de estímulos es poder crear modelos, es por esto que la información resulta perfecta para hacer el contraste con los modelos trabajados para una neurona individual.

La relevancia de este experimento radica en la forma de obtención de los resultados que resulta familiar a estas alturas del documento, por medio de la inyección de estímulos a neuronas localizadas dentro del cerebro de los ratones se hace la captura de los datos y se genera un modelo neuronal que permita replicar los resultados obtenidos.

Las herramientas de modelado cerebral BMTK (*Brain Modeling Toolkit*), y el formato SONATA (Dai et al., 2020) permiten acceder a diferentes modelos. El lenguaje de programación oficial es *Python* y hay una variedad de módulos y paquetes que permiten la exploración de los datos y la creación de otros modelos, para más información puede consultarse la documentación oficial del BMTK[3].

Este tipo de metodología que parte de un organismo físico hasta llevarlo a un modelo computacional resulta análogo a lo realizado hasta el momento.

El acceso a esta información es inmediato por medio del portal web de la

[3]Documentación del BMTK https://alleninstitute.github.io/bmtk/

herramienta[4], con el apoyo de la herramienta se puede llevar a cabo un filtrado selectivo, los datos de interés en este momento son las células provenientes de ratones transgénicos localizadas en el VISp (Corteza visual primaria) y con el modelo biofísico activo.

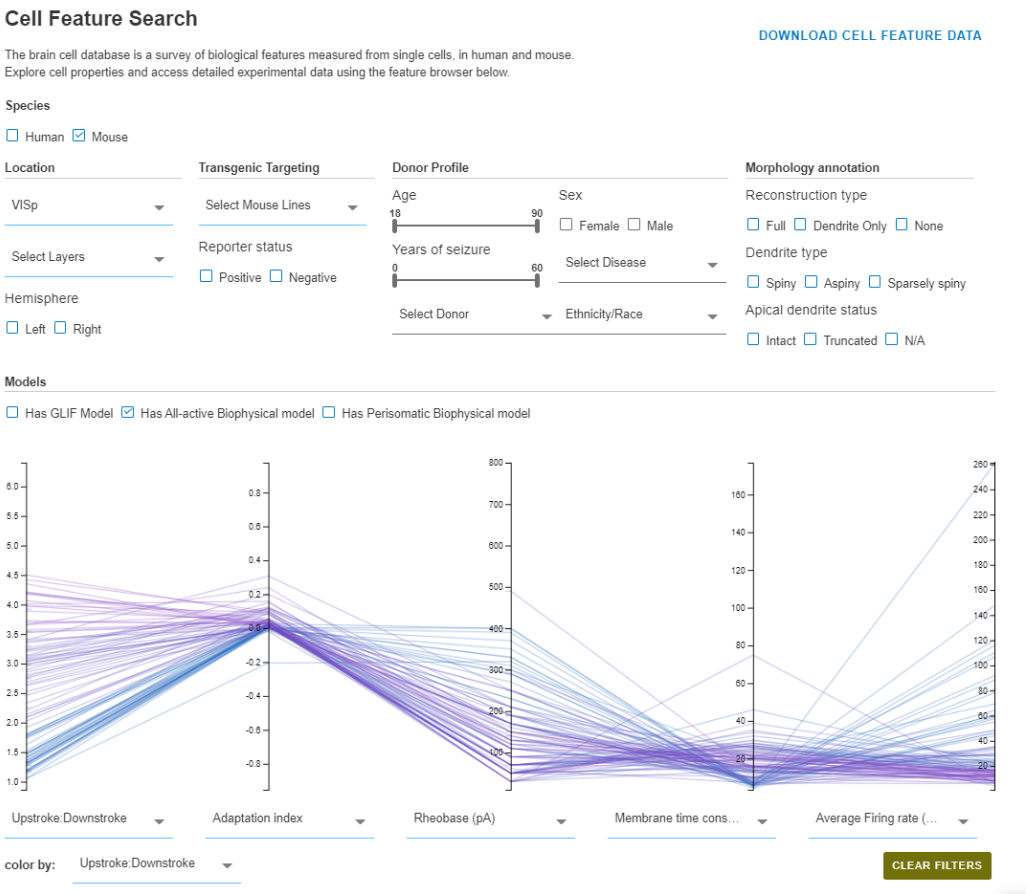


Figura 4.1: Por medio de la selección de diversos filtros se puede acotar las búsquedas

Tras la selección de los filtros se arrojan resultados con un breve resumen de la información existente sobre cada coincidencia.

[4]Herramienta web BMTK https://celltypes.brain-map.org/data

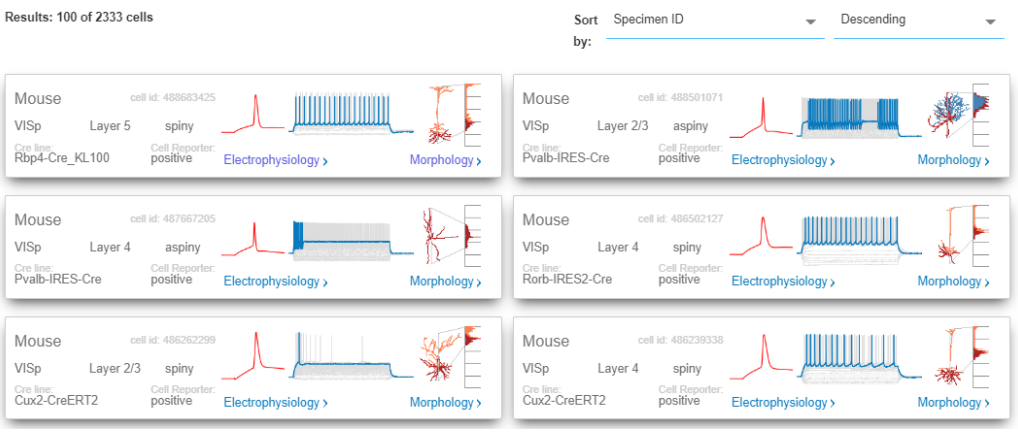


Figura 4.2: En la pantalla de resultados se puede ver un resumen de la información correspondiente a cada célula

De donde se puede seleccionar una célula a revisar, la elegida es la célula con ID 488683425, la cual se puede consultar en el siguiente enlace: http://celltypes.brain-map.org/experiment/electrophysiology/488683425

Tras la selección de la información electrofisiológica se observa el siguiente menú (ver figura 4.3):

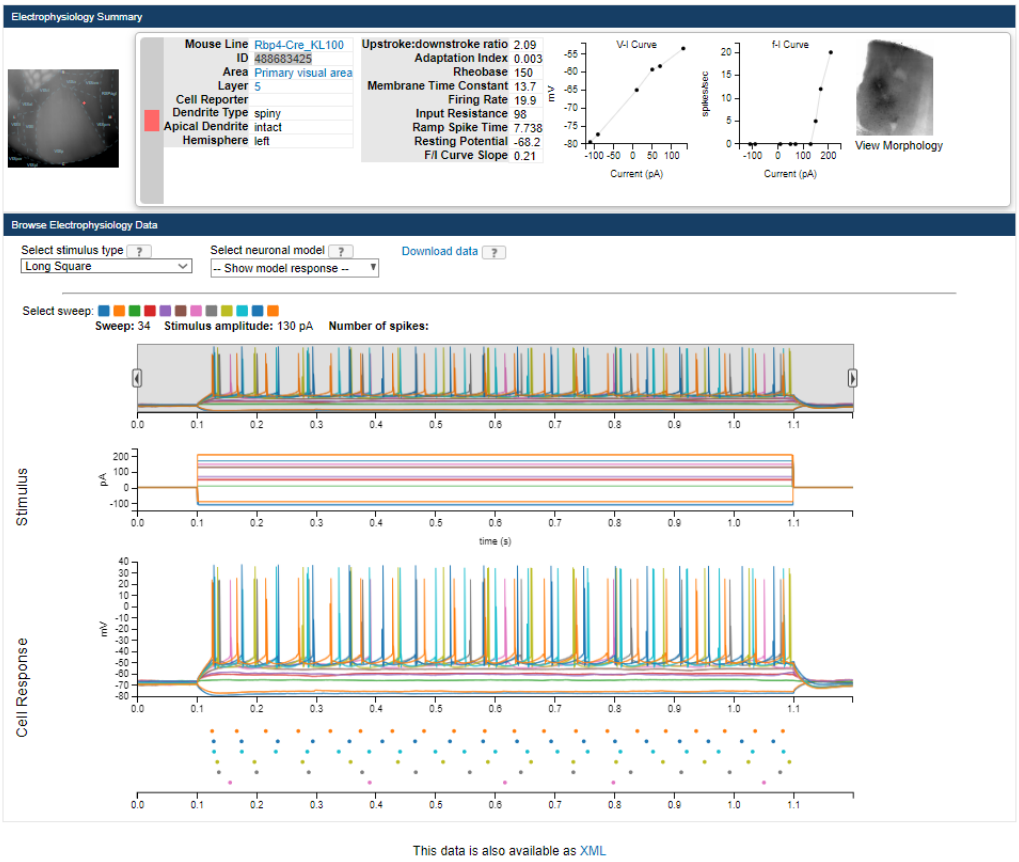


Figura 4.3: Tras la selección del experimento se puede visualizar el encabezado de los datos y la información electrofisiológica

La figura 4.3 contiene dos paneles, en la parte superior muestra, el resumen electrofisiológico, con la información general sobre el experimento, en la segunda sección se puede hacer un filtrado enfocado únicamente en la información electrofisiológica.

El segundo panel es interesante por la información que contiene. Estas figuras permiten hacer un contraste superficial con las gráficas obtenidas anteriormente por medio de los modelos Integrate and Fire y Hodgkin Huxley.

Los estímulos se dividen en barridos `sweep` donde la neurona va variando la amplitud del estímulo y la cantidad de picos presentes en la neurona.

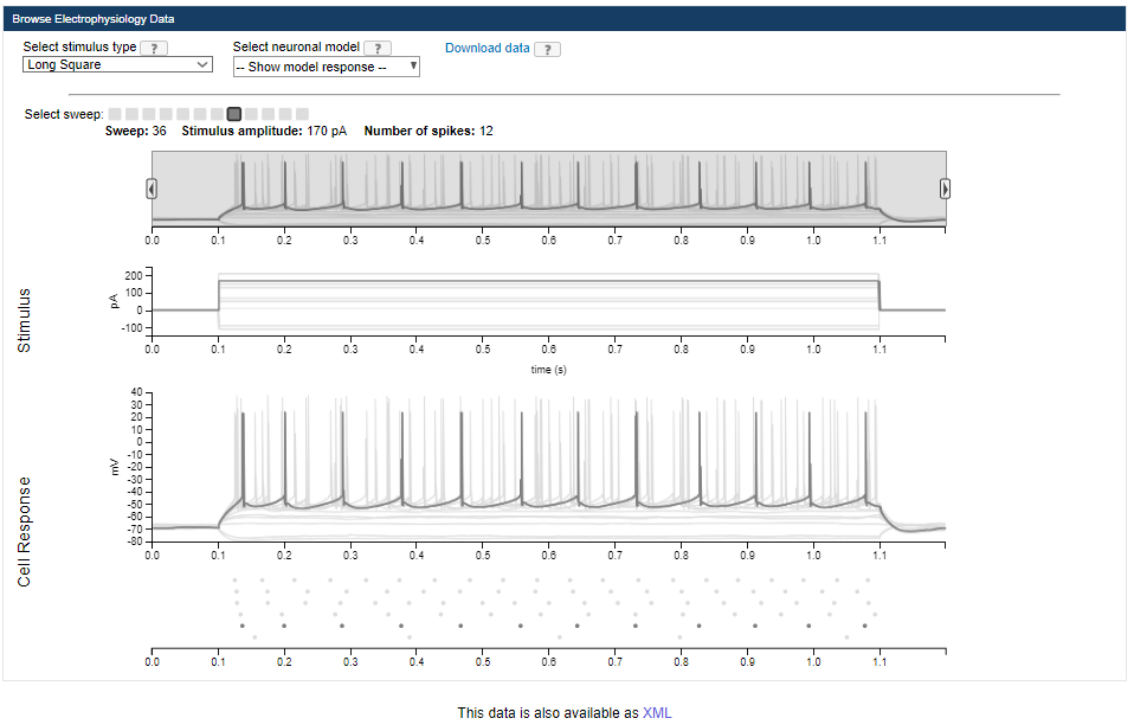


Figura 4.4: Se puede seleccionar el barrido de manera individual permitiendo ver el tipo de estimulo y la respuesta de la célula

Con la configuración mostrada en la figura 4.4 se puede observar una reacción en el potencial de membrana similar a la obtenida en experimentos previos, aunque no resulta suficiente para realizar el contraste para lo cual es necesario recuperar los datos individuales del experimento, disponibles por dos métodos de acceso.

Un primer método es por medio de la descarga de los datos en formato XML, el XML contiene un resumen de la información existente de la célula, de los barridos se consulta información general del estímulo, por lo que es necesario pasar al segundo método de acceso a la información, el archivo NWB (La descarga del documento NWB es por medio del link de "**Download data**").

La lectura del archivo se realizó por medio del Allen SDK, y como con el simulador NEST, se instaló en *Ubuntu* corriendo en WSL2, creando un ambiente en *Miniconda*, con *Python* en su versión 3.7 por temas de compatibilidad.

```
conda create -n AllenSDK python=3.7 anaconda
conda activate AllenSDK
```

El añadido de Anaconda prepara los elementos necesarios para la ejecución del ambiente de *Jupyter Notebook*, a instalación del Allen SDK es de manera directa con un comando `pip`.

```
pip install allensdk
```

Dentro de un nuevo cuadernillo de *Jupyter Notebook* se va a hacer la lectura del archivo NWB y la importación de las librerías necesarias.

In [1]:
```
import numpy as np
import matplotlib.pyplot as plt
from allensdk.core.nwb_data_set import NwbDataSet
```

Las librerías necesarias son *Numpy* para el procesamiento de los datos, *Matplotlib* para la presentación de los mismos y el Allen SDK para leer los `datasets`, es a partir de este punto cuando comienza el procesamiento del archivo, comenzando con la lectura de los datos:

In [2]:
```
file_name = '488683423_ephys.nwb'
data_set = NwbDataSet(file_name)
```

Al contar ya con la información almacenada en la variable de `data_set` se puede hacer una extracción de los diferentes barridos presentes en el experimento.

In [3]:
```
sweep_numbers = data_set.get_sweep_numbers()
sweep_numbers
```

En la información almacenada se puede observar un arreglo numérico con el número de barrido individual para cada estímulo.

Out [3]:
```
[12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 28, 30,
    31, 34, 35, 36, 38, 4, 43, 44, 46, 47, 48, 5, 50, 51,
    53, 56, 57, 58, 59, 6, 64, 65, 66, 68, 72, 8, 9]
```

Para la extracción de la información es necesario seleccionar un barrido y dar la indicación para acceder a esa información y esta asignación da la posibilidad de hacer la lectura de cada uno de los elementos individuales.

In [4]:
```
spike_times = data_set.get_spike_times(sweep_number)
stimulus = sweep_data['stimulus']
response = sweep_data['response']
sampling_rate = sweep_data['sampling_rate']
index_range = sweep_data['index_range']
```

Con los datos se puede realizar la gráfica de resultados por ejemplo para el barrido 36 la gráfica resultado es la siguiente en el caso del objeto `response`:

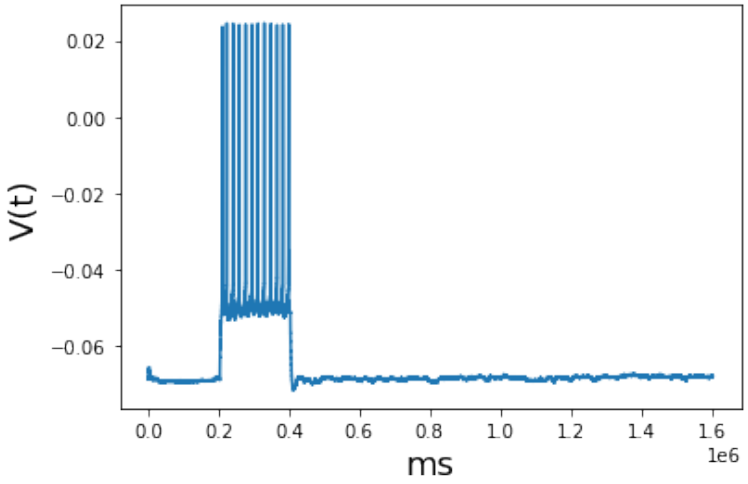


Figura 4.5: A pesar de que la escala se encuentra normalizada se puede observar que se trata de la misma respuesta que en el portal web

La información del XML contiene más detalles sobre el experimento como que el estímulo fue 170 pA y un estado de reposo de -68 mV, fue un estímulo cuadrado de un segundo así que se inyectó la corriente durante un segundo de manera constante.

Continuando con la base de datos de conectividad sináptica ("Synaptic Physiology", 2019), donde se estudia como es que el cerebro de ratones y algunos humanos reaccionan a los estímulos por medio de experimentos basados en pinzas, muy similares a los realizados por Hodgkin y Huxley.

Durante un experimento se apuntan a 8 neuronas mediante electrodos de vidrio. Cada neurona es estimulada por potenciales de acción mientras se mide el potencial en las otras neuronas, si dos neuronas están conectadas por una sinapsis se observan las corrientes sinápticas de unos pocos ms por cada pico presináptico.

Cada sinapsis en la base de datos está atada a una cantidad indeterminada de conexiones presinápticas, por lo que se cuenta con un potencial postsináptico (**PSP**) del cual se pueden medir propiedades como lo son amplitud promedio, latencia de la respuesta del pico, la forma del potencial PSP y la varianza entre intento e intento.

Durante un experimento un conjunto de estímulos es proporcionado a las neuronas conectadas, para visualizar las características mencionadas previamente, estos estímulos consisten en 8 pulsos inyectados a la neurona para detectar la plasticidad de la neurona a corto plazo, seguido por un retraso y 4 pulsos más para detectar la recuperación.

Las frecuencias utilizadas para estimular las neuronas tiene un periodo de descanso de 250 ms entre los 8 pulsos de inducción y los 4 de recuperación, estos estímulos varían entre frecuencias.

Para cuantificar la demora se parte de una frecuencia de 50 Hz y se varia la demora entre los pulsos de inducción y los 4 de recuperación así es posible tener una mejor imagen de cómo es que las neuronas se recuperan del estímulo inicial.

Los estímulos de frecuencia mixtos están conformados por un 8 estímulos principales a 30 Hz y a partir de ahí 30 pulsos en intervalos irregulares que van entre los 5 y los 100 ms. El propósito de este tipo de estímulo permite explorar un rango mayor en las frecuencias presentes en el modelo.

Cada prueba contiene un tipo de frecuencia como los mencionados anteriormente y son repetidos por lo menos cinco veces con un periodo de 15 segundos entre las pruebas repetidas, midiendo tanto las células presinápticas como las postsinápticas (“Experimental stimuli”, 2019).

La documentación para el experimento además de los estímulos, define la estructura reconocible de las neuronas así como su morfología y localización dependiendo de la capa cortical.

Gracias a la versatilidad de los ratones transgénicos se realizó un etiquetado de las subclases excitatorias e inhibitorias de las neuronas, más detalles sobre esta clasificación se puede encontrar en el documento de caracterización transgénica[5].

Para evitar que la entrada de datos no relevantes a la base de datos se aplicó una serie de procesos de análisis evitando datos incompletos e información sobre células muertas, por medio del estímulo y con base en la interacción con sus células vecinas es posible identificar las conexiones y así agregar esta conexión a la base de datos.

Volviendo a la *aisynphys*, tratándose de un subconjunto de datos del *Allen Brain Atlas* se encuentran los elementos almacenados en diferentes archivos de *sqlite*, cada uno con un nivel diferente de especificación, datos presentes en la tabla 4.1.

La base de datos de la sinapsis contiene los resultados de muchos experimentos y la información almacenada por cada uno de estos se puede clasificar de manera general en:

Metadata experimental: Especies, Regiones del cerebro y condiciones experimentales. Propiedades celulares: Localización, morfología, reportadores transgénicos y características electrofisiológicas. Propiedades sinápticas:Fuerza, latencia, forma del PSP, varianza y plasticidad en cortos periodos de tiempo.

Todo almacenado en una tabla *SQL* organizada de la siguiente manera (ver figura 4.6):

[5]Caracterización transgénica http://connectivity.brain-map.org/transgenic

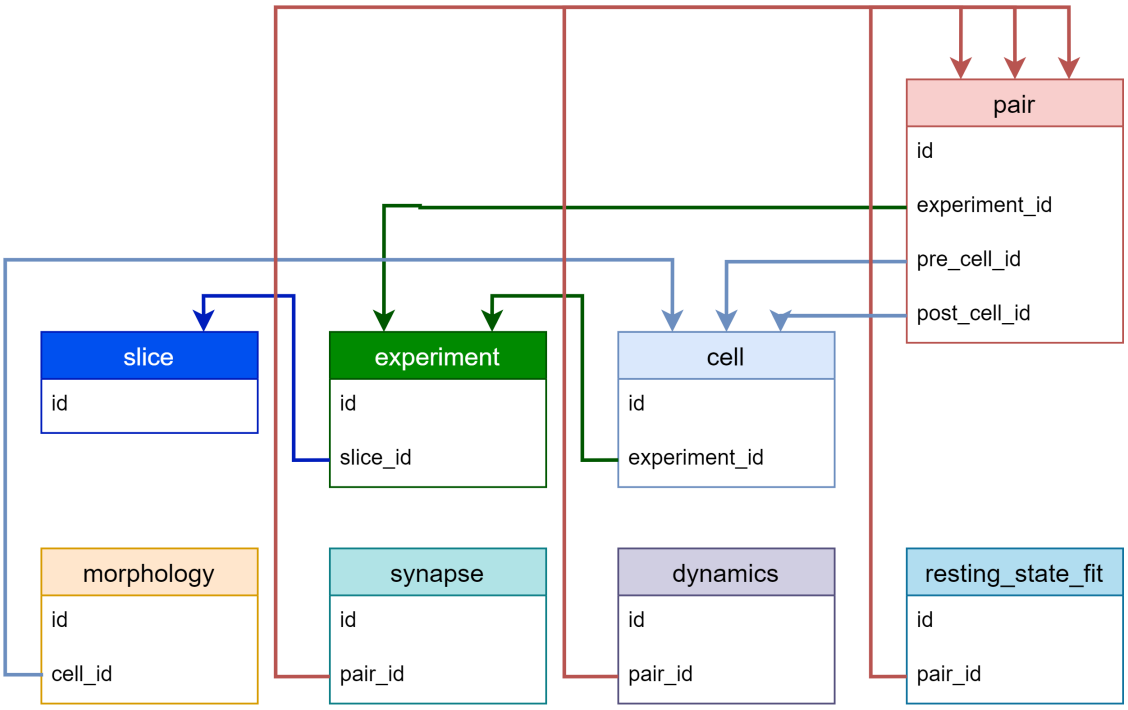


Figura 4.6: El acceso a los recursos se hace mediante el ingreso a una base de datos relacional con la estructura mostrada anteriormente

Para el trabajo con estos datos se cuenta con distintas alternativas, que van desde la herramientas visuales hasta las APIs de consulta, pero para evitar la saturación de información se seleccionó la opción de la consulta mediante la API. A su vez para este tipo de consultas se cuenta con dos accesos principales que se pueden resumir, como una instalación normal y el acceso por medio de un repositorio montado en *Binder*.

Binder [6] es una plataforma de *hosting* con soporte por parte de *Google Cloud*, *OVH*, *GESIS Notebooks* y el instituto Turin donde por medio de un repositorio de *Git* se pueden montar cuadernillos interactivos de *Jupyter* multipropositos, la utilidad para este proyecto es un acceso rápido y sencillo a las herramientas que nos proporciona el *Aysinphys*. La segunda alternativa es la instalación local por medio de una plataforma como lo es *Anaconda*.

A diferencia de ocasiones anteriores la instalación se llevó a cabo dentro de la plataforma de Anaconda, la cual consta de la descarga de la última versión del

[6]Ver sitio web http://binder.org.

repositorio existente:

```
git clone https://github.com/alleninstitute/aisynphys
cd aisynphys
```

Y haciendo la preparación del ambiente para *Miniconda*:

```
conda env create --name aisynphys \
--file desktop-environment.yml
```

Como en ocasiones anteriores la ejecución está apoyada con *Anaconda*, además desde la descarga del repositorio se puede contar con algunos elementos de ayuda para un mejor entendimiento de la información presentada, dentro de estos cuadernillos se encuentra el de tutorial que servirá de guía en los pasos siguientes para empezar a probar las partes del *Aisynphys.*

Se puede obtener la especificación sobre los experimentos y la obtención de los datos desde el documento oficial.

Se puede decir que la cantidad de datos involucrada con el *Aisynphys* es inmensa y de un alto grado técnico, asimismo esto sólo es una fracción del *Allen Brain Atlas* y uno de tantos proyectos documentados con el NWB. El simple desarrollo del NWB podría ser un tema a desarrollar por su cuenta, se está convirtiendo en un referente para cualquier centro de investigación que tenga la intención de colaborar en el proyecto en conjunto que es la *Open Science* de neurología computacional.

Es por esta magnitud que para el contraste de la simulación y experimentación, se seleccionó una de las herramientas más accesibles con las que cuenta el *Aisynphys.*

La recolección de la información sináptica permitió el uso de herramientas de *Machine Learning* para la clasificación de diversos tipos de neuronas basados en sus interacciones (Gouwens et al., 2019) pero se trata de una fuente almacenada en bases de datos con diferentes niveles de anidación entre los datos, es debido a esto que se realizará el análisis con el apoyo las herramientas interactivas del Instituto Allen.

Las herramientas interactivas, a diferencia de una consulta directa utilizando el SDK del Instituto Allen, despliegan la información de manera más directa, agilizando un poco más este proceso de obtención de información y contraste de manera general.

La herramienta que se seleccionó para el contraste en el documento es el *Matrix Analyzer*, gracias a esta herramienta se es posible hacer un despliegue rápido de un volumen más grande de la información.

Cabe destacar que a diferencia de las herramientas utilizadas anteriormente, donde se aprovechaba el entorno WSL, en esta ocasión se realizó el la integración directa en *Windows*, esto debido a que su construcción basada en *Python* requiere el uso de *Pyqtgraph* que despliega una ventana con la información, algo más complicado de permitir desde WSL.

Como su nombre lo indica, el `matrix analyzer`[7] hace un análisis de la información del *Asynphys* y la despliega a manera de matriz, ubicando los grupos de células presinápticas en las filas y las células postsinápticas en las columnas.

Su despliegue parte de la instalación previa del ambiente de *Aisynphys*, por lo que sólo resta activarlo:

```
conda activate aisynphys
```

Navegar a la ruta donde está la raíz del *Aisynphys* y ejecutar el siguiente comando

```
python tools/matrix_analyzer.py --db-version=DB_SIZE
```

Es así como el *script* despliega la ventana encargada de mostrar la información. Entre la información que se envía como parámetro al *script* se encuentra el *DB_SIZE* donde se especifica el tamaño de la base de datos a utilizar, las propiedades se pueden ver en la tabla 4.1, se puede encontrar más información sobre esta base de datos en la página principal del *Allen Brain Map*.

[7]Ver sitio web https://aisynphys.readthedocs.io/en/master/matrix_analyzer.html

Cuadro 4.1: Dependiendo del tamaño de la base de datos son incluidos más metadatos sobre cada experimento

	Pequeño (∼170 MB)	**Mediano (∼7 GB)**	**Completo (∼ 160 GB)**
Metadatos experimentales fuente del tejido, condiciones experimentales	X	X	X
conexiones químicas / eléctricas probabilidades de conexión	X	X	X
propiedades celulares clases transgénicas, capa, morfología, fisiología intrínseca	X	X	X
propiedades promedio por-conexión fuerza, cinética, dinámica, respuestas sinápticas promedio, propiedades de conexión eléctricas	X	X	X
modelos de liberación sináptica por conexión parámetros de liberación cuántica, plasticidad de término-corto	X	X	X
propiedades de respuesta por estímulo ajuste de curvas para respuestas sinápticas individuales		X	X

Continua en la página siguiente.

Cuadro 4.1 – Continua de la página anterior.

	Small (∼170 MB)	**Medium** (∼7 GB)	**Full** (∼ 160 GB)
propiedades de pico por-estímulo temporización y mediciones de picos presinápticos individuales		X	X
estímulos / grabación de metadata		X	X
datos de respuesta por-estímulo datos de series temporales para cada pico presináptico y su registro postsináptico correspondiente			X

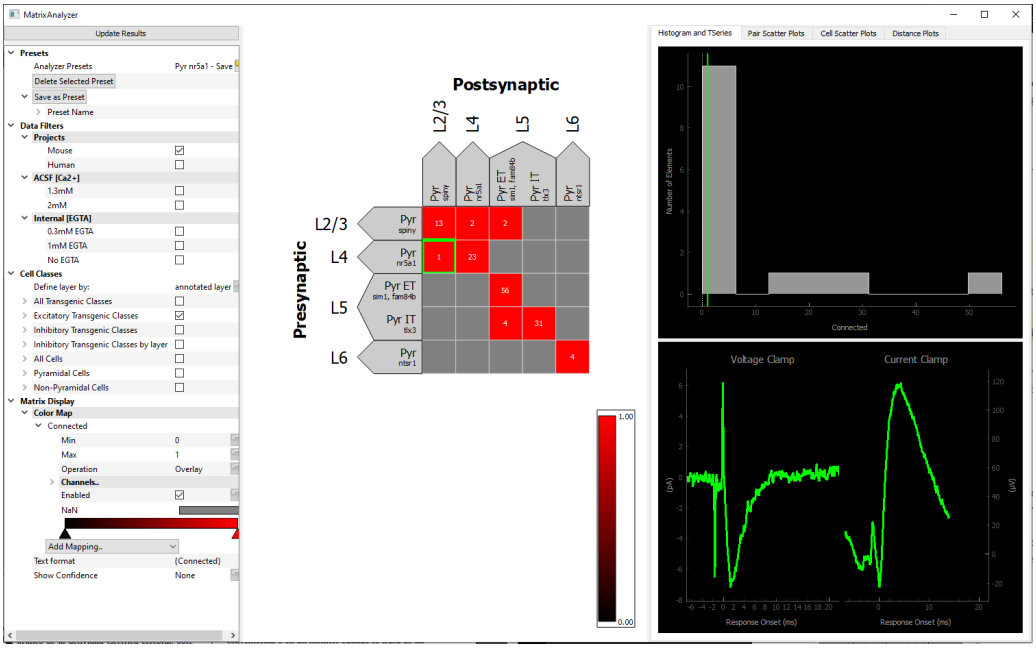


Figura 4.7: Vista general del *Matrix Analyzer* parte del atlas de conexión sináptica

Estas bases de datos se descargan como ficheros *sqlite*, y dependiendo de las necesidades del experimentador se hace la descarga de bases de datos.

Posterior a la ejecución del *script* aparece la interfaz gráfica del *matrix analyzer* (ver figura 4.7), que se puede separar en 3 secciones bien diferenciadas, estas son el panel de control, el panel que contiene la matriz y el panel de gráficas.

El panel de control (ver figura 4.8) permite seleccionar con base en los requerimientos las células que entrarán a consideración para el análisis, por medio de la selección de una configuración predeterminada o la elección de parámetros de manera individual.

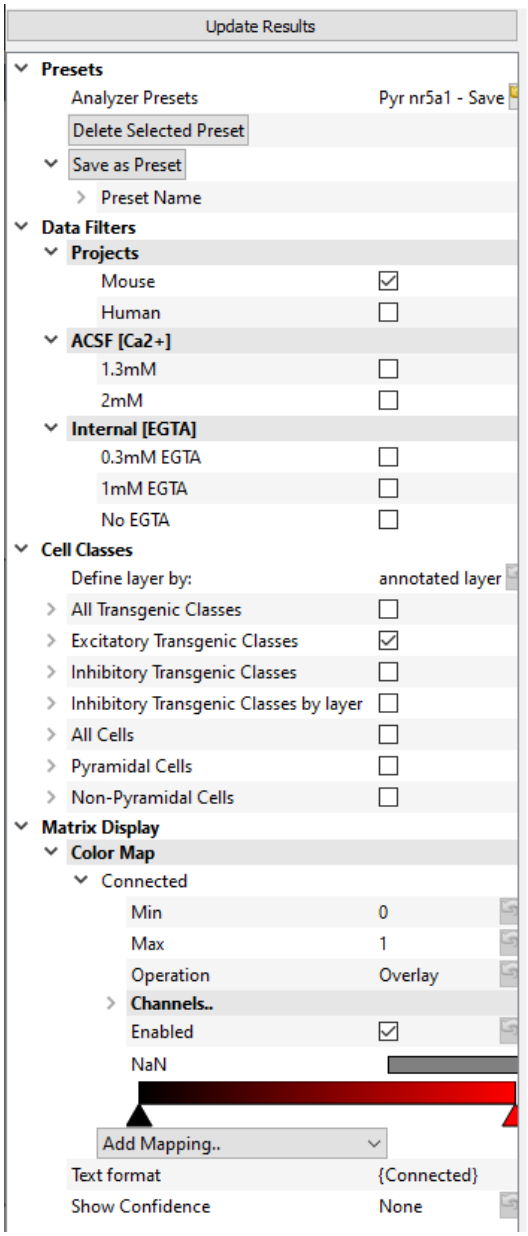


Figura 4.8: Panel de control de Matrix Analyzer

Sobre este primer panel es necesario destacar la última sección, *Matrix Display* por medio de esta selección se determina que data se incluirá en la matriz y cómo se seleccionan los distintos colores, un ejemplo de esta selección son las probabilidades de conexión entre las neuronas o la amplitud de sus PSP, con base en los parámetros y esta última selección se pueden sesgar los datos presentados en el siguiente panel, el panel de matriz.

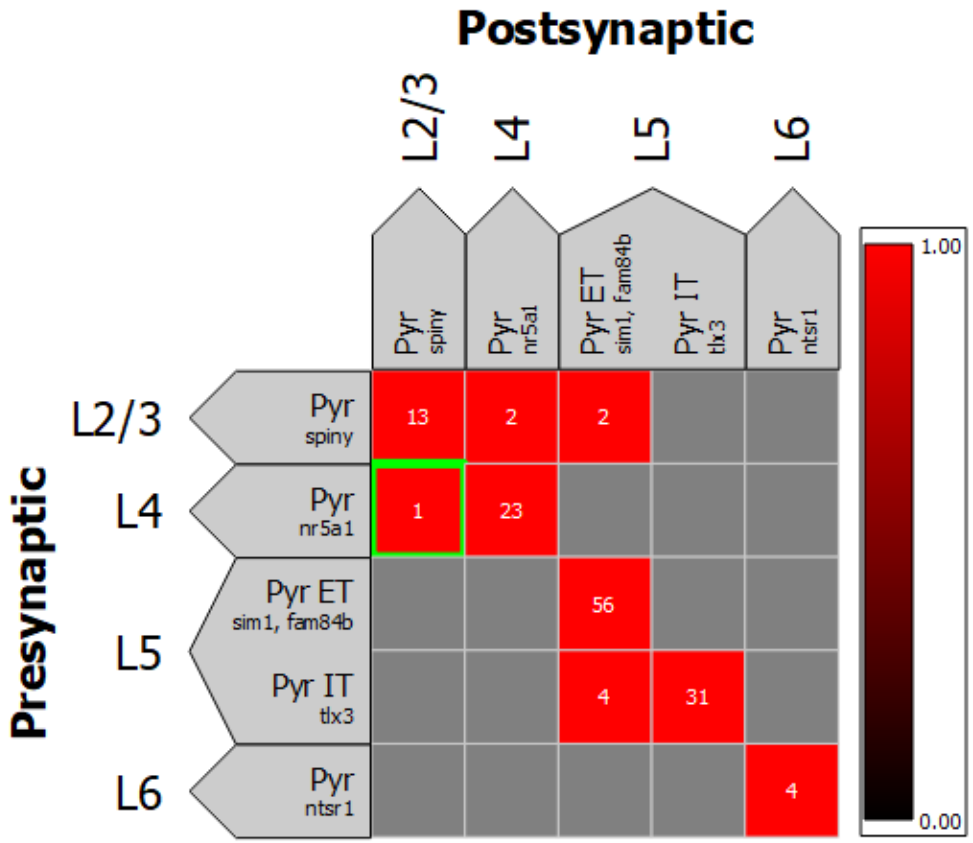


Figura 4.9: Conexiones neuronales obtenidas tras el filtrado en el *Matrix Analyzer*

Figura 4.10: En la parte superior de este panel de control se encuentran las pestañas con los tipos de gráficas a mostrar

En este segundo panel (ver figura 4.9) se encuentra una matriz que resume los datos consultados por los filtros iniciales simplificando los resultados y mostrándolos de manera consecuente con la relación existente entre las regiones del cerebro que se tomaron en cuenta, designando las neuronas presinápticas a las filas y las

postsinápticas en las columnas, con los resultados alineados solo resta hacer la selección de una celda y pasar al tercer panel de gráficas.

Este último panel, (ver figura 4.10) cuenta con 3 pestañas en la parte superior:

- *Histograms and TSeries*: La parte superior del panel es el histograma donde en caso de tratarse de métricas como la probabilidad de conexión el eje y representa el número de elementos y en casos como la amplitud de los PSP el eje y representa el número de los pares simpáticamente conectados. En la parte inferior se observan las series de tiempos mostrando el promedio de las respuestas postsinápticas cuando un elemento es seleccionado, y de manera análoga al histograma los resultados son dependientes de la selección del *Matrix Display*, mostrando sólo la pinza de corriente en el caso de la amplitud PSP o el voltaje en otros casos como la amplitud de conexión.

- *Scatter Plots*: En esta segunda pestaña se puede visualizar la información por medio de una gráfica de dispersión, pero a diferencia de la información mostrada en la primera pestaña, se puede seleccionar de manera independiente a la matriz, aquí se puede hacer un contraste entre diferentes datos correlacionados como los son los PSP y PSC.

- *Distance Plots*: La información que se puede obtener desde esta pantalla abarca principalmente la relación entre la probabilidad de conexión y la distancia entre las neuronas.

En conclusión, el *Matrix analyzer* es una herramienta muy dinámica que simplifica la labor de la lectura e interpretación de los experimentos sinápticos llevados a cabo, esto resulta de vital importancia para el contraste.

Es importante destacar que no es una de las alternativas existentes para conocer la información y en casos más especializados podría resultar de mayor valor interaccionar directamente con la API, pero esto conlleva un esfuerzo extra en la especialización de este documento, esfuerzo que para propósito de este trabajo, no resulta necesario.

Con base en el *Matrix analyzer* se hacen una serie de consultas individuales para hacer una captura de datos, en particular es de interés conocer el caso para las neuronas excitatorias, una clasificación que sólo se encuentra disponible para las neuronas de ratón transgénico.

Tras la selección de un grupo de neuronas se puede visualizar su información en el panel de graficado de datos, de esta pantalla se visualizan las siguientes gráficas.

Posterior a la selección de un experimento en particular se puede hacer la revisión en la consola donde se despliega el identificador de la pareja correspondiente a la selección, con este indicador se puede avanzar a la siguiente herramienta *Synaptic dynamics*, en este caso particular se hizo la selección de la neurona que interconecta la capa 4 presináptica y la 2/3 postsináptica, la identificación de la pareja es la:

```
<Pair 1550615153.761 3 2>
```

Con *Synaptic dynamics* se puede visualizar de manera individual cada experimento detallado anteriormente, la vista es similar a la que producía el *spike recorder* del simulador NEST y cada tipo de estímulo puede explorarse de manera individual, al realizar una selección dando *click* en el lateral izquierdo se puede elegir el experimento previamente mencionado (ver figura 4.11).

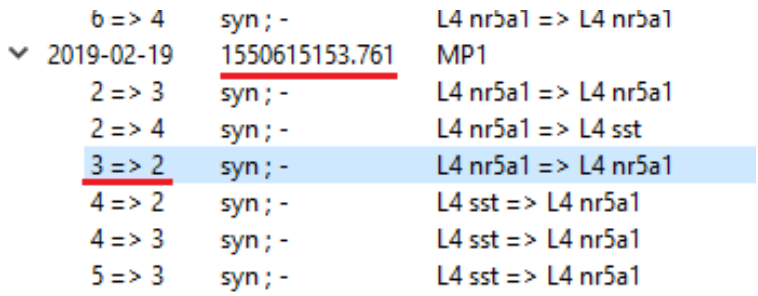


Figura 4.11: En la parte izquierda se puede observar los resultados ordenados por fecha de experimentación

Del lado derecho se encuentran las gráficas con los estímulos, en la parte superior se encuentra el estímulo a la neurona presináptica y en la parte inferior está la reacción de la neurona postsináptica (ver figura 4.12).

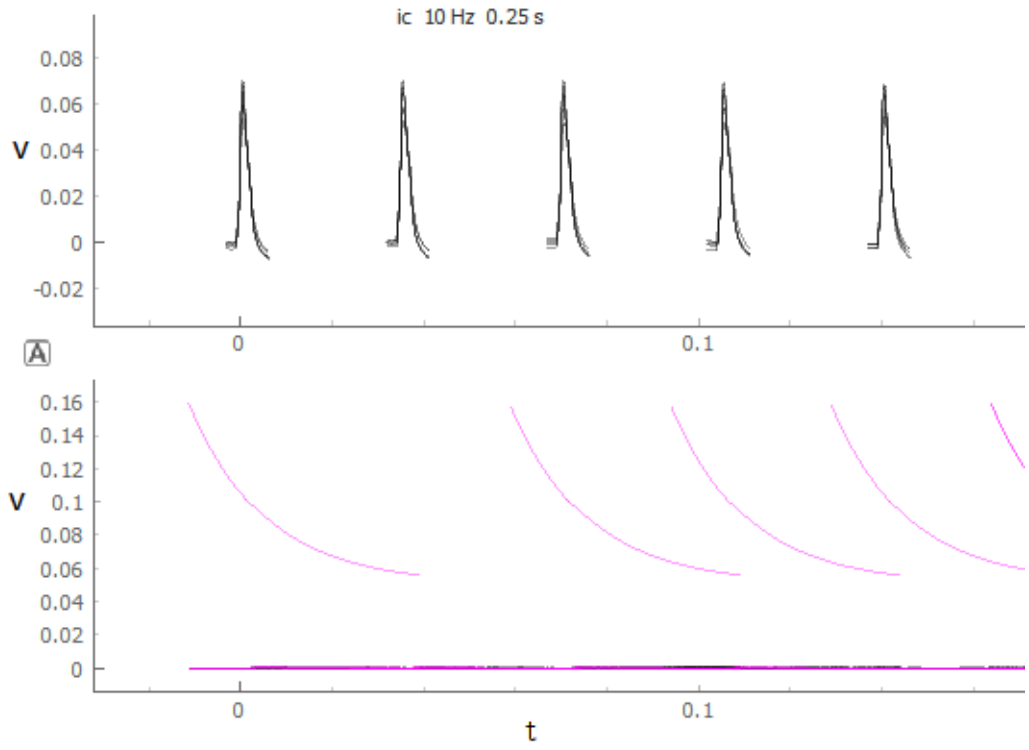


Figura 4.12: En la gráfica se pueden observar los estímulos presinápticos y los estímulos postsinapticos

La información obtenida durante estos experimentos fue posteriormente aprovechada por el Instituto Allen para llevar a cabo la tarea de realizar un clasificación de las neuronas aprovechando técnicas de *Machine learning*, a pesar del apoyo de las herramientas gráficas, resulta bastante complicado obtener información tan clara y limpia como fue con el contraste realizado con las neuronas individuales.

Como conclusión de la información sináptica obtenida con el *Aisynphys*, las neuronas interaccionan a un nivel macro por lo que aislar la interacción de los pares sin tomar en cuenta cómo estas afectan a las demás resulta trivial en cuanto a potenciales de acción, pero a su vez me gustaría rescatar un dato que puede resultar útil a la hora de definir los parámetros de una configuración neuronal simulada en NEST y es la probabilidad que nos regresa el *Matrix Analyzer*, como se observa en la figura 4.13.

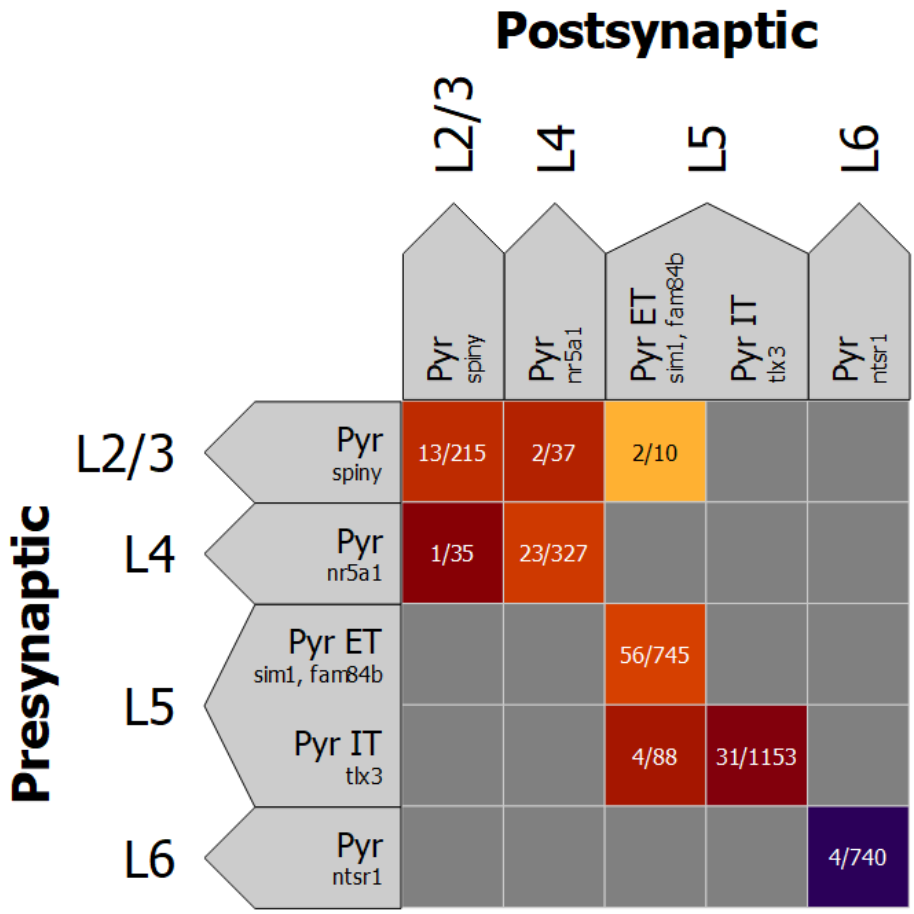


Figura 4.13: En la matriz se puede observar de manera general la probabilidad de conexión entre neuronas de diferentes capas

Con esta información se puede hacer una propuesta de conexión entre neuronas más avanzada que las configuraciones vistas con anterioridad donde se trataba de probabilidades muy artificiales de interacción entre neuronas.

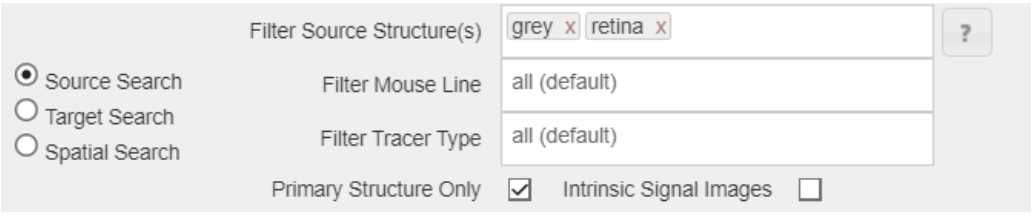


Figura 4.14: La selección de filtros en el AMBCD permite discriminar la información dependiendo de la cantidad de datos sobre el experimento a localizar

Para concluir con la exploración del **Allen Brain Atlas** se seleccionó el AMBCD,

el AMBCD es un mapa tridimensional que despliega la estructura neuronal del cerebro del ratón, por medio de la herramienta web es posible hacer una exploración de la conexión de determinadas zonas neuronales.

Figura 4.15: El AMBCD cuenta con un selector visual donde con una representación virtual del cerebro del ratón se puede seleccionar uno de los experimentos disponibles

El acceso es por medio del portal principal del atlas, al acceder se puede obtener una sección de filtros donde se pueden seleccionar distintas partes del cerebro y localizar los experimentos para cada neurona estimulada.

Tras la selección de los filtros, se pueden visualizar las coincidencias en un panel de resultados, permitiendo la selección gráfica (ver figura 4.15) o por medio de un listado de los diferentes experimentos (ver figura 4.16).

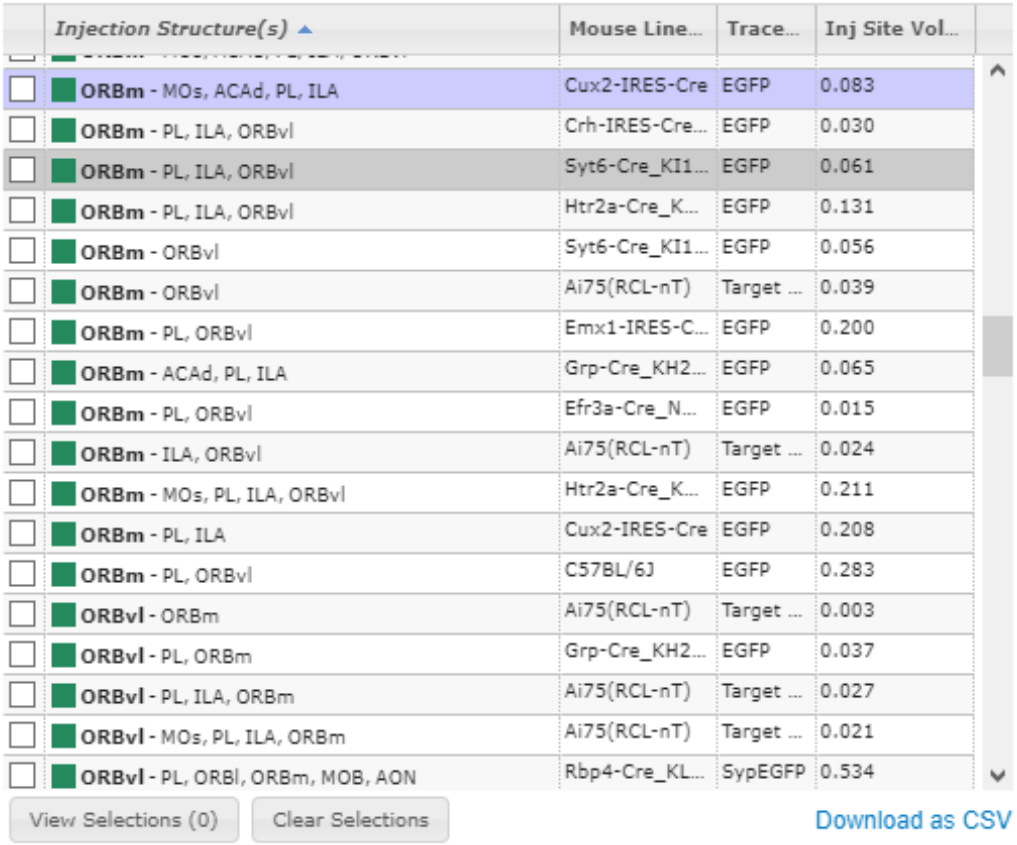


Figura 4.16: El AMBCD cuenta con una tabla de selección, permitiendo obtener la información del experimento con base en las características del área

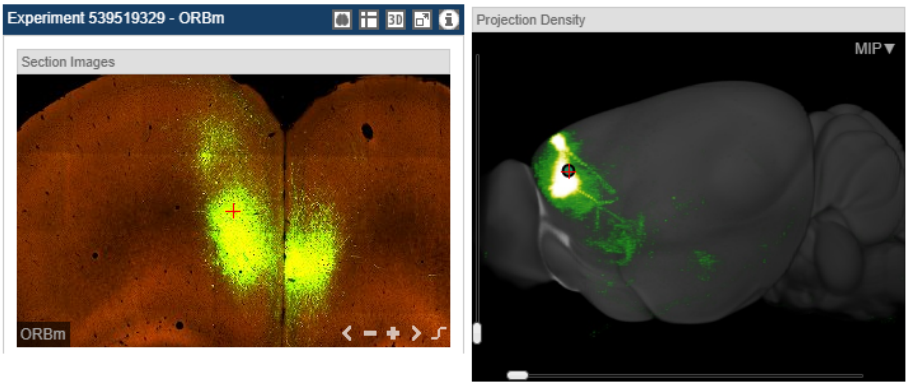


Figura 4.17: Las imágenes muestran las partes afectadas por el estimulo del experimento de los dos fotones, del lado izquierdo se muestra una vista tomográfica y del lado derecho una representación en 3d

Posterior a la selección del experimento, el AMBCD muestra las regiones afectadas del cerebro (ver figura 4.17), demostrando que un pequeño estimulo

neuronal es posible de enviar información a distintas partes de cerebro y aumentando el número de neuronas involucradas en cuestión de milésimas de segundo.

Conociendo el identificador del experimento se puede visualizar la información con más detalle, utilizando la beta de la herramienta de exploración [8], al seleccionar un elemento se muestra un modelo del cerebro del ratón con las neuronas excitadas (ver figura 4.18).

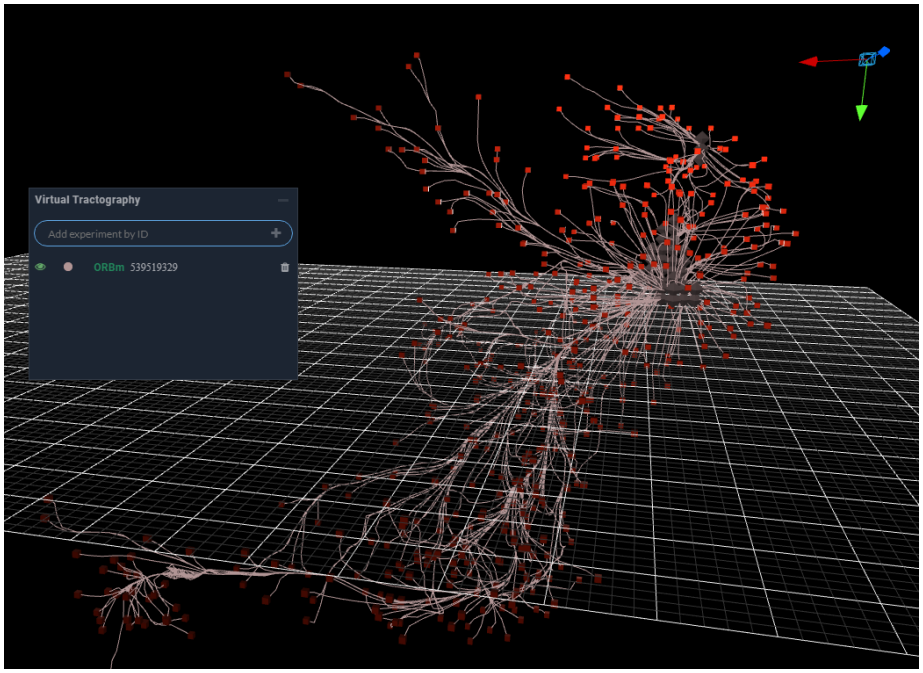


Figura 4.18: Por medio del explorador se puede ver la escala que adopta la exitación de una neurona en el cerebro de un ratón

Esta herramienta desarrollada por *Allen Brain Map* contiene información accesible para cualquier usuario, una representación computarizada del funcionamiento del cerebro de un ratón y permite cuantificar en el orden de cuantas neuronas son excitadas en un momento de tiempo, mostrando ramificaciones y permitiendo explorar secciones del cerebro particulares como es la corteza visual primaria o aquellos experimentos relacionados con la retina.

Finalemente, la existencia de este tipo de herramientas con datos abiertos o herramientas como el simulador NEST facilitan una introducción al trabajo llevado

[8] Allen Brain Explorer https://connectivity.brain-map.org/3d-viewer?v=1&types=STREAMLINE&STREAMLINE=539519329

a cabo en el campo de neurociencia computacional, hay más alcances y posibilidades a trabajar tocando la superficie del área, hay implementaciones que con un grado más alto de especialización podrían ser posibles, el simulador NEST cuenta con la posibilidad de los nodos de manera espacial como con el AMBCD, realizar conexiones con probabilidades con el apoyo del *Aisynphys* y todo esto utilizando modelos como los vistos en el *Allen Cell Types Database*, gracias a este tipo de proyectos un individuo se puede introducir al campo de la neurociencia computacional.

Capítulo 5

Conclusiones

Este trabajo muestra una introducción a la neurociencia computacional, explora las simulaciones y el procesamiento de bancos de datos.

El trabajo toma como eje centra la neurona y su funcionamiento, durante su comunicación en el sistema nervioso. Modelando los medios y la interacción de una neurona.

En este contexto discutimos el modelo matemático desarrollado por Alan Lloyd Hodgkin y Andrew Fielding Huxley. A través de este modelo se explora la interacción neuronal por medio de las matemáticas. Para la simulación, se ha usado una herramienta desarrollada por el instituto NEST, el simulador NEST, finalmente explorando el *Allen Brain Map* del instituto Allen para realizar una comparativa entre los resultados experimentales de neuronas simuladas con neurona reales.

Estos recursos fueron utilizados para explorar la hipótesis:

> En el contexto de la neurociencia computacional, es relevante
> reproducir y comparar una configuración neuronal simulada
> con los resultados experimentales y los modelos teóricos;
> implementando técnicas y conceptos recientes, e incorporando
> nuevas herramientas en los modelos existentes.

Los recursos pueden ser desde modelos teóricos, computacionales y bancos de datos, estos permitieron dar una introducción al área de la neurociencia computacional y como gracias al desarrollo tecnológico, un entusiasta de la

neurociencia puede obtener una gran variedad de información, disponible para su consulta e interacción.

Proyectos como el *Allen Brain Map* y el simulador NEST con la filosofía de *Open Science* ayudan a incentivar la curiosidad y el desarrollo tecnológico en el área de la neurociencia computacional.

Bibliografía

Berg, H. C. (2018). *Random walks in biology*. Princeton University Press.

Carratalá, E., Compta, V., & Dotres, C. (2000). *Larousse enciclopédico universal* (Universal, Vol. 3). Santa Fe de Bogota, Colombia.

Dayan, P., & Abbott, L. F. (2001). *Theoretical neuroscience: computational and mathematical modeling of neural systems*. Computational Neuroscience Series.

Feyissa, A. M., & Tatum, W. O. (2019). Adult EEG. *Handbook of clinical neurology, volumen 160*, 103-124.

Halliday, D., Resnick, R., & Walker, J. (2013). *Fundamentals of physics*. John Wiley & Sons.

Koslow, S. H., & Subramaniam, S. (Eds.). (2005). *Databasing the brain*. Wiley-Blackwell.

Melo Florián, A. (2011). *Cerebro, mente y conciencia*. Internal Medical Publishing.

Ottoson, D. (1983). *Physiology of the nervous system*. Macmillan Press.

Puig, W. R., Torres Isabel, A., & Borjas Caridad, D. (2002). *Morfología humana*. Editorial Ciencias Médicas.

Purve, D., Augustine, G., Fitzpatrick, D., & Hall, W. (2012). *Neuroscience* (3°). Sinauer Associates.

Mesografía

Albin, S. (2019). Stephanie Albin. https://www.nwb.org/2019/02/26/nwbn-2-0-final-released/

Brazier, Y. (2018). Neuroscience: Overview, history, major branches. https://www.medicalnewstoday.com/articles/248680

Commission, E. (2016). Open innovation, open science, open to the world: a vision for Europe.

Dai, K., Hernando, J., Billeh, Y. N., Gratiy, S. L., Planas, J., Davison, A. P., Dura-Bernal, S., Gleeson, P., Devresse, A., Dichter, B. K., Gevaert, M., King, J. G., Van Geit, W. A. H., Povolotsky, A. V., Muller, E., Courcol, J.-D., & Arkhipov, A. (2020). The SONATA data format for efficient description of large-scale network models. *PLOS Computational Biology*, *16*(2), 1-24. https://doi.org/10.1371/journal.pcbi.1007696

de Schepper, R., Eppler, J. M., Kurth, A., Nagendra Babu, P., Deepu, R., Spreizer, S., Trensch, G., Pronold, J., Vennemo, S. B., Graber, S., Morales-Gregorio, A., Linssen, C., Benelhedi, M. A., Mørk, H., Morrison, A., Terhorst, D., Mitchell, J., Diaz, S., Kitayama, I., ... Plesser, H. E. (2022). *NEST 3.2* (Ver. 3.2). Zenodo. https://doi.org/10.5281/zenodo.5886894

DrBob, Z. (2011). Cone-response.svg. https://commons.wikimedia.org/wiki/File:Cone-response-de.svg

Experimental stimuli. (2019). http://portal.brain-map.org/explore/connectivity/synaptic-physiology/synaptic-physiology-experiment-methods/experimental-stimuli

Fee, M., & Zysman, D. (2018). *9.40 Introduction to Neural Computation*. Massachusetts Institute of Technology: MIT OpenCourseWare. https://ocw.mit.edu License: Creative Commons BY-NC-SA

for Brain Science, A. I. (2004). Reference atlas :: Allen Brain Atlas: Mouse brain. https://mouse.brain-map.org

Gewaltig, M.-O., & Diesmann, M. (2007). NEST (Neural Simulation Tool). *Scholarpedia*, *2*(4), 1430.

Gleeson, P. (2015). Hodgkin Huxley Tutorial.

Gouwens, N. W., Sorensen, S. A., Berg, J., Lee, C., Jarsky, T., Ting, J., Sunkin, S. M., Feng, D., Anastassiou, C. A., Barkan, E., Bickley, K., Blesie, N., Braun, T., Brouner, K., Budzillo, A., Caldejon, S., Casper, T., Castelli, D., Chong, P., ... Koch, C. (2019). Classification of electrophysiological and morphological neuron types in the mouse visual cortex. *Nature Neuroscience*, *22*(7), 1182-1195. https://doi.org/10.1038/s41593-019-0417-0

HDF5. (2019). https://www.hdfgroup.org/solutions/hdf5/

How neurons communicate - biology. (2015). https://cnx.org/contents/GFy_h8cu@9.87:cs_Pb-GW@5/How-Neurons-Communicate

Kazilek, C., & Cooper, K. (2010). Rods and Cones. https://askabiologist.asu.edu/rods-and-cones

Liu, N. X. (2020). Can the al/cr ratio alone be used to determine the magnitude and/or progression of myopia? https://reviewofmm.com/can-the-al-cr-ratio-alone-be-used-to-determine-the-magnitude-and-or-progression-of-myopia/

López Lombana, A. d. P., & Hurtado Giraldo, H. (1993). La célula de Schwann. *Biomédica*, *13*(4), 207-217. https://doi.org/10.7705/biomedica.v13i4.2075

Luo, H., Hasegawa, K., Liu, M., & Song, W.-J. (2017). Comparison of the Upper Marginal Neurons of Cortical Layer 2 with Layer 2/3 Pyramidal Neurons in Mouse Temporal Cortex. *Frontiers in Neuroanatomy*, *11*, 115. https://doi.org/10.3389/fnana.2017.00115

NobelPrize.org. (2009). The nobel prize in physiology or medicine 1963. https://www.nobelprize.org/prizes/medicine/1963/summary/

OpenStax. (2016). Cómo se comunican las nuronas:figura3. https://openstax.org/books/biology-2e/pages/35-2-how-neurons-communicate

Philibert, J. (2006). One and a half century of diffusion: Fick, einstein before and beyond.

Quasar. (2011). Neuron. https://es.wikipedia.org/wiki/Neurona%5C#/media/Archivo:Neuron%5C_Hand-tuned.svg

Ramón Romero, F., Mansilla Olivares, A., & Rivera Cruz, A. (2015). Propiedades eléctricas de las membranas. http://www.facmed.unam.mx/Libro-NeuroFisio/

Rod. (2007). https://www.britannica.com/science/rod-retinal-cell

Rodgers, K. (2007). Cone. https://www.britannica.com/science/cone-retinal-cell

Significado de Neurona. (2019). https://www.significados.com/neurona/

Synaptic Physiology. (2019). https://portal.brain-map.org/explore/connectivity/synaptic-physiology

Terakita, A. (2005). The opsins. https://doi.org/10.1186/gb-2005-6-3-213

Wheway, G., Nazlamova, L., Turner, D., & Cross, S. (2019). 661W Photoreceptor Cell Line as a Cell Model for Studying Retinal Ciliopathies. https://doi.org/10.3389/fgene.2019.00308