



BENEMÉRITA UNIVERSIDAD AUTÓNOMA DE PUEBLA

FACULTAD DE CIENCIAS DE LA COMPUTACIÓN

“Simulación computacional de redes neuronales naturales pertenecientes al Generador Central de Patrones de la médula espinal”

TESIS

QUE PARA OBTENER EL GRADO DE:
Licenciado en Ciencias de la Computación

PRESENTA
Imanti Méndez Fontus

DIRECTORES DE TESIS

Dr. Elías Manjarrez López
Laboratorio de Neurofisiología
Integrativa, Instituto de Fisiología

Dr. Manuel Martín Ortíz
Facultad de Ciencias de la
Computación

A la memoria de las mujeres de mi familia:

*Gessie Fontus
Maud Paultre
Alda Bognanni*

A mi familia Luis, Michele, Masai y Aleida.

A Pavel, Cachi, Liz, Dafne y a toda la banda .

A mis revisoras de estilo Susana y Sara.

A los Dres. Manuel Martín y Elías Manjarrez
y a mi revisora Dr. Rosa García Tamayo.

A todos los integrantes del laboratorio de
Neurofisiología Integrativa.

**A todos gracias por sus enseñanzas e incondicional apoyo. Este proyecto
no hubiera sido posible sin la presencia de cada uno de ustedes**

Resumen

En el presente trabajo se llevó a cabo una aplicación capaz de simular el comportamiento de las redes neuronales naturales conocidas como Generador Central de Patrones (CPG por sus siglas en inglés). Estas redes de neuronas, ubicadas a lo largo de la médula espinal, están encargadas de producir una alternancia entre la flexión y la extensión de los músculos de las extremidades en movimientos rítmicos como el rascado y la locomoción. Desde que Graham Brown (Brown, 1911) propuso por primera vez la hipótesis de conexiones recíprocas inhibitorias para explicar el funcionamiento del CPG espinal, se han hecho numerosos avances con el fin de describir a este conjunto de neuronas. En este trabajo en particular se hace uso de la arquitectura de CPG propuesta por Brownston y Wilson (Brownstone & Wilson, 2008) así como la de Unidad Generadora de Ráfagas (UBG por sus siglas en inglés) propuesta por Grillner (Grillner, 2006). La representación de las neuronas de la red CPG se llevó a cabo por medio del modelo neuronal matemático conocido como FitzHugh-Nagumo, gracias al nombre de sus creadores. Las motoneuronas son modeladas haciendo uso de la fórmula de Izhikevich (Izhikevich, 2003), también nombrada así por su creador Eugene Izhikevich. Finalmente, las sinapsis entre neuronas fueron simuladas con la fórmula reducida de conductancia sináptica propuesta por Destexhe (Destexhe, Mainen, & Sejnowski, 1994). En la presente tesis se introduce un programa computacional, que consiste en una completa aplicación capaz de auxiliar a científicos del área de las neurociencias en el proceso de modelado y simulación de las redes CPG. Esta herramienta ofrece una interfaz intuitiva de trabajo, que permite a los usuarios visualizar en tiempo real la estructura de la red, así como calibrar los parámetros de las fórmulas que intervienen en la simulación. Los resultados numéricos muestran la viabilidad del programa para simular de manera muy realista los resultados experimentales obtenidos en el laboratorio, así como para generar nuevas hipótesis acerca del funcionamiento del CPG espinal.

Índice

Resumen.....	3
Introducción.....	6
Capítulo 1: Fundamentos biológicos.....	8
1.1 Sistema nervioso.....	8
1.2 Neurona.....	9
1.3 Sistema motor.....	11
1.4 Generador Central de Patrones.....	11
Capítulo 2: Simulación computacional.....	15
2.1 Simulación en neurociencias.....	16
Trabajos previos.....	17
2.2 Modelos matemáticos de neurona.....	18
Hodgkin-Huxley.....	18
FitzHugh-Nagumo.....	19
Izhikevich.....	19
Conductancia sináptica.....	21
Capítulo 3: Metodología.....	24
3.1 Investigación y recopilación de bibliografía	24
3.2 Requisitos (historias de usuario).....	25
3.3 Análisis.....	27
3.4 Diseño.....	29
Módulo Principal (Core).....	29
Poblaciones neuronales.....	30
Conexiones sinápticas.....	32
Estructuras de datos.....	33
Modelo por compartimientos.....	34
Arquitectura de la red.....	36
Módulo de graficado.....	37
Módulo de graficado 3D.....	38
Módulo de configuración interactiva.....	39
Módulo de administración de proyectos.....	39
3.5 Implementación de módulos.....	40

3.6 Documentación y entrega.....	40
Capítulo 4: Resultados.....	42
4.1 Desempeño del programa.....	42
4.2 Análisis de las señales.....	44
4.3 Caso de uso: usando CPG Network.....	49
Ejecutar CPG Network por primera vez.....	49
El proyecto inicial.....	50
Animando la red CPG.....	51
Conclusiones.....	53
Perspectivas.....	55
Apéndice.....	57
I. Integración numérica para ecuaciones diferenciales ordinarias.....	57
II. Programación Extrema.....	61
Bibliografía.....	63

Introducción

En los últimos años la computación se ha vuelto de vital importancia para el quehacer científico. Cada vez más los centros de investigación invierten y promueven el uso de software que complemente y enriquezca su trabajo. De la misma manera, cada vez más profesionales y empresas del área de tecnologías de la información orientan sus esfuerzos para ofrecer productos y servicios a las disciplinas científicas.

En particular en el área de neurociencias, la demanda de aplicaciones especializadas ha crecido enormemente. En la actualidad se cuenta con un gran número de herramientas, tanto comerciales como de software libre, que apoyan el trabajo científico. Sin embargo, debido al gran poder de cómputo y complejidad que se requieren, muchas de estas herramientas son diseñadas para propósitos muy específicos y presentan un tiempo de vida corto. Es por esto que se vuelve de vital importancia fomentar el crecimiento de proyectos interdisciplinarios y continuos específicos en diversos ambientes de investigación.

El proyecto presentado en esta tesis es el resultado de un año de trabajo en el Laboratorio de Neurofisiología Integrativa del Instituto de Fisiología de la BUAP, en donde, se desarrollaron, entre otras aplicaciones, un simulador de redes Generadoras Centrales de Patrones (CPG por sus siglas en inglés), que ayuda a los investigadores a probar nuevas teorías acerca de la organización de este sistema neuronal.

Las redes CPG están ubicadas en los mamíferos a lo largo de la médula espinal, y controlan movimientos rítmicos como el rascado y la locomoción; en específico en este trabajo se optó por usar el modelo matemático de neurona de FitzHugh-Nagumo e imitar la arquitectura propuesta por Brownston y Wilson (1981) así como la de Unidad Generadora de Ráfagas (UBG por sus siglas en inglés). Por lo tanto, el objetivo principal del trabajo es ofrecer a investigadores del área de neurociencias, una herramienta robusta y versátil que complemente el trabajo de experimentación y modelado de este tipo de redes neuronales.

Los objetivos específicos de la presente tesis son:

- Crear un sistema versátil que ofrezca flexibilidad al momento de crear las estructuras y asignar los parámetros de los diversos niveles del modelado de un CPG.
- Ofrecer herramientas intuitivas que permitan a los usuarios organizar y almacenar los experimentos que se efectúen por medio de este simulador.
- Implementar y comparar diversos métodos de integración numérica con tal de aplicar el que mejor se acople a la simulación propuesta. En específico, implementar el método Runge-Kutta y algunas de sus optimizaciones.
- Incorporar en la medida de lo posible (tiempo) un módulo de simulación gráfica por medio del cual se ejemplifique la organización anatómica del CPG que controla el

movimiento rítmico de la extremidad inferior de gatos.

Para llegar a cumplir estos objetivos se realizó un proceso de ingeniería de software característico de cualquier aplicación computacional funcional. En este caso, se decidió seguir los lineamientos de la metodología conocida como Programación Extrema (XP por sus siglas en inglés); gracias a ella se logró completar una aplicación capaz de simular, de manera realista, la actividad eléctrica, tipo sinusoidal, generada por neuronas del CPG de la médula espinal del gato.

El presente documento, explica de manera detallada las etapas que se cumplieron para llevar a cabo el proyecto. Los primeros dos capítulos plantean el marco teórico. El capítulo 1 hace un breve recorrido conceptual por los mecanismos anatómicos y biológicos necesarios para entender el funcionamiento de las redes CPG. Este capítulo termina con una revisión histórica de algunos de los modelos de redes CPG propuestos. El capítulo 2 introduce a los temas centrales relacionados con la simulación de procesos naturales por medio de una computadora. A su vez, en este capítulo se presentan tres de los más importantes programas de simulación neuronal que existen actualmente en el mercado.

El capítulo 3 hace una revisión, paso por paso, de las etapas que se llevaron durante la ingeniería del programa. Este capítulo muestra las fases comunes que involucran un proceso de ingeniería de software; durante estas fases, al haber usado la metodología conocida como XP, se llevaron a cabo muchas otras subetapas que no son incluidas en este documento.

Posteriormente se presentan los resultados obtenidos con el programa. Se hace una comparación entre señales obtenidas experimentalmente y aquellas generadas por el simulador. Además se muestra una sección de introducción al uso del programa.

Finalmente, se cierra la descripción de este proyecto con una sección de conclusiones (tanto personales como relacionadas con el programa y los resultados) y otra de perspectivas y mejoras.

En el apéndice se encuentran dos secciones: la primera muestra una pequeña introducción a los métodos de integración numérica que se utilizaron para llevar a cabo las simulaciones de los modelos neuronales implementados; la segunda es una breve explicación de los antecedentes y modos de uso de la metodología de software XP usada en el desarrollo de la aplicación.

Capítulo 1: Fundamentos biológicos

1.1 Sistema nervioso

El sistema nervioso es una red de células especializadas que coordina las acciones de un animal y envía señales desde una parte del cuerpo hasta otra. Está compuesto por dos tipos de células, las neuronas y las células gliales o glía. Las neuronas se encargan de coordinar las acciones de los animales por medio de señales químicas y eléctricas enviadas de un extremo al otro del organismo. Las células gliales, por su parte, desempeñan principalmente, la función de soporte y protección de las neuronas (Kandel, Schwartz, & Jessell, 2000).

En la mayoría de los animales, el sistema nervioso está dividido en dos partes: el sistema nervioso central y el sistema nervioso periférico.

El sistema nervioso periférico está compuesto de neuronas sensoriales, agrupaciones de neuronas llamadas ganglios y nervios, craneales y espinales, que emergen del sistema nervioso central y recorren todo el cuerpo.

El sistema nervioso central está formado por el cerebro y la médula espinal. Éste es una estructura esencialmente bilateral y simétrica. En él residen todas las funciones superiores, tanto las cognitivas como las emocionales. Sus partes más importantes son: la médula espinal, bulbo raquídeo, protuberancia, el cerebelo, mesencéfalo, diencéfalo y los hemisferios cerebrales (figura 1.1). Las neuronas en el sistema nervioso central están interconectadas en complejos arreglos, y transmiten señales electroquímicas entre ellas.

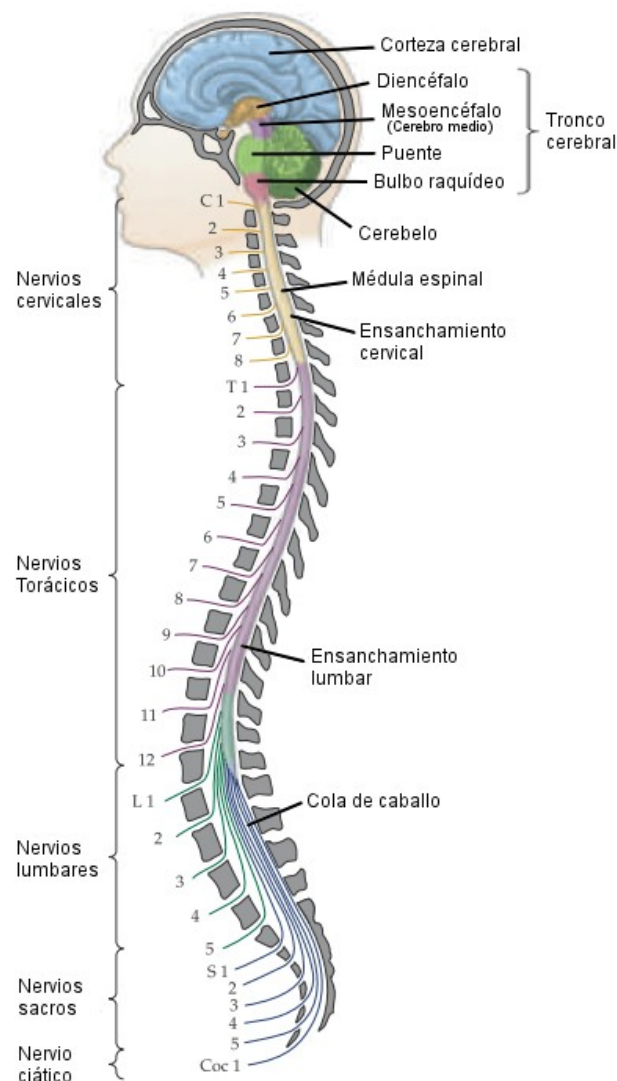


Figura 1.1. Anatomía del sistema nervioso central en el humano. Para su estudio, se divide en siete secciones o subsistemas: médula espinal, bulbo raquídeo, puente o tronco encefálico, cerebelo, mesencéfalo, diencefalo y hemisferios cerebrales. Imagen modificada del libro *Neuroscience* (Purves & Williams, 2001).

1.2 Neurona

Las neuronas son las unidades elementales del sistema nervioso. Su principal característica es la excitabilidad de su membrana plasmática; están especializadas en la recepción de estímulos y conducción del impulso nervioso (en forma de potencial de acción) entre ellas o con otros tipos celulares como, por ejemplo, las fibras musculares de la placa motora.

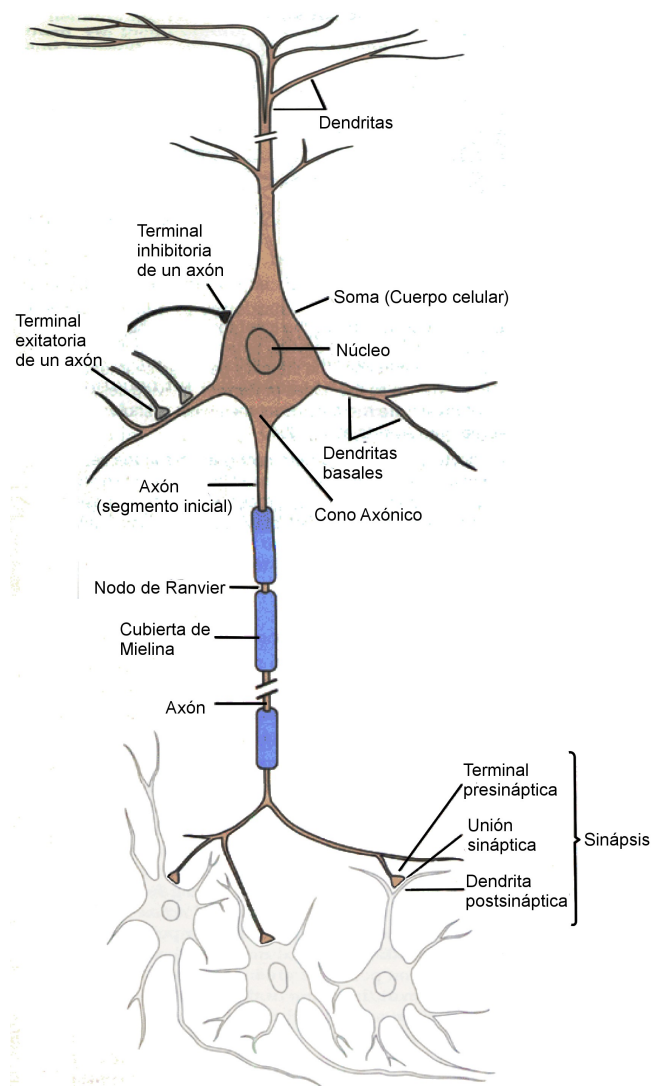


Figura 1.2. Anatomía de la neurona. Una neurona tiene cuatro regiones morfológicas: soma, dendritas, axón y terminales sinápticas. El axón, recubierto por mielina, se extiende alejándose del soma hasta las terminales sinápticas. Un potencial de acción inicia en el cono axónico y viaja a través del axón regenerándose en cada nodo de Ranvier, hasta alcanzar alguna de estas terminales. Imagen modificada del libro *Principles of Neural Science* (Kandel, Schwartz, & Jessell, 2000).

Las neuronas tienen cuatro regiones morfológicas definidas: el cuerpo celular, las dendritas, el axón y las terminales presinápticas. El cuerpo celular (soma) es el centro metabólico de la célula; en él se encuentra, entre otras cosas, el núcleo celular. Del soma nacen dos tipos de procesos: numerosas dendritas y un largo axón tubular. Las dendritas se alargan en forma de ramas de un árbol y constituyen el aparato central para la recepción de señales provenientes de otras neuronas. En contraste, el axón se extiende alejándose del soma y es la unidad conductora principal para llevar señales hacia otras neuronas (Kandel, Schwartz, & Jessell, 2000).

Las neuronas reciben, analizan e integran información por medio de señales eléctricas. Estas señales, llamadas potenciales de acción, son impulsos nerviosos rápidos y transitorios, con una amplitud promedio de 100 mV y una duración de 1 ms, que viajan a

través del axón hasta una terminal presináptica. Los potenciales de acción son iniciados en una región especializada del axón llamada cono axónico. La amplitud del potencial de acción que viaja por el axón se mantiene constante gracias a que éste es regenerado en intervalos regulares en las zonas conocidas como nodos de Ranvier. Para aumentar la velocidad a la que los potenciales de acción son conducidos, los axones son envueltos en una cubierta aislante de mielina (Keynes & Aidley, 2001).

1.3 Sistema motor

Los movimientos, tanto voluntarios como involuntarios, son producidos por patrones espacio-temporales de contracciones musculares dirigidas por el cerebro y la médula espinal. Son muchos los circuitos neuronales que intervienen en la creación y regulación de estos movimientos. Para su estudio, el sistema motor suele dividirse en tres niveles o subsistemas principales: la médula espinal, el tronco cerebral y la corteza cerebral (Kandel, Schwartz, & Jessell, 2000).

El primero de estos subsistemas consiste en la circuitería local ubicada en la materia gris de la médula espinal. Las neuronas más relevantes de este subsistema son las llamadas motoneuronas (que envían sus axones desde la médula espinal hasta las fibras musculares) y las interneuronas (que son la más importante fuente de entradas sinápticas para las motoneuronas). Todos los comandos de movimiento, ya sea reflejo o voluntario, convergen hacia los músculos por medio de las neuronas motoras; como lo postuló el neurofisiólogo inglés Charles Sherrington, éstas son “la vía final común” para el movimiento (Purves & Williams, 2001).

El siguiente nivel es el tronco cerebral; en él se reciben entradas de la corteza cerebral y los núcleos subcorticales y se proyectan hacia la médula espinal. Este subsistema contribuye en tareas como el control de la postura o los movimientos dirigidos de brazos y manos.

El nivel superior está ubicado en la corteza cerebral. Tanto la corteza motora como otras áreas premotoras se conectan a la médula espinal por medio del tracto cortico-espinal. Estas vías de motoneuronas, que comienzan en la corteza cerebral, son esenciales para la iniciación de movimientos voluntarios y para la coordinación de secuencias de movimientos complejos.

1.4 Generador Central de Patrones

De los estudios hechos por Graham Brown a inicios del siglo pasado, hoy se sabe que la médula espinal aislada es capaz de generar descargas rítmicas de actividad recíproca entre neuronas motoras flexoras y extensoras en las extremidades. Graham Brown propuso que las contracciones en los músculos flexores y extensores se controlan por dos sistemas, a los cuales llamó hemicentros y se inhiben mutuamente. En la actualidad, a estas redes neuronales ubicadas en la médula espinal y encargadas de generar una

salida rítmica hacia las motoneuronas que inervan a los músculos esqueléticos, se les conoce como Generadores Centrales de Patrones (CPG por sus siglas en inglés), y existen diversas arquitecturas que intentan explicar los complejos patrones que se han observado experimentalmente.

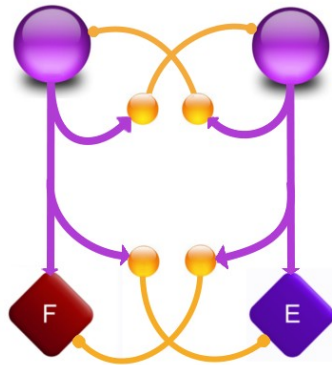


Figura 1.3. Modelo propuesto por Graham Brown. En morado, tenemos la interneuronas que forman parte del hemicentro; en naranja, las interneuronas que median la mutua inhibición. En rojo y azul se ilustran las motoneuronas flexoras y extensoras respectivamente. Texto e imagen modificados de la tesis del M.C. Jesús Ángel Tapia López (Tapia-López, 2008).

El primer modelo de CPG, propuesto por Graham Brown a principios del siglo pasado, es el denominado modelo de hemicentros (figura 1.3). Los aspectos más importantes de este modelo son: 1) Cada miembro es controlado por un CPG individual. 2) Cada CPG contiene dos grupos de interneuronas excitadoras que controlan la actividad de las motoneuronas flexoras y extensoras. 3) Las conexiones mutuamente inhibitorias entre los dos hemicentros asegura que sólo un centro puede estar activo a la vez. 4) Un proceso indefinido de “fatiga” reduce gradualmente la excitación en el centro activo. 5) El cambio de fase (entre hemicentros) ocurre cuando la reducción en la excitabilidad de uno de los centros decrece por debajo de un valor crítico, y el centro opuesto es liberado de su inhibición. 6) La inhibición entre motoneuronas antagonistas está fuertemente ligada a la excitación de su centro opuesto (McCrea & Rybak, 2008).

El modelo del hemicentro describe la alternancia entre músculos flexores y extensores, sin embargo, se han encontrado diversos patrones que este modelo es incapaz de explicar. Por esta razón, durante la segunda mitad del siglo XX y principios del XXI, muchos neurocientíficos han propuesto una gran variedad de modelos que son capaces de generar patrones más complejos.

Uno de los patrones más complejos que se han encontrado son las llamadas “ausencias” (*deletions*, en inglés). Durante un episodio de movimientos rítmicos estereotipados pueden ocurrir fallas espontáneas (ausencias) en la descarga de las motoneuronas. Las fallas pueden clasificarse en dos grupos: reiniciadoras y no reiniciadoras. Al primer conjunto pertenecen aquellas fallas en las cuales el periodo es afectado y se reinicia el ciclo de actividad alternada. Mientras que las fallas del segundo grupo mantienen un periodo constante.

Para explicar estas ausencias se han propuesto modelos en los cuales la generación del ritmo recae en una población de interneuronas independientes que es responsable de

dar forma a los patrones de la descarga. De este modo, los errores en la transmisión sináptica del formador de patrones (PF por sus siglas en inglés), no afectarían al generador de ritmo (RG por sus siglas en inglés). De esta forma, el RG podría mantener la actividad rítmica y la actividad de los músculos aun cuando se presenten ausencias de actividad en las motoneuronas extensoras (Tapia-López, 2008).

Uno de los modelos de CPG diseñados para poder explicar estas ausencias es el propuesto por Brownston y Wilson (2008) (figura 1.4). En este modelo se propone que el RG, en lugar de poseer una estructura de hemicentros, está compuesto de una red heterogénea de neuronas conectadas eléctricamente que presentan propiedades intrínsecas de oscilación (neuronas Hb9). Esta asimetría que presenta el RG se extiende al PF mediante una inhibición entre la salida del RG y el hemicentro flexor.

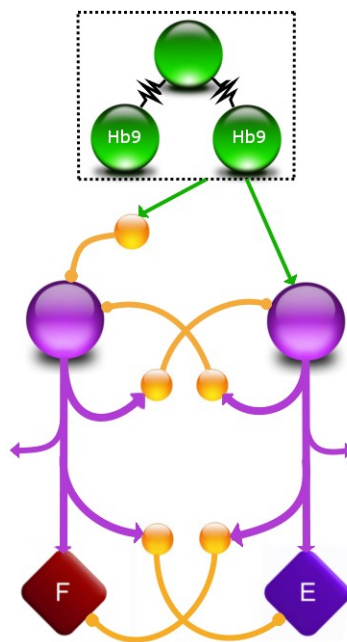


Figura 1.4. Modelo asimétrico propuesto por Brownston y Wilson (2008). El generador de ritmo consta de una red de neuronas conectadas por enlaces eléctricos que les permite sincronizarse y oscilar. El hemicentro flexor está controlado disinápticamente mediante una neurona inhibitoria para explicar la actividad rítmica en músculos flexores, sin que su antagonista descargue ráfaga. Texto e imagen modificados de la tesis del M.C. Jesús Ángel Tapia López (Tapia-López, 2008).

Otro modelo que intenta explicar los complejos patrones generados durante episodios de movimientos rítmicos estereotipados es el conocido como unidad generadora de ráfagas (UBG por sus siglas en inglés) propuesto por Grillner (1981) (figura 1.5). Por medio de diversos experimentos, Grillner demostró que existe un gradiente de excitabilidad para el generador de patrones de la locomoción; es decir, segmentos más rostrales (L4-L5) generan una actividad alternada más estable que segmentos caudales. Por otra parte, observó que hay un gradiente de propagación en la salida motora; más especialmente, raíces ventrales rostrales tienen descargas, antes que las descargas de raíces ventrales localizadas caudalmente (Tapia-López, 2008).

Por lo tanto, este modelo consiste en múltiples osciladores acoplados, donde cada unidad presenta una organización en hemicentro. La actividad entre estos módulos estaría regulada por conexiones sinápticas excitatorias e inhibitorias.

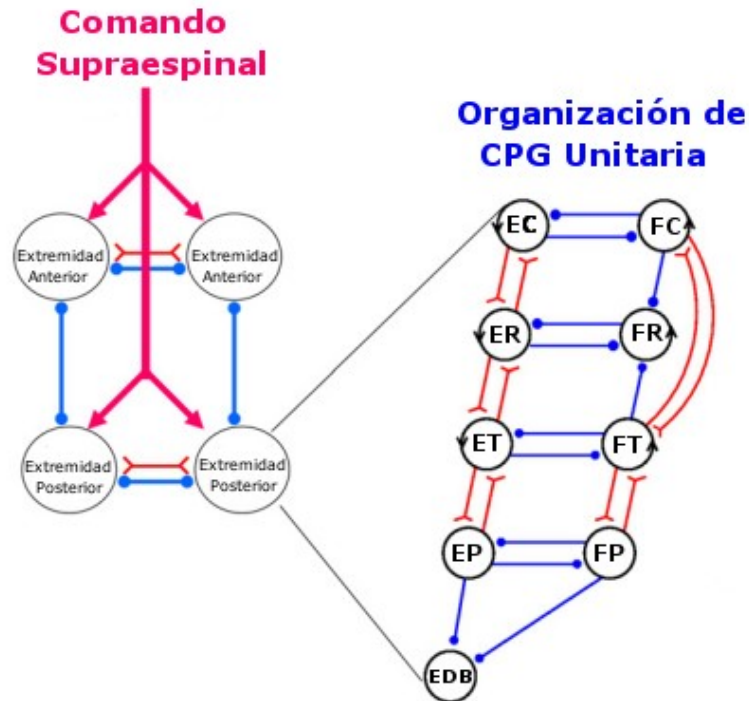


Figura 1.5. Modelo del UBG (Unidad Generadora de Ráfagas) propuesto por Grillner. A la izquierda, las regiones de comando en el tronco cerebral controlan los niveles de actividad de las redes CPG para cada uno de los miembros. A la derecha, cada CPG unitario controla una articulación. FC y EC representan el control de la flexión y extensión de la cadera respectivamente, lo mismo para la rodilla (FR, ER), tobillo (FT, ET) y pie (FP, EP). Un caso especial es el del EDB, ya que presenta un patrón diferente. Texto e imagen modificados de “Biological Pattern Generation: The Cellular and Computational Logic of Networks in Motion” (Grillner, 2006).

Capítulo 2: Simulación computacional

La simulación, a grandes rasgos, es el proceso de diseñar un modelo, basado en un sistema real o imaginario, y llevar a cabo experimentos con ese modelo. Es usada en prácticamente todas las disciplinas relacionadas con ingeniería, ciencia y tecnología. Desde su definición formal, en la década de los cincuenta, la simulación ha sido adaptada a una gran variedad de aplicaciones (Smith, 1999). Dependiendo del área en la cual se trabaje, sus definiciones pueden variar sustancialmente. A continuación, se muestran tres definiciones que ayudan a conceptualizar de mejor manera el término simulación.

El modelado y análisis por simulación es el proceso de recrear un sistema real por medio de un modelo matemático computarizado (Chung, 2003).

La simulación es la generación de pseudodatos partiendo de un modelo o base de datos (Thompson, 1999).

Simulación es la imitación del comportamiento de un proceso (sistema) del mundo real a través del tiempo. Simular implica generar un historial artificial del sistema, por medio del cual, se pueden inferir características operacionales del mismo [...] La simulación es usada para describir y analizar el comportamiento de los sistemas, generar preguntas sobre nuevos estados de un sistema, y ayudar al diseño de nuevos sistemas reales. (Banks, 1998).

Uno de los pioneros en el área de simulación fue John von Neumann. A finales de la década de los cuarenta, él concibió la idea de ejecutar múltiples repeticiones de un modelo, almacenar información estadística y predecir comportamientos del sistema real mediante este modelo. A esto en la actualidad se le conoce como el método de Monte Carlo, en el cual se usan variaciones aleatorias para representar comportamientos que no podrían ser modelados con exactitud, pero sí a través de caracterizaciones estadísticas (Woolfson & Pert, 1999).

Existen distintos tipos de clasificación de los sistemas de simulación. Una de las más importantes es la que hace referencia a la aleatoriedad de un sistema; y propone dos clases de simuladores: determinísticos y estocásticos. Los modelos determinísticos son aquellos que no contienen elementos de probabilidad mientras que los modelos estocásticos dependen de elementos aleatorios (Perros, 2008).

Otro tipo de clasificación hace la separación entre simulaciones con eventos discretos o continuos, tomando en cuenta la manera en que las variables de estado cambian. Los eventos discretos contienen variables que cambian instantáneamente en distintos puntos en el tiempo. Contrariamente, en las simulaciones continuas, las variables cambian permanentemente mediante una función en donde el tiempo es una de las variables (Smith, 1999).

2.1 Simulación en neurociencias

La simulación en neurociencias es una parte del área conocida como neurociencias computacionales. Esta área se enfoca al estudio de las funciones del cerebro desde el punto de vista del procesamiento de información dentro de las estructuras que componen al sistema nervioso. A su vez, es un área que vincula diversos campos científicos como neurociencias, ciencias cognoscitivas, ingeniería eléctrica, ciencias de la computación, física y matemáticas (Churchland, Koch, & Sejnowski, 1993).

En su artículo “Computational Neuroscience: Scope and Structure” (Perkel, 1993), Donald Perkel explica una forma de clasificar a las aplicaciones de las neurociencias computacionales según distintos niveles de análisis de las neurociencias. En esta clasificación él encuentra 8 categorías principales:

1. El nivel básico trabaja con la biofísica y bioquímica de la membrana neuronal y el citoplasma que la rodea. En este nivel se estudian fenómenos como cinética de los canales iónicos, dinámica de receptores o síntesis y captación de neurotransmisores, entre otras.
2. El siguiente nivel estudia subunidades funcionales de las células nerviosas, como la sinapsis o los árboles dendríticos. En este segundo nivel se pueden incluir también estudios sobre las propiedades de excitabilidad de la membrana neuronal.
3. El tercer nivel trabaja con la célula completa. Está relacionado no sólo con la dinámica funcional de la célula sino también con otros fenómenos como su actividad metabólica o su morfología.
4. El cuarto nivel lleva a cabo pequeños circuitos de células nerviosas. En este nivel se pueden llevar a cabo estudios sobre la inhibición recíproca y lateral, los reflejos o la generación de patrones motores.
5. El quinto nivel trabaja redes de mediana y larga escala. Para este punto se requieren simplificaciones significativas en la descripción de la neurona original. Los estudios en este nivel se relacionan con áreas como la corteza cerebral o la corteza visual.
6. El siguiente nivel está dedicado a estudiar sistemas funcionales completos o porciones del cerebro definidas anatómicamente. En este sexto nivel llega a ser imposible representar neuronas o conexiones sinápticas individualmente, en lugar de esto se hace uso de técnicas como programación por bloques.
7. Los niveles 7 y 8 se han explorado mínimamente en proyectos reales. El séptimo nivel implica trabajar con todo el cerebro o incluso con el sistema nervioso central. El octavo incluiría también los sistemas sensorial y motor.

Desde que Perkel presentó esta clasificación, la computación ha sufrido muchos avances con respecto a nuevas y más eficientes tecnologías. Es de este modo que programas como los mostrados en la siguiente sección e, incluso, el que se presenta en este trabajo, son capaces de trabajar en el quinto nivel de esta clasificación, utilizando modelos de neurona considerablemente complejos.

Para la representación de redes neuronales por computadora existen dos familias principales de algoritmos: los síncronos y los asíncronos. En el caso de los algoritmos síncronos, el estado de las neuronas de la red es actualizado periódicamente, lo cual permite la aplicación de muchos tipos de modelos diferentes, sin requerir la modificación del algoritmo. Contrariamente, los algoritmos asíncronos actualizan únicamente el estado de las neuronas que evidencien un cambio significativo (producir o recibir un potencial) en cada tiempo de la simulación (Brette et al., 2007). Para la implementación del programa presentado en este trabajo se utilizaron algoritmos síncronos gracias a la necesidad de desplegar los resultados dependientes del tiempo.

Trabajos previos

Desde finales de la década de los ochenta varios laboratorios alrededor del mundo incursionaron en el área de simulación en neurociencias. Muchos de estos proyectos han tenido tiempos de vida cortos, acabando únicamente como referencias históricas para gente que desea incursionar en este campo; sin embargo, muchos otros de estos sistemas han perduran y, en la actualidad, son invaluable herramientas para neurocientíficos. A continuación se listan los más importantes sistemas de simulación neuronal, conocidos y usados en muchos laboratorios alrededor del mundo. Estos datos fueron extraídos del artículo “Simulation of networks of spiking neurons: A review of tools and strategies” (Brette et al., 2007), el cual representa una importante revisión de los avances en simulación neuronal en los últimos años.

- NEURON: Es un ambiente de simulación que permite crear y usar modelos empíricos de neuronas biológicas y circuitos neuronales. Inicialmente ganó reputación por ser un ambiente adecuado para la simulación de modelos de células basados en “conductancias”. Sin embargo, actualmente se han publicado cientos de artículos y libros en donde se simulan redes neuronales completas, con miles de neuronas, basadas en “conductancia”, o en el concepto de “integración y disparo”. Uno de los aspectos más importantes de NEURON es la variedad de herramientas de control conceptual que brinda; es decir, ofrece una gran similitud con conceptos biológicos, que ayudan al fácil entendimiento del programa por parte de científicos con poco o nulo conocimiento en el área de simulación computacional. NEURON tiene también tres modos distintos para ser ejecutado en ambientes paralelos y se encuentra accesible a todo público, sin costo para sistemas operativos Unix/Linux, OS X y Microsoft Windows. Para mayor información consultar la página <http://www.neuron.yale.edu/neuron/>.
- GENESIS (GEneral NEural Simulation System): Fue diseñado para ser un sistema extensible para la simulación de modelos realistas de neuronas y sistemas biológicos. Al hablar de modelos realistas, los autores se refieren a aquellos modelos que están basados en la organización anatómica y fisiológica de neuronas, circuitos y redes. Con la extensión PGENESIS (Parallel GENESIS) es posible ejecutar este simulador en casi cualquier cluster paralelo, SMP, supercomputadora o red de estaciones de trabajo que soporten los protocolos MPI (Message Passing Interface) y/o PVM (Parallel Virtual Machine). El sistema puede ser adquirido sin costo para sistemas operativos Unix/Linux, OS X y Microsoft Windows. Para mayor información consultar la página <http://genesis-sim.org/>.

- NEST (Neural Simulation Technology): Es una iniciativa fundada en el año 2001 que tiene como propuesta colaborar en el desarrollo de métodos de simulación para redes neuronales biológicamente realistas. Sus objetivos principales son: 1) El desarrollo de nuevos métodos y algoritmos para simulación. 2) El desarrollo de nuevas herramientas de análisis y visualización. 3) La creación de colecciones de información y recursos relacionados con la simulación neuronal. El sistema desarrollado por esta iniciativa está diseñado para ser altamente paralelo y puede ser adquirido sin costo para sistemas operativos Unix/Linux, OS X y Microsoft Windows. Para mayor información consultar la página <http://www.nest-initiative.org/index.php>.

2.2 Modelos matemáticos de neurona

El primer modelo matemático de neurona fue propuesto en 1907 por el neurocientífico francés Louis Lapicque (Abbott, 1999). En la actualidad, este modelo es conocido como “integración y disparo”, y es en realidad el origen de una familia entera de modelos bajo el mismo nombre. Desde entonces se han realizado grandes avances en el estudio de los mecanismos responsables de la generación de potenciales de acción neuronales, y se han propuesto varios modelos matemáticos para representar estos mecanismos.

Hodgkin-Huxley

En el año de 1952, Alan Lloyd Hodgkin y Andrew Huxley describieron un modelo matemático para explicar los mecanismos iónicos relacionados con la iniciación y propagación de los potenciales de acción en el axón gigante de calamar. Por este trabajo, ganaron el premio Nobel (1953) de fisiología y medicina.

Al usar innovadoras técnicas experimentales, obtuvieron el siguiente sistema de ecuaciones

$$\begin{aligned}
 C \frac{dV}{dt} &= I - \overbrace{\bar{g}_K n^4 (V - E_K)}^{I_K} - \overbrace{\bar{g}_{Na} m^3 h (V - E_{Na})}^{I_{Na}} - g_l (V - E_l) \\
 \frac{dn}{dt} &= \alpha_n(V)(1-n) - \beta_n(V)n \\
 \frac{dm}{dt} &= \alpha_m(V)(1-m) - \beta_m(V)m \\
 \frac{dh}{dt} &= \alpha_h(V)(1-h) - \beta_h(V)h
 \end{aligned}
 \tag{2.a}$$

en donde los términos etiquetados como E_K y E_{Na} representan los potenciales de inversión para el potasio (K) y el sodio (Na), y las funciones $\alpha(V)$ y $\beta(V)$ representan

las transiciones de apertura y cierre de dichos canales. Los términos I_K y I_{Na} en conjunto representan las corrientes totales de potasio y sodio respectivamente.

Como se observa en el sistema de ecuaciones anterior, la membrana del axón gigante de calamar posee dos corrientes principales de iones: Na y K . Muchas de las células en el sistema nervioso central poseen corrientes adicionales, con diversas dinámicas de activación y desactivación. A pesar de esto, el formalismo propuesto por Hodgkin y Huxley es el modelo mayormente aceptado para describir la cinética de las neuronas (Izhikevich, 2006).

FitzHugh-Nagumo

Poco tiempo después de que el modelo de Hodgkin y Huxley fuese publicado, Richard FitzHugh e, independientemente, Nagumo Arimoto, iniciaron un análisis de las propiedades matemáticas de estas ecuaciones. Ambos propusieron un sistema bidimensional, el cual es una simplificación útil de las ecuaciones de Hodgkin-Huxley.

El modelo de FitzHugh-Nagumo está representado por el siguiente sistema de ecuaciones

$$\frac{dV}{dt} = f(V) - W + I \quad 2.b$$

$$\frac{dW}{dt} = \phi(a + bV + cW). \quad 2.c$$

en donde V representa el potencial de membrana y W es conocida como la variable de recuperación. La ecuación 2.b muestra el modo más genérico de representar este modelo; por lo general, la función $f(V)$ es sustituida con una función parabólica de la forma

$$f(V) = \frac{V^3}{3}. \quad 2.d$$

Los parámetros ϕ , a , b y c de la ecuación 2.c están relacionados con la cinética de la variable de recuperación (Tapia-López, 2006).

Izhikevich

El modelo propuesto por Eugene M. Izhikevich (Izhikevich, 2003) es uno de los tantos conocidos como modelos de integración y disparo. La innovación que representa este modelo está relacionada con el modo en el que Izhikevich lo diseñó, haciendo uso de la teoría de bifurcaciones. En lugar de trabajar simplificando un modelo existente, él empezó por observar que distintas dinámicas de bifurcación representan transiciones entre momentos de los modelos neuronales, como pueden ser la activación tónica, los trenes

de potenciales de acción o la recuperación post inhibitoria. Mapeando diferentes tipos de bifurcaciones con estos mecanismos neuronales, logró encontrar un modelo muy simple que, al darle los parámetros adecuados, puede reproducir la mayoría de los patrones de disparo conocidos para una neurona individual. Gracias a que este modelo no fue construido a partir de algún mecanismo biológico, puede ser difícil relacionar sus parámetros con componentes neuronales, sin embargo, para simulación de redes completas ofrece una gran ventaja en términos de cantidad de procesamiento. Este modelo está definido por las siguientes ecuaciones:

$$\frac{dv}{dt} = 0.04v^2 + 5v + 140 - u - I \quad 2.e$$

$$\frac{du}{dt} = a(bv - u) \quad 2.f$$

Adicionalmente, se usa la siguiente “regla de reseteo”:

$$\begin{aligned} &\text{si } v=30 \text{ entonces} \\ &v \leftarrow c \text{ y } u \leftarrow u + d \end{aligned} \quad 2.g$$

Al igual que en el modelo de FitzHugh-Nagumo, la variable v es el potencial de membrana, la variable u es conocida como la variable de recuperación y la variable I representa la corriente externa. Las demás variables en las ecuaciones son parámetros para el modelo y se explican a continuación:

- El parámetro a describe una escala de tiempo para la variable de recuperación. Mientras menor sea su valor mayor será el tiempo de recuperación después de un potencial de acción.
- El parámetro b describe la sensibilidad de la variable de recuperación a las fluctuaciones subumbral del potencial de membrana. Mientras mayor sea su valor hay mayor probabilidad de observar oscilaciones subumbral y umbrales de disparo más pequeños.
- El parámetro c representa el valor de reseteo para potencial de membrana después de un potencial de acción.
- El parámetro d representa el valor de reseteo para la variable de recuperación después de un potencial de acción.

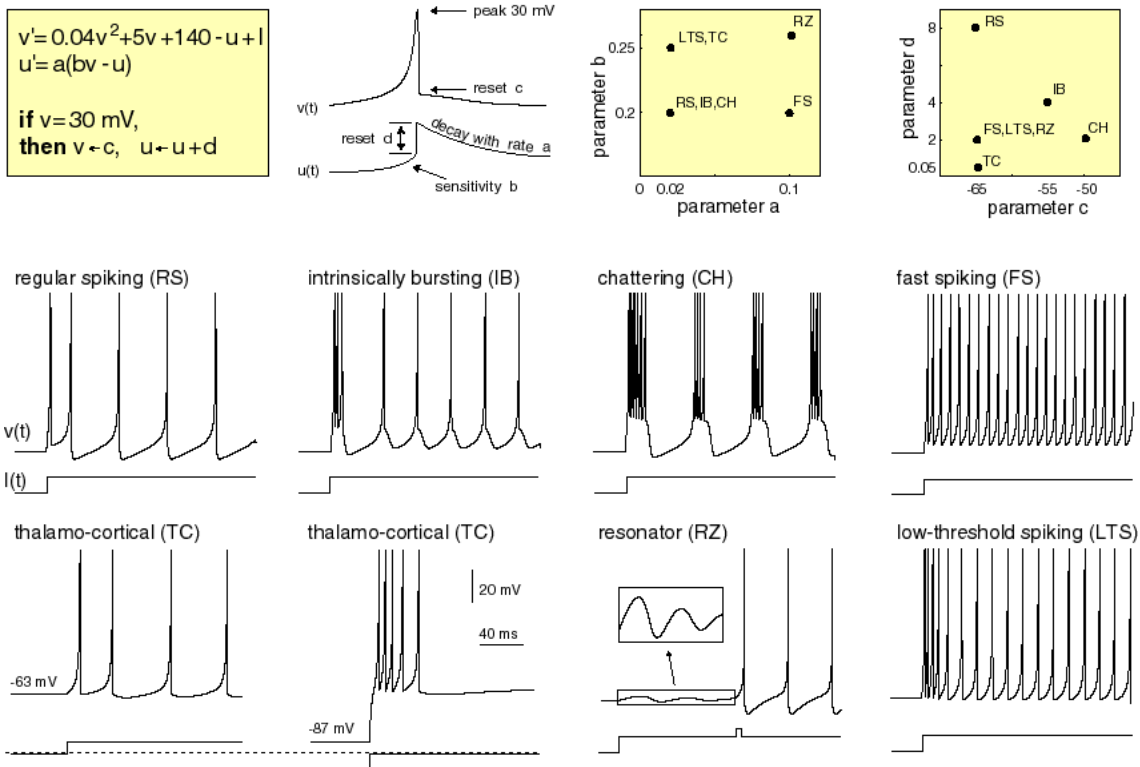


Figura 2.1. El modelo de izhikevich es capaz de imitar el comportamiento de distintos tipos de neuronas modificando sus parámetros. En la simulación presentada en este trabajo se hace uso de neuronas de tipo chattering (CH). [Una versión electrónica de la figura y sus permisos de reproducción pueden ser consultados bajo ningún costo en www.izhikevich.com].

Conductancia sináptica

Uno de los métodos más utilizados para representar eventos sinápticos es calculando la conductancia sináptica directamente del modelo cinético de enlace de receptores (Perkel, Mulloney, & Budelli, 1981). Este modelo hace una relación de la cantidad de moléculas neurotransmisoras que libera la terminal presináptica, y la cantidad de estas moléculas que son ligadas con receptores postsinápticos. La siguiente formula cinética de primer orden ejemplifica este modelo:



donde R y TR son los receptores postsinápticos sueltos y ligados respectivamente, α y β son constantes que indican la tasa en la cual los neurotransmisores se ligan y quedan sueltos. Siendo r la fracción de neurotransmisores ligado, esta cinética puede ser definida por la ecuación

$$\frac{dr}{dt} = \alpha [T](1-r) - \beta r \quad 2.i$$

donde $[T]$ es la concentración del neurotransmisor.

Un método reducido para la implementación de este modelo, es el propuesto por Destexhe y colaboradores (Destexhe, Mainen, & Sejnowski, 1994). En él, este grupo de investigación hace uso de evidencia experimental que sugiere que $[T]$ se incrementa y decrecienta muy rápidamente, esto les da la posibilidad de interpretar a $[T]$ como un pulso de un solo tiempo y los lleva a deducir la siguiente fórmula:

1. Durante un pulso ($t_0 < t < t_1$), $[T] = T_{max}$ y r es calculada por

$$r(t-t_0) = r_\infty + [r(t_0) - r_\infty] * e^{-(t-t_0)/\tau_r} \quad 2.j$$

donde

$$r_\infty = \frac{\alpha T_{max}}{\alpha T_{max} + \beta} \quad 2.k$$

y

$$\tau_r = \frac{1}{\alpha T_{max} + \beta} \quad 2.l$$

2. Después de un pulso $t > t_1$, $[T] = 0$ y r es calculada por

$$r(t-t_1) = r(t_1) e^{-\beta(t-t_1)} \quad 2.m$$

Si la unión de neurotransmisores a los receptores postsinápticos abre directamente los canales asociados a un ion, entonces la conductancia total de los canales de la sinapsis es r multiplicada por el valor máximo de conductancia de la sinapsis, $\bar{g}_{syn}$. Entonces, la corriente sináptica, I_{syn} , es calculada por

$$I_{syn}(t) = \bar{g}_{syn} r(t) [V_{syn} - E_{syn}] \quad 2.n$$

donde V_{syn} es el potencial postsináptico y E_{syn} es el potencial de Nernst.

Para ilustrar de mejor manera el funcionamiento de los modelos neuronales junto con el de conexión sináptica, en la figura 2.2 se muestra un diagrama que explica la propagación de la corriente desde una neurona hasta otra.

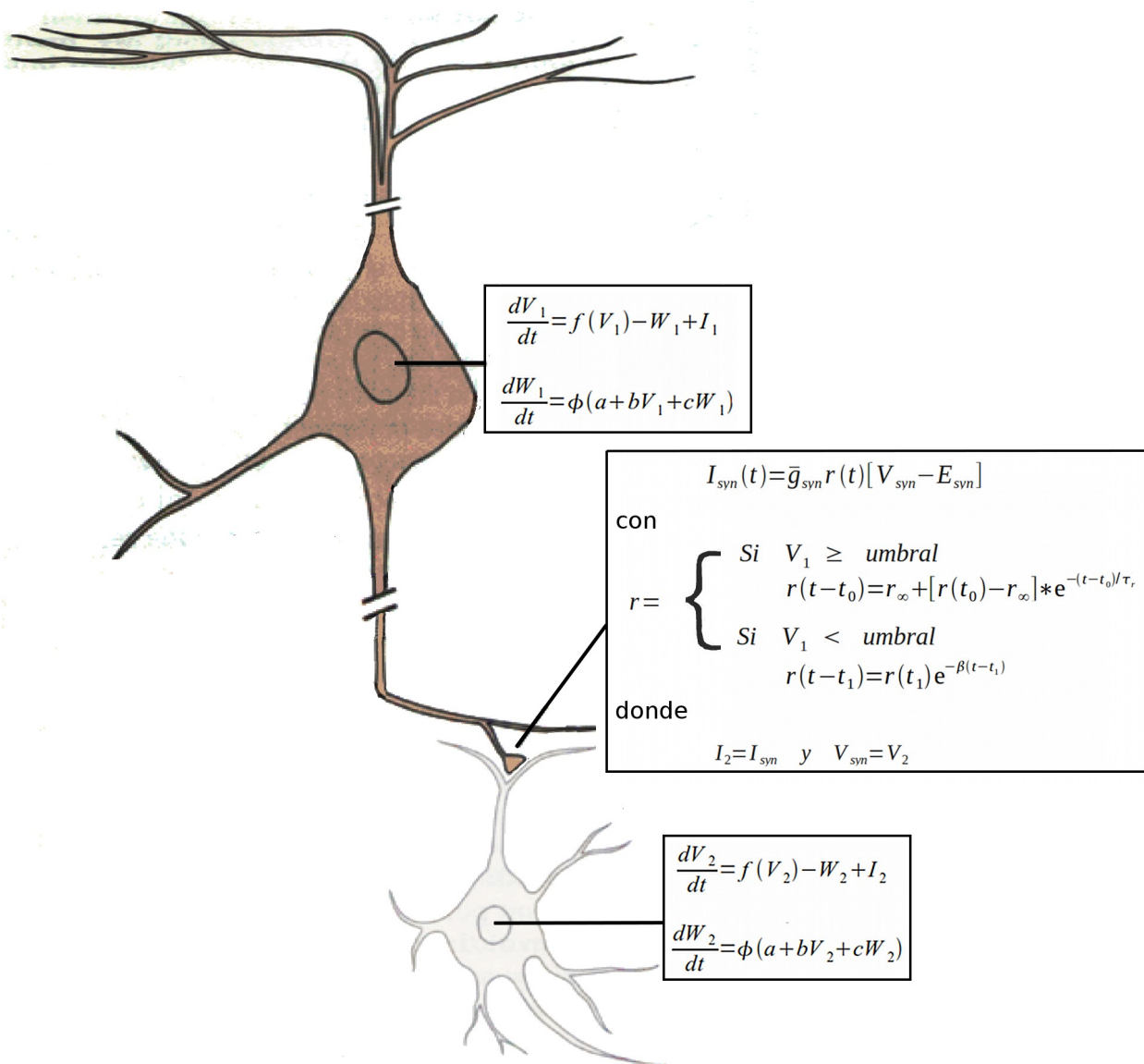


Figura 2.2. Diagrama de la propagación de un potencial de acción de una neurona a otra. En éste caso, las dos neuronas representadas en el dibujo implementan el modelo neuronal de FitzHugh-Nagumo, siendo las variables V_1 el potencial de membrana la neurona presináptica y, V_2 el potencial de membrana de la neurona postsináptica. La fórmula para la conexión sináptica hace uso de los potenciales de membrana de ambas neuronas: El potencial de la neurona presináptica se compara con un umbral pre-establecido y selecciona el tipo de ecuación que se usará para resolver el valor de R , mientras que, el potencial de la neurona postsináptica, es un factor que afecta directa y proporcionalmente al valor de la corriente que esta conexión sináptica transmitirá.

Capítulo 3: Metodología

Previo al inicio de diseño e implementación del programa, se tomó la decisión del tipo de ingeniería de software que se trabajaría. Al hacer la estimación y la elección de la metodología a utilizar se tomaron en cuenta dos factores importantes característicos del diseño de software científico:

- Muchos de los conceptos, metáforas y usos del software en cuestión son altamente especializados. En la mayoría de los programas que sirven de soporte a la investigación científica, se deben trabajar temas muy específicos de áreas de especialización científica. En el caso del programa presentado en esta tesis, pertenece al área de neurociencias. De este modo, fue prioritario el mantener una sólida comunicación con los clientes, en este caso los integrantes del Laboratorio de Neurofisiología Integrativa de la BUAP, para así poder corroborar el avance del proyecto en cada una de sus etapas.
- Como una buena parte del software científico, este sistema debía asegurarse de cumplir enteramente los requisitos de un módulo antes de poder continuar con la siguiente etapa en la ingeniería del sistema. Existen muchas áreas en donde lo importante es el sistema en general, el todo. En este caso, fue mucho más importante completar y mejorar pequeñas partes del sistema, que proporcionar una gran cantidad de funcionalidades como lo plantean los requisitos iniciales.

Al tomar en cuenta estas dos características medulares sobre el proyecto, se decidió usar la metodología de software conocida como Programación Extrema (XP por sus siglas en Inglés) (Donovan Wells (Don Wells), 2009). En el Apéndice al final de este documento se encuentra una sección dedicada a explicar las fases de la XP.

Las siguientes secciones muestran los pasos de la ingeniería para el sistema presentado. Tómese en cuenta que muchas de las secciones, si bien no son parte de la XP, representan los pasos básicos que la mayoría de las metodologías de software usan. En el caso de la XP, cada uno de estos pasos se condensa y se aplica dentro de cada iteración de la ingeniería.

3.1 Investigación y recopilación de bibliografía

En esta etapa se hizo una extensa investigación sobre los temas relacionados con la simulación de redes neuronales artificiales y, específicamente, la de redes CPG. Con el apoyo de los integrantes del Laboratorio de Neurofisiología Integrativa, esta etapa sirvió para familiarizarse con los conceptos biológicos y médicos relacionados con el proyecto.

A continuación se listan los temas más importantes de investigación que fueron requeridos así como una de las referencias bibliográficas usadas en cada uno de ellos.

- Neurociencias: se cubrieron conceptos básicos del estudio del sistema nervioso central y su funcionamiento. Texto base: *Essentials of neural science and behavior* (Kandel, Schwartz, & Jessell, 1996).
- Generador Central de Patrones: fue necesario entender la estructura, ubicación y características de estas redes neuronales. Texto base: "Organization of mammalian locomotor rhythm and pattern generation" (McCrea & Rybak, 2008).
- Modelos neuronales: se investigaron los tipos de modelos biológicos de neurona con tal de familiarizarse con su uso en sistemas computacionales. Texto base: *Dynamical Systems in Neuroscience: The Geometry of Excitability and Bursting* (Computational Neuroscience) (Izhikevich, 2006).
- Integración numérica: se hizo una revisión de diversos métodos de integración numérica con el propósito de encontrar el más adecuado para ser aplicado en el sistema. Texto base: *Ecuaciones Diferenciales y Problemas con Valores en la Frontera* (Boyce & DiPrima, 2004).

3.2 Requisitos (historias de usuario)

Para recopilar las historias de usuario se llevo a cabo una etapa de familiarización con los conceptos que el sistema debía manejar. Con la ayuda de varios expertos se consultó diversa bibliografía introductoria relacionada con neurociencias y, en específico, con la modelación de redes CPG. Una vez concluida esta etapa, se redactaron conjuntamente los requisitos que debería cumplir el sistema (historias de usuarios).

A continuación se reproducen algunas de las historias de usuarios recopiladas. Únicamente se muestran aquellas que ayudaron a modelar la arquitectura general del sistema, las demás están relacionadas con funciones de la interfaz de usuario o con algún concepto de neurociencias que no es importante discutir aquí.

Se requiere un sistema que sea capaz de simular el funcionamiento de las redes CPG. En específico, se requiere simular una arquitectura de dos capas similar a la planteada por Brownston y Wilson compuesta por una capa generadora de ritmo y una formadora de patrones. La arquitectura debe incluir también el modelo de UBG conectando longitudinalmente las capas de la arquitectura de CPG en una estructura en tres dimensiones.

El sistema debe representar la acción de seis motoneuronas (en la última capa) que correspondan a la flexión y extensión de los músculos de la cadera, rodilla y tobillo de un gato.

La capa formadora de patrones debe conectarse a las motoneuronas por medio de tres segmentos a distintas alturas (longitudinalmente) de la arquitectura de red. El sistema debe ofrecer la opción de configurar el número de nodos, en cada una de las capas de la arquitectura de red, y la posición de los segmentos, en los cuales se conectará la capa formadora de patrones con las motoneuronas. Adicionalmente, el sistema debe poseer la

capacidad de conectar y desconectar nodos de la red a petición del usuario.

Cada nodo de la red debe estar compuesto por un número configurable de neuronas. Estas neuronas deben ser simuladas por medio del modelo FitzHugh-Nagumo y deben compartir entre sí los valores de las fórmulas del modelo. El sistema debe ofrecer la capacidad de configurar cada nodo de la red con distintos valores. Las poblaciones de neuronas inhibitorias deben ser simuladas con alguna función particular y se debe dar la opción para configurar parámetros de esta fórmula también.

Se debe incluir un factor de diversidad en el comportamiento de las neuronas, es decir, tanto los valores de las fórmulas usadas como los de las conexiones sinápticas deben presentar ligeras variaciones aleatorias que pueden incrementar o disminuir a petición del usuario.

El sistema debe graficar las señales correspondientes a las tres capas de la arquitectura de red (generador de ritmo, formador de patrones y motoneuronas) así como permitir al usuario guardar estos datos en archivos en formato CSV.

Para analizar estas historias de usuarios se hizo un proceso de extracción de verbos y sustantivos, que después serán representados en el programa como objetos y operaciones. La tabla siguiente muestra el resultado de un proceso como éste sobre las historias de usuarios.

Palabra	Párrafo	Cómo tratarlo	Observaciones
simular	1	operación	La idea central del sistema es la simulación. Muchos de los objetos que se creen deberán ofrecer este comportamiento.
red CPG	1	objeto	Éste quizás se el objeto más alto en la jerarquía.
generador de ritmo	1	objeto	Parte del objeto red. Probablemente exista un objeto “capa” que englobe a todas las capas de la red.
generador de patrones	1	objeto	Parte del objeto red. Probablemente exista un objeto “capa” que englobe a todas las capas de la red.
motoneuronas	2	objeto	No es claro si las motoneuronas serán un objeto por separado o formarán una capa.
nodo	2	objeto	Cada capa será constituida por diversos nodos
conectar / desconectar	3	operación	Todos los elementos de las capas deben tener la capacidad de conectarse con otros elementos.
configurar	3	operación	Muchos de los objetos de la red serán configurables, ya sea internamente o en su comunicación con otros.
FitzHugh-Nagumo	4	objeto	Se necesitará relacionar cada nodo con un objeto de este tipo.
graficar / guardar	6	operación	Esta operación está claramente relacionada con la interfaz de usuario que se use.

En el análisis completo se hace una revisión mucho más extensa que no parece necesario presentar aquí. Esta tabla representa un buen ejemplo del proceso que se llevó

a cabo.

Además es importante destacar desde un inicio los casos de uso más importantes que el software ofrecerá. En la figura se muestran ejemplos de los casos de uso que fueron generados durante esta etapa.

3.3 Análisis

En esta etapa (parte de la fase de exploración en la XP) se hizo el primer prototipo del sistema. En este momento fue muy importante la comunicación con los clientes, en nuestro caso el Dr. Elías Manjarrez López y los integrantes del Laboratorio de Neurofisiología de la BUAP (de ahora en adelante referidos como expertos en este documento).

Es importante mencionar que por diversas razones, en la etapa previa al diseño de este sistema se tenía claro que se utilizaría algún lenguaje orientado a objetos que ofreciera fácil acceso al desarrollo de interfaces de usuario. Mucho del análisis que se muestra a continuación está completamente orientado a este tipo de lenguajes.

Tomando en cuenta el análisis de las historias de usuarios y los casos de uso generados, se creó un prototipo arquitectónico. Este prototipo debía contener una idea general de la arquitectura interna del sistema así como una separación de módulos funcionales por medio de los cuales se pudiera iniciar el proceso de iteraciones.

El primer trabajo fue establecer el tipo de objetos que se diseñarían así como las interacciones entre ellos. Para esto se hizo uso nuevamente del apoyo de expertos para entender y plasmar en el prototipo conceptos como población neuronal, capa, sinapsis excitatorias e inhibitorias, factor de diversidad o modelo neuronal. La figura 3.1 muestra el diseño al cual se llegó. Las conclusiones que se derivaron en este proceso fueron:

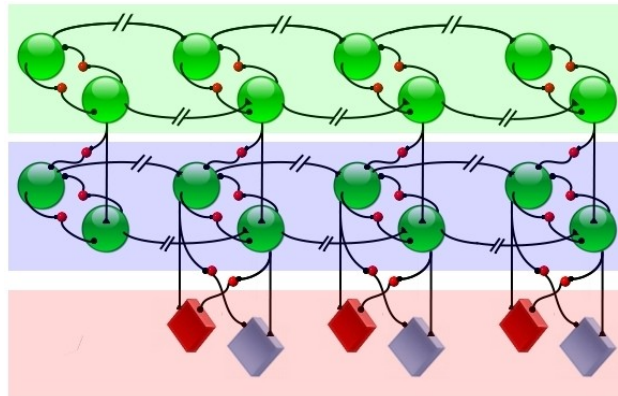


Figura 3.1. Diagrama de la arquitectura de red neuronal usada en el simulador. En el área sombreada de verde, el generador de ritmo; en la sombreada de azul, el formador de patrones y en la sombreada de rojo, las neuronas motoras. Las esferas de color verde representan poblaciones de interneuronas, las esferas en color rojo conexiones inhibitorias y los rombos de color rojo y azul son poblaciones de neuronas motoras extensoras y flexoras, respectivamente. Los sesgos en las conexiones longitudinales indican que el número de estructuras verticales conectadas entre sí es variable. Diagrama modificado de (Pérez et al., 2009).

- En la parte más baja de la jerarquía de objetos se encuentran los “nodos neuronales” y las “conexiones sinápticas”. Cada “nodo neuronal” está asociado a un objeto de tipo “modelo neuronal” que alberga la información necesaria para simular matemáticamente algún modelo de neurona (ejemplo: FitzHugh-Nagumo). Por su parte, las “conexiones sinápticas” están asociadas a objetos análogos llamados “modelos sinápticos”.
- La red completa se representa a través de la conjunción de tres objetos: “formador de patrones”, “generador de ritmo” y “motoneuronas”. Cada uno de estos objetos está constituido a su vez tanto de “nodos neuronales” como de “conexiones sinápticas”.
- Cada “nodo neuronal” tiene la capacidad de distinguir en qué capa se encuentra y si es una motoneurona o un nodo regular. Ofrece también la capacidad de modificar todos sus parámetros internos.
- Cada “conexión sináptica” tiene la capacidad de distinguir qué tipo de sinapsis es: excitatoria o inhibitoria. Ofrece también la capacidad de modificar todos sus parámetros internos.

Posteriormente se llevó a cabo la separación funcional del sistema. Se establecieron los módulos con los cuales se trabajaría y se detallaron las características que tendría cada uno. A continuación se explica cada uno de estos módulos funcionales:

- **Módulo Principal (Core).** Este módulo está constituido por todos los objetos medulares de la arquitectura: “nodos neuronales”, “conexiones sinápticas”, todas las capas, “red”. Cada uno de estos objetos debe tener una interfaz estable para interactuar con el exterior puesto que todos los objetos de los demás módulos trabajarán exhaustivamente con ellos.
- **Módulo de Graficado.** Todos los objetos del módulo principal poseen un valor numérico

que representa su estado en algún momento de la simulación. El módulo de graficado será el encargado de presentar estos valores en forma de gráficos de señales eléctricas. Todos los objetos de este módulo deben ser capaces de interactuar con el usuario por medio de la interfaz gráfica.

- **Módulo de Graficado 3D.** La idea de este módulo es lograr hacer una presentación 3D en tiempo real de la red que se esté simulando. Este módulo hará uso de varias características de los objetos del módulo principal, como pueden ser el tipo y posición del objeto en la red. El resultado planeado es un dibujo interactivo de la red muy parecido a lo que se muestra en la figura 3.1.
- **Módulo de Configuración Interactiva.** Los objetos del módulo principal ofrecen una interfaz para ser configurados, sin embargo, acceder a esta interfaz requeriría de modificar el código y recompilar el sistema. Por esta razón, el trabajo de este módulo es ofrecer al usuario una interfaz gráfica para actualizar los valores de cada objeto en tiempo real.
- **Módulo de Administración de Proyecto.** Aunque no se incluyó mucho acerca de este módulo en las historias de usuarios, se consideró buena idea ofrecer herramientas al usuario para poder trabajar con el sistema por medio de proyectos. Esto implica incluir operaciones como guardar y cargar diversos tipos de archivo y estados del programa.

3.4 Diseño

El diseño de software (fase de iteraciones en la XP) fue la etapa de mayor trabajo del proyecto. Los diseños, diagramas y esquemas que se presentan a continuación representan las últimas iteraciones en cada uno de los módulos funcionales del sistema.

Parece pertinente mencionar que, desde las primeras etapas del diseño, se decidió que se trabajaría con el lenguaje de programación Java™ de Sun Microsystems, Inc. En las descripciones de los módulos que se presentan a continuación se hacen algunas referencias a conceptos relacionados con este lenguaje de programación.

Módulo Principal (Core)

El módulo principal representa el núcleo de la aplicación CPG Network. Por sí solo es el encargado de crear, actualizar y simular las redes CPG.

Para explicar de mejor manera los contenidos de este módulo, las secciones siguientes detallan el funcionamiento de los objetos más importantes que lo componen desde una visión “bottom-up” (desde lo específico hasta lo general); es decir, se explican primero las estructuras más bajas en la jerarquía de objetos.

Poblaciones neuronales

Las poblaciones de neuronas son representadas en el sistema como objetos de tipo "NeuronPopulation". Este tipo de objetos contiene toda la información necesaria tanto para modelar como para simular una población neuronal por medio de alguno de los modelos neuronales presentados en la sección 2.2 "Modelos matemáticos de neurona". En la figura 3.2 se muestra el diagrama de clases correspondiente a todos los objetos relacionados con las poblaciones neuronales.

El objeto "NeuronPopulation" presenta 8 variables miembro esenciales:

- La variable "numberOfNeurons" es un entero positivo que representa la cantidad de neuronas que la población contendrá.
- La variable "diversityFactor" es un número real que indica la cantidad de diversidad que será aplicada a cada una de las neuronas de esta población. Esta diversidad es implementada como la suma de un valor aleatorio contenido en el intervalo $[+\text{"diversityFactor"}, -\text{"diversityFactor"}]$.
- Las variables "step" e "integrationCicle" son el tamaño de paso y el tamaño de ciclo del algoritmo de integración numérica que se esté usando. Hasta la versión 1.6-r de programa CPG Network, el método de integración predefinido es una implementación del método de Runge-Kutta de cuarto orden. En la sección "Integración numérica para ecuaciones diferenciales ordinarias" se encuentra una breve explicación de éste y otros métodos.
- Tanto la variable "model" como el arreglo "modelDiv" son objetos de tipo "NeuronalModel". Esto quiere decir que contendrán alguno de los objetos que descienden de la clase "NeuronalModel", y que están asociados a algún modelo neuronal en particular. El arreglo "modelDiv" sirve para mantener un registro de la diversidad aplicada a los parámetros del modelo neuronal, sin tener que modificar los parámetros originales.
- Los arreglos "V" y "W" representan los valores del potencial de campo y la variable de recuperación respectivamente. En ambos casos, los valores retornados por las funciones "getFieldPotential()" y "getRecoveryVariable()" son calculados como la suma de los valores contenidos en cada uno de estos arreglos.

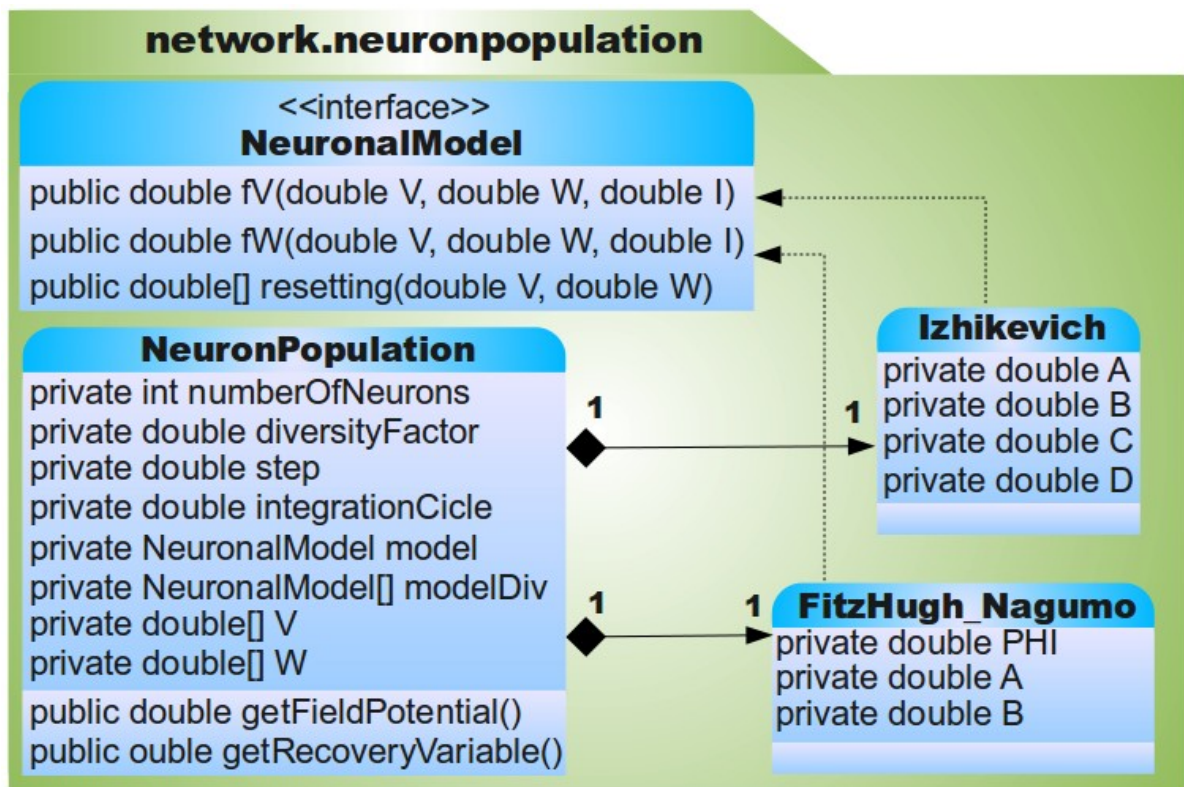


Figura 3.2: Diagrama de clases de las poblaciones neuronales

Por su parte, la interfaz “NeuronalModel” se encarga de agrupar a todas las clases que representen alguno de los modelos neuronales. Esta interfaz ofrece tres funciones miembro principales:

- “fV()” y “fW()” son las funciones que hacen el cálculo de las dos ecuaciones de los modelos neuronales. Tanto el modelo de FitzHugh-Nagumo como el modelo de Izhikevich están compuestos por un sistema de ecuaciones diferenciales de dos variables; éstas dos funciones representan a cada una de las ecuaciones de los sistemas. Como se puede notar, hasta ahora la arquitectura del sistema es capaz de representar únicamente aquellos modelos neuronales con dos variables, como una perspectiva, se tiene la idea de modificar estos objetos para poder representar cualquier tipo de modelo neuronal.
- La función “resetting()” se encarga de calcular la hiperpolarización potencial de membrana que algunos modelos necesitan hacer explícitamente. Modelos como el de FitzHugh-Nagumo son capaces de generar este comportamiento de manera intrínseca, por lo cual simplemente harán caso omiso de esta función.

Los objetos “FitzHugh-Nagumo” e “Izhikevich” implementan a la interfaz “NeuronalModel” y generan las características adecuadas para representar a algún modelo neuronal. Hasta la versión 1.6-r del programa CPG Network, estos dos son los únicos objetos implementados para representar modelos neuronales.

Conexiones sinápticas

Las conexiones sinápticas entre cada población neuronal se representan internamente como objetos de tipo “SynapticConexion”. Este tipo de objetos contiene toda la información necesaria tanto para modelar como para simular conductancias sinápticas por medio de la fórmula presentada en la sección 2.2 “Modelos matemáticos de neurona”. En la figura 3.3 se muestra el diagrama de clases correspondientes a los objetos relacionados con las conexiones sinápticas.

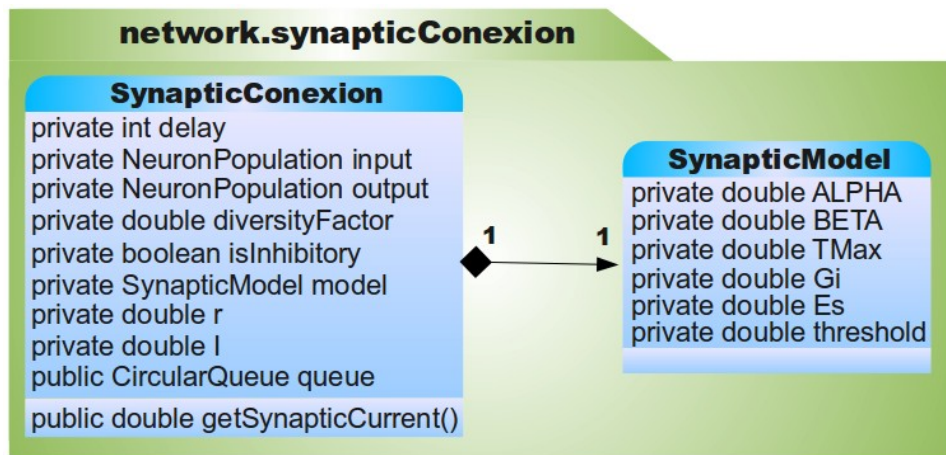


Figura 3.3: Diagrama de clases de las conexiones sinápticas

El objeto “SynapticConexion” presenta 9 variables miembro esenciales:

- “delay” es un entero que representa el tiempo de retraso que ésta conexión sináptica tardará en llevar la información desde la población neuronal origen hasta la población neuronal destino.
- “input” y “output” son variables de tipo “NeuronPopulation” que alojan apuntadores a las poblaciones neuronales origen y destino respectivamente.
- “diversityFactor”, de la misma manera que con las poblaciones neuronales, es un real que indica la cantidad de diversidad que será aplicada a esta conexión sináptica.
- “isInhibitory” es una variable de tipo booleano que sirve como bandera para indicar si esta conexión sináptica es excitadora o inhibidora.
- “queue” es una instancia de “CircularQueue” que sirve para almacenar temporalmente los datos de entrada de la conexión sináptica.
- “r” e “I” son variables relacionadas con la fórmula de conductancias sinápticas, utilizada

para la simulación de las conexiones sinápticas. “r” es una variable de estado necesaria para alojar información del previo estado en la simulación. “I” representa la corriente sináptica de esta conexión para cualquier tiempo dado de la simulación.

El objeto “SynapticModel” es el encargado de almacenar los parámetros necesarios para llevar a cabo la fórmula de conductancia sináptica.

Estructuras de datos

Para la implementación de este sistema se diseñó una serie de estructuras de datos orientadas a objetos, sin las cuales hubiera sido imposible administrar a los demás objetos de la red. La figura muestra un diagrama de clase de las estructuras de datos más importantes que se implementaron.

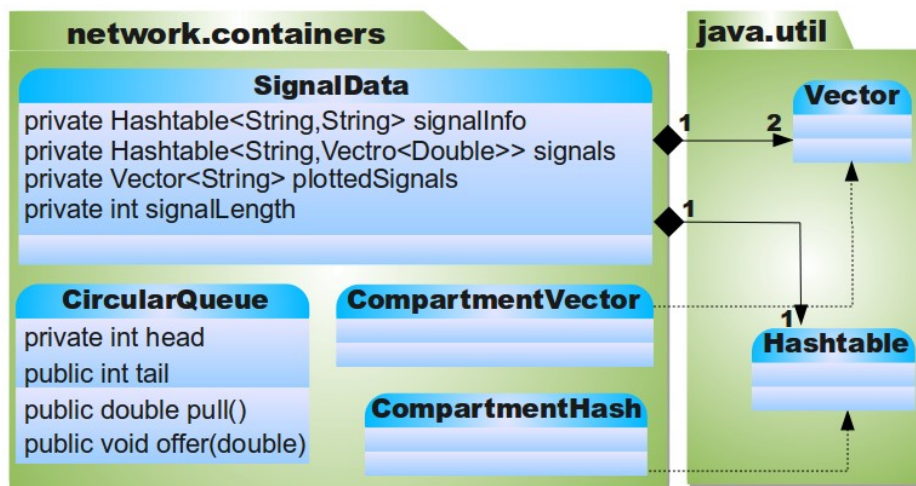


Figura 3.4: Diagrama de clases de las estructuras de datos diseñadas

El objeto “SignalData” es probablemente el más importante dentro del paquete “network.containers”. Es el encargado de llevar un registro de los valores de los objetos en una simulación. Como se explicó anteriormente, objetos como los “nodos neuronales” tienen un valor que se actualiza en cada tiempo de la simulación; en el caso de los “nodos neuronales” este valor en realidad representa el potencial de campo asociado a la población neuronal que el nodo contiene. “SignalData” es el objeto encargado de almacenar estos valores en cada tiempo de la simulación para todos los objetos que posean la capacidad de simular.

“SignalData” tiene cuatro variables miembro esenciales:

- “signalInfo” es una tabla Hash que mapea nombres de objetos con su descripción. Esta información es muy útil para todos los objetos relacionados con la interfaz.
- “plottedSignals” es un arreglo que indica que señales están siendo graficadas en el módulo de graficado. Esta lista es mayormente usada por objetos del módulo de graficado.

- “signalLength” es el tamaño de los arreglos de datos para cada señal almacenada en este objeto. Este dato también es importante para saber el tiempo de simulación que se está trabajando.
- “signals” es una tabla Hash que mapea nombres de objetos con arreglos de datos. Esta variable contiene todos los datos de simulación del sistema

El objeto “CircularQueue” no es más que la implementación de una cola circular. Entre sus estructuras de datos, el lenguaje Java no ofrece una clase de este tipo entre sus colecciones. Este objeto es usado por las “conexiones sinápticas” para mantener un registro de sus valores previos en una simulación.

El objeto “CompartmentVector” es el encargado de mantener en memoria arreglos de objetos de tipo “Compartment”; en la siguiente sección se explicarán este tipo de objetos. La idea de tener una clase como ésta es facilitar el proceso de añadir y eliminar objetos de la red.

El objeto “CompartmentHash” es una tabla Hash que mapea nombres de objetos con los objetos en sí. Este objeto es crucial para el proceso de simulación de la red, y sirve de gran ayuda en operaciones como búsquedas de objetos en la red.

Modelo por compartimientos

En este proyecto se toma la idea de la programación por compartimientos para llevar a cabo la simulación de la red. Cada objeto simulable es un compartimiento; o mejor dicho, todo objeto que se pueda simular debe ser un compartimiento. La figura 3.5 muestra un diagrama de clase con los objetos más importantes del paquete “network.compartment”.

La interfaz “Compartment” es el objeto principal en este paquete. Todo objeto en la red que desee tener un comportamiento de compartimiento y, por ende, poderse simular, debe ser declarado como una implementación de esta interfaz.

Algo que hay que mencionar acerca del modelo por compartimientos adoptado en este programa, es el hecho de que ningún compartimiento tiene conocimiento de con qué compartimientos está conectado. Lo único que se sabe desde uno de estos objetos es que otros compartimientos están conectados hacia éste. Durante las primeras etapas de diseño se pensó mucho si convendría adoptar este tipo de comportamiento o si sería mejor implementar una estructura más al estilo de un grafo. Se llegó a la conclusión de usar compartimientos por la gran facilidad que ofrecen en el momento de la simulación. En realidad, al usar este modelo en combinación con la estructura de datos “CompartmentHash”, el proceso de simulación se reduce a una simple iteración sobre todos los objetos de la red, llamando a su operación para simular en cada uno de ellos.

Adicionalmente, la interfaz “Compartment” ofrece algunas operaciones relacionadas con la información que cada compartimiento aloja. Esta información comprende campos como nombre, descripción y algunas otras banderas.

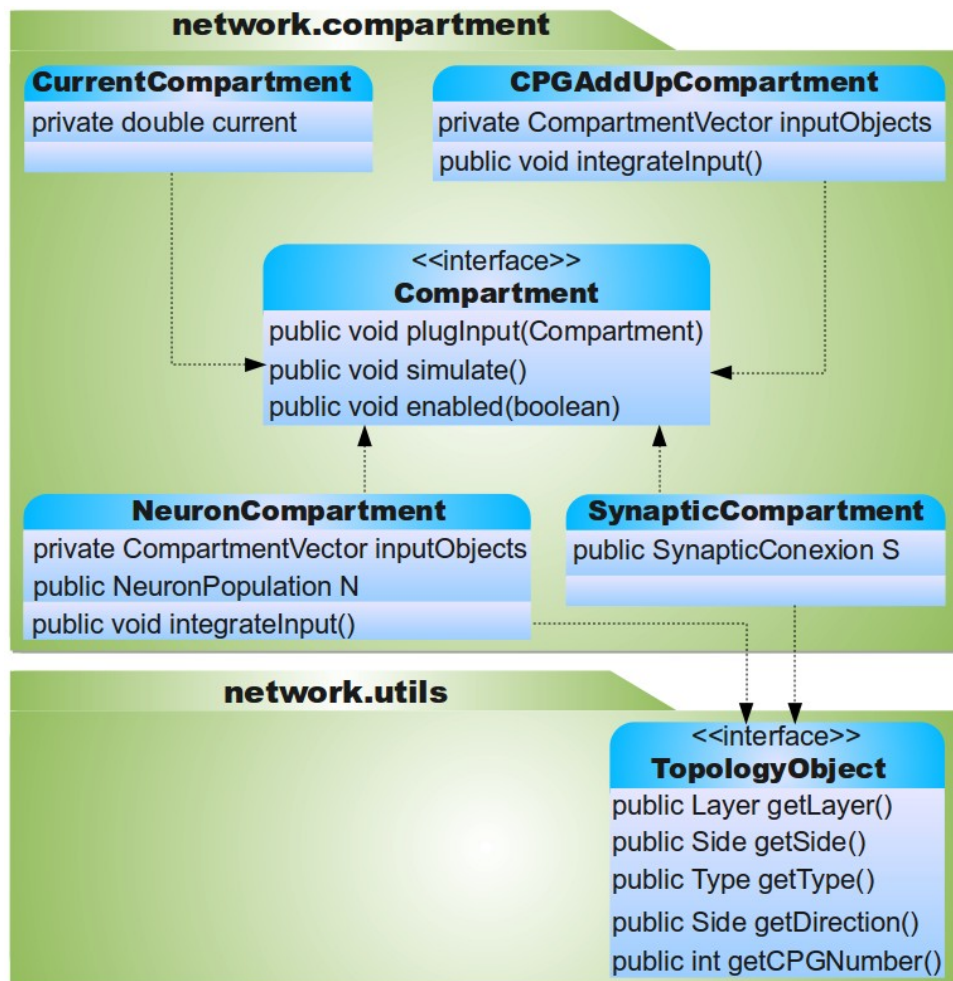


Figura 3.5: Diagrama de clases de los objetos compartimento

Hasta la fecha, en la versión 1.6-r del programa CPG Network, se tienen cuatro clases que descienden de la interfaz “Compartment”; “SynapticCompartment”, “NeuronCompartment”, “CurrentCompartment” y “CPGAddUpCompartment”. A continuación se muestra una pequeña descripción de cada una de ellas.

- “NeuronCompartment” es el objeto que aloja objetos del tipo “NeuronPopulation”. Éstos representan los nodos neuronales de la red. Este tipo de compartimentos también desciende de otra interfaz llamada “TopologyObject” que le agrega información necesaria para ubicar al nodo neuronal dentro de la red.
- “SynapticCompartment” es el objeto que aloja objetos del tipo “SynapticConexion”. Éstos representan todas las conexiones sinápticas de la red. Este tipo de compartimentos también desciende de otra interfaz llamada “TopologyObject” que le agrega información necesaria para ubicar a la conexión sináptica dentro de la red.
- “CurrentCompartment” es un objeto invisible para el usuario. La función de este tipo de objetos es emitir un pulso constante de corriente continua a todos los compartimentos que se encuentren conectados a él. En toda la red solo hay dos de estos compartimentos, conectados a los primeros nodos neuronales de las capas formadora

de patrones y generadora de ritmo.

- “CPGAddUpCompartment” es el objeto que cumple la función de los electrodos de lectura en el sistema. Trabajan sumando todas las entradas que tengan. En la red habrá el mismo número de estos compartimientos como de CPG's. Cada uno de ellos estará asociado a cuatro nodos neuronales, dos del generador de ritmo y dos del formador de patrones, para cada CPG.

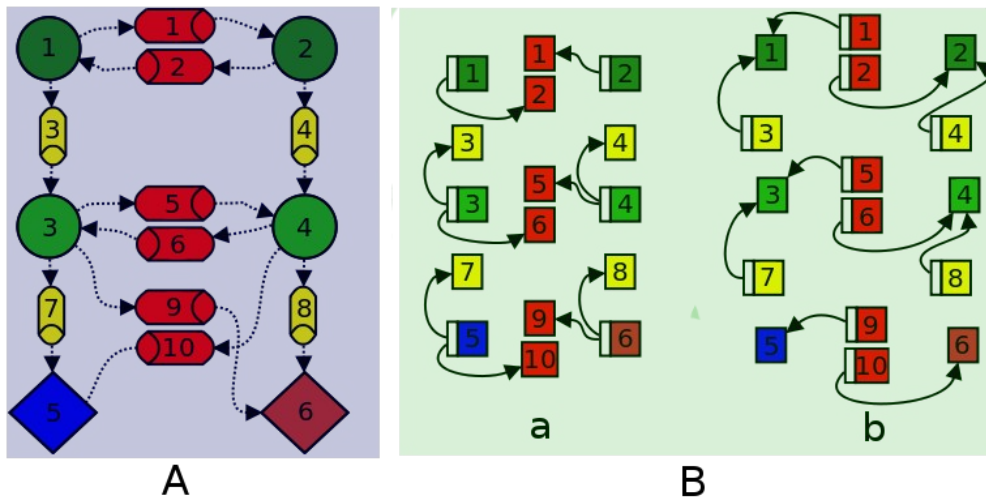


Figura 3.6: Ejemplo de la representación interna de una red por medio del modelo por compartimentos.

En la figura 3.6 se muestra un ejemplo de como se representa internamente una red por medio del modelo por compartimentos utilizado en el sistema. La red original (figura 3.6 A) muestra una estructura de CPG con el modelo de Brownston y Wilson. Los nodos neuronales están representados con esferas y las conexiones sinápticas con cilindros. Internamente esa red sería representada como la combinación de los diagramas en la figura 3.6 B.a y B.b. La primera muestra las conexiones de todos los objetos de tipo “NeuronCompartment” y la segunda las de los objetos de tipo “SynapticComaprtment”.

Arquitectura de la red

Finalmente, una vista más externa de la arquitectura del sistema muestra las relaciones entre los objetos más altos de la jerarquía. Todos estos objetos contienen múltiples instancias de los demás objetos descritos previamente. En la figura 3.7 se muestra el diagrama de clases que corresponde al paquete “network”.

El objeto padre en este paquete es llamado “Network”. Una instancia de este objeto representa una red CPG completa y funcional. Entre otras variables y operaciones miembro, los más destacados son:

- “numberOfCPGs” es un entero que indica, como su nombre lo dice, el número de CPG's verticales que la red contiene. Por diseño, este número no puede ser menor a tres ni mayor a 100.

- “MNPositions” es un arreglo de enteros con tres posiciones, que indica en cuáles CPG's deberán ser posicionadas cada una de las poblaciones de motoneuronas. En total hay 6 de estas poblaciones y agrupadas en pares se posicionan en tres lugares distintos de la red.
- La funcionalidad más importante de la red es ser capaz de simularse. Por propósitos de interacción con el usuario, esta simulación se lleva a cabo por medio de llamados sucesivos a la operación “simulate()”, la cual está encargada de simular sólo un paso o tiempo a la vez.

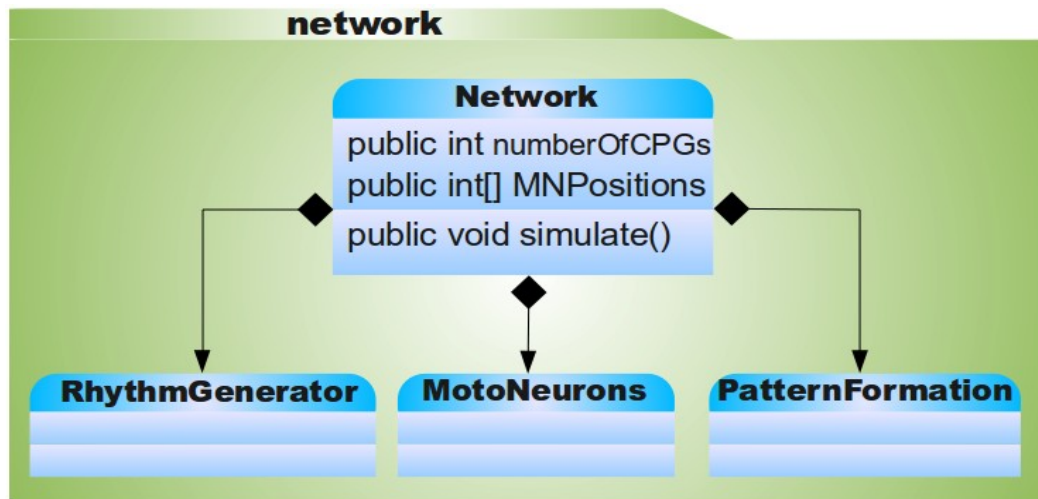


Figura 3.7: Diagrama de clases de la Arquitectura de la red

Los otros tres objetos del paquete “network” representan a cada una de las capas (verticalmente) de la red. Estos objetos son “RhythmGenerator” (el Generador de Ritmo), “PatternFormation” (el Formador de Patrones) y “MotoNeurons” (la capa de motoneuronas). Cada uno de ellos ofrece diversos atributos y operaciones que ayudan al objeto “Network” en el proceso de modelación y simulación de la red.

Módulo de graficado

Para este módulo se diseñó e implementó un graficador de señales digitales. Su objetivo principal es ofrecer al usuario una interfaz sencilla e intuitiva en la cual pueda visualizar el comportamiento de las señales generadas por los objetos de la red.

El resultado tangible de este módulo es un panel, llamado panel de señales, que despliega, a petición del usuario, conjuntos de objetos de la red en una gráfica con ejes cartesianos, donde el eje horizontal representa el tiempo, y el eje vertical el valor de cada señal.

Para intentar reducir la curva de aprendizaje de los usuarios, y en vista de que se tiene perfectamente ubicado el tipo de usuario del programa, se hizo todo lo posible por reproducir la presentación y las funciones de módulos similares en otros paquetes de

software científico.

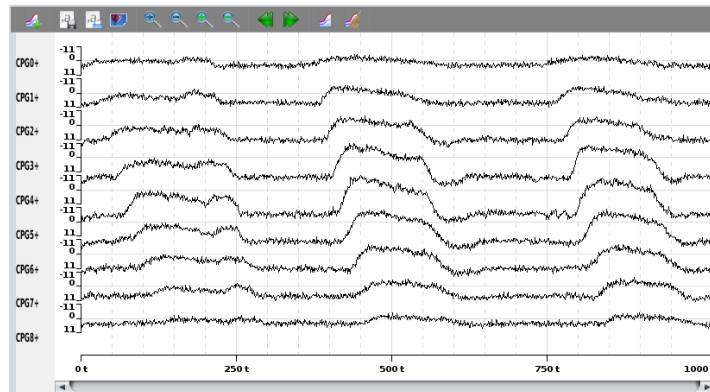


Figura 3.8: Impresión de pantalla del panel de señales.

La figura 3.8 muestra una imagen del panel de señales en ejecución. Éste es el objeto con mayor cantidad de funciones en la ventana principal. Está compuesto por tres elementos: el panel de etiquetas, que muestra el nombre de las señales graficadas; el panel de dibujo, que despliega las señales; y la caja de herramientas, con diversas funciones que se pueden realizar sobre el panel.

Cuando se trabaja sobre éste panel, es posible escalar la vista de las señales tanto en el eje vertical como el horizontal, así como mover el puerto de visión de izquierda a derecha. El conjunto de señales que se muestran puede ser configurado por el usuario, ya sea seleccionando objetos individuales o conjuntos de objetos predefinidos. Adicionalmente, es posible guardar y cargar archivos de señales en distintos formatos.

Para más información acerca del uso de este y otros paneles de la interfaz, se recomienda leer el *Manual de Usuario del CPG Network* entregado junto con este documento.

Módulo de graficado 3D

Este módulo, como su nombre lo dice, es el encargado de mostrar una representación visual de la red CPG. Después de varias iteraciones en este módulo, se llegó a la conclusión de usar una herramienta de programación 3D integrada al lenguaje de programación Java; así se eligió utilizar la herramienta JPCT (<http://www.jpct.net/>).

El objetivo principal de este módulo es ofrecer al usuario una interfaz 3D, ligera e interactiva, para visualizar el estado actual de la red CPG que esté modelando. De esta manera, como resultado del trabajo en este módulo se llegó al diseño e implementación del panel llamado panel de visualización que se muestra en la figura 3.9.

Este panel muestra al usuario un dibujo interactivo 3D del estado de la red CPG. En este dibujo se incluyen tres objetos de la red: las poblaciones neuronales, las conexiones sinápticas y los compartimentos “CPGAddUpCompartment” conocidos como electrodos.

La interacción con este panel es similar a la que presenta cualquier aplicación 3D; se

puede acercar y alejar del objeto, rotar la vista y seleccionar algún objeto dibujado. Adicionalmente, la caja de herramientas de este panel ofrece opciones como guardar una imagen del dibujo o recargar la vista.

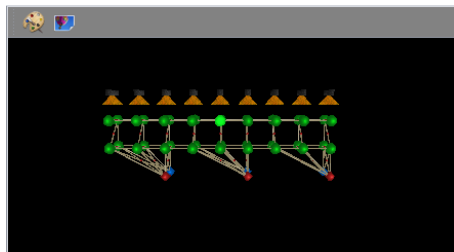


Figura 3.9: Impresión de pantalla del panel de Visualización

Para más información acerca del uso de éste y otros paneles de la interfaz, se recomienda leer el *Manual de Usuario del CPG Network* entregado junto con este documento.

Módulo de configuración interactiva

Con el objetivo de ofrecer al usuario una opción para actualizar los parámetros de los objetos de la red a través de la interfaz principal, dentro de este módulo se diseñó e implementó un panel llamado panel de configuración.

Este panel presenta una completa interfaz compuesta de una caja de herramientas, una sección de tablas de parámetros, y una sección de información. Cada tipo de objeto de la red tiene un modo distinto de panel de configuración, en el cual se presentarán todos sus parámetros dentro de tablas para que el usuario pueda editarlos.

Para más información acerca del uso de éste y otros paneles de la interfaz, se recomienda leer el *Manual de Usuario del CPG Network* entregado junto con este documento.

Módulo de administración de proyectos

El objetivo de este módulo es ofrecer al usuario distintos modos para gestionar un proyecto. Esto significa diseñar formatos de archivos específicos para el sistema e incluir modos para cargar y guardar estos archivos dentro de la interfaz.

Al observar las necesidades de los usuarios, se decidió crear dos formatos de archivo para esta aplicación:

- Archivos de red: almacenan toda la información necesaria para representar una red CPG. Estos archivos son guardados con extensión '*.cpg' y están compuestos de texto. Se dividen en tres secciones según el tipo de información que contienen. Al inicio, después de los encabezados, se listan las propiedades generales de la red. A

continuación vienen las declaraciones de todos los objetos de la red, en donde se incluyen tanto poblaciones neuronales como conexiones sinápticas. Al final del archivo se añade la sección de bitácora del proyecto en donde se almacena texto que el usuario escribió para describir a la red almacenada.

- Archivos de señales; almacenan todos los datos correspondiente a las señales de los objetos de la red CPG. Estos archivos están en el formato conocido como “valores separados por comas” (CSV por sus siglas en inglés) y son guardados con extensión '*.csv'. Están compuestos por dos secciones: la lista de objetos y la sección de datos. Después del encabezado se encuentra la lista de objetos, en la cual se lista en la primera fila el número de señales y posteriormente, el nombre y descripción de todas ellas. La segunda sección es la de datos, en donde se incluyen en columnas los valores para cada señal.

Dentro de la interfaz principal se ofrecen distintos modos de acceder a las funciones para guardar y cargar estos archivos. Adicionalmente algunos de los paneles en la interfaz ofrecen la opción de guardar la información en forma de imagen, usando el formato '*.png'.

3.5 Implementación de módulos

Como se mencionó previamente, la implementación del sistema se llevó a cabo en el lenguaje de programación Java™ de Sun Microsystems, Inc (<http://java.sun.com/>). Esta decisión fue tomada gracias a la alta capacidad de migración entre sistemas operativos que este lenguaje ofrece. Cabe señalar que se utilizó un ambiente integrado de programación para facilitar esta etapa.

Para el módulo de Graficado 3D se utilizó la biblioteca de gráficos JPCT (<http://www.jpct.net/>). Se decidió usar esta biblioteca por ser 100% compatible con Java y por no necesitar de hardware adicional para su ejecución.

Como técnica de codificación se utilizó un sistema de control de versiones basado en el famoso Sistema de Versiones Concurrentes (CVS por sus siglas en inglés). De esta manera, todas las versiones del software presentado se encuentran disponibles para consulta, documentadas y almacenadas ordenadamente.

Para seguir con la filosofía de la metodología XP, se hizo uso del modelo de pruebas ofrecido por Java, JUnit. Gracias a este modelo se lograron cumplir muchas de las entregas a tiempo y libres de errores.

3.6 Documentación y entrega

La entrega del sistema presentado en este documento a los integrantes del Laboratorio de Neurofisiología Integrativa, BUAP, consta de dos elementos principales:

- El código y los ejecutables de la aplicación. El programa se entrega listo para su uso y acompañado de los códigos fuentes correspondientes a la última versión trabajada. El código se encuentra bajo licencia GPL2.
- Un manual de usuario y uno técnico. Junto con el software, se entrega un manual de usuario donde se explican todas las funciones que el programa ofrece. Además, se facilita la documentación técnica en formato JavaDoc, con el propósito de que se continúe con el diseño del sistema.

Adicionalmente, se tiene pensado diseñar una página web destinada al programa, en donde todos los elementos de la entrega se encuentren accesibles al público en general.

Capítulo 4: Resultados

El resultado más notorio de este trabajo es la aplicación CPG Network. Como se dijo anteriormente, la entrega del programa se hizo directamente a los integrantes del laboratorio de Neurofisiología Integrativa de la BUAP, junto con los manuales tanto de usuario como técnicos. Las siguientes secciones muestran los rasgos más importantes de la aplicación entregada. El caso de uso presentado al final de este capítulo es parte del manual de usuario que acompaña al programa.

4.1 Desempeño del programa

Para medir el desempeño de la aplicación CPG Network se tomaron en cuenta dos parámetros: el tiempo de ejecución y la cantidad de memoria usada. Las mediciones de estos parámetros se llevó a cabo únicamente durante el proceso de simulación de la red, que representa el momento crítico en el cual el programa consume mayor cantidad de recursos.

Como se explicó en la sección de metodología, todas las operaciones de búsqueda y recorrido de las estructuras de datos se llevan a cabo por medio de tablas Hash. Este tipo de tablas ofrece una complejidad constante en general y una complejidad lineal en el peor caso. Por esta razón, parece poco significativo incluir un análisis de los recursos utilizados por el programa en procesos fuera del momento crítico mencionado.

Tanto el tiempo de ejecución como la cantidad de memoria que el programa consume durante una simulación dependen directamente de algunos valores específicos de la red que se esté modelando. En particular, la cantidad de tiempos de simulación que se efectuarán y la cantidad de objetos que componen la red son los valores que impactan más significativamente. La cantidad de tiempos de simulación se maneja como un parámetro desde la ventana principal, y la cantidad de objetos de la red depende directamente de cuántos CPG's se encuentren acoplados. En la figura 4.1 se muestran los resultados de dos pruebas realizadas sobre el programa variando estos parámetros.

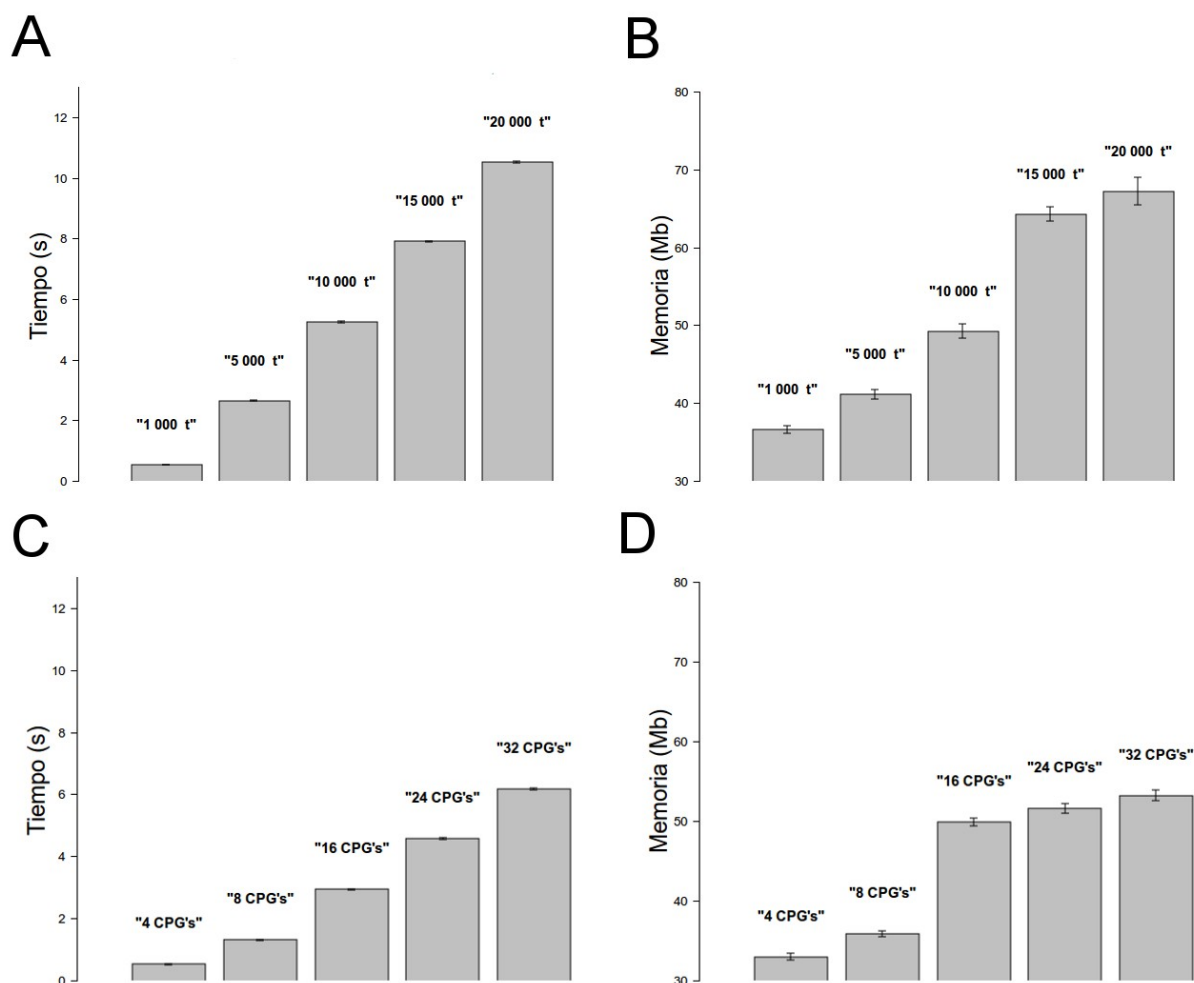


Figura 4.1: Resultados de las pruebas de desempeño sobre el programa. En **A** y **B** se muestran el tiempo de ejecución y la memoria utilizada respectivamente en una prueba modificando el tiempo de simulación. En **C** y **D** el tiempo de ejecución y la memoria utilizada en una prueba modificando el número de CPG's de la red.

Para obtener estos resultados se obtuvieron promedios de 30 ejecuciones con cada una de las configuraciones de parámetros mencionada. Como se puede ver, el aumento en el tiempo de ejecución es proporcional al aumento del tiempo de la simulación, sin embargo, el crecimiento del uso de memoria sigue un patrón menos estereotipado. Esto tiene mucha relación con la gestión de memoria que presenta el lenguaje de programación Java, en el cual la liberación de espacio se realiza en intervalos definidos internamente por la Máquina Virtual de Java (JVM por sus siglas en inglés).

Es importante mencionar que además de los parámetros evaluados en esta prueba, muchos otros influyen (de menor manera) en los recursos necesarios por el programa para llevar a cabo una simulación.

4.2 Análisis de las señales

La prueba más importante que se llevo a cabo sobre el sistema tiene que ver con el análisis de las señales generadas por la simulación. En esta prueba se compararon señales reales obtenidas en experimentos con aquellas arrojadas por el sistema.

En la figura 4.2 se muestra una gráfica que muestra tanto los registros de experimentos con gatos como de la salida del simulador. En el inserto A, las señales en color negro muestran la actividad eléctrica registrada con un arreglo de multielectrodos, posicionado entre las láminas lumbar 4 y sacro 1 de la medula espinal de un gato. A estas señales se les conoce como Potenciales de Dorso de la Médula (PDMs). En rojo y azul se muestra la actividad electroneurográfica de dos músculos de las extremidades posteriores del gato; en este caso las señales corresponden a los músculos Gastrocnemio (extensor) y Tibial Anterior (flexor) respectivamente. En el inserto B, se encuentran las salidas del simulador. Las señales en negro corresponden a la actividad de los objetos llamados “CPG+”, que se comportan como electrodos haciendo un promedio de la actividad de las poblaciones neuronales debajo de ellos. En rojo y azul se encuentran las señales de las motoneuronas “MNe0” (extensora) y “MNf0” (flexora) respectivamente.

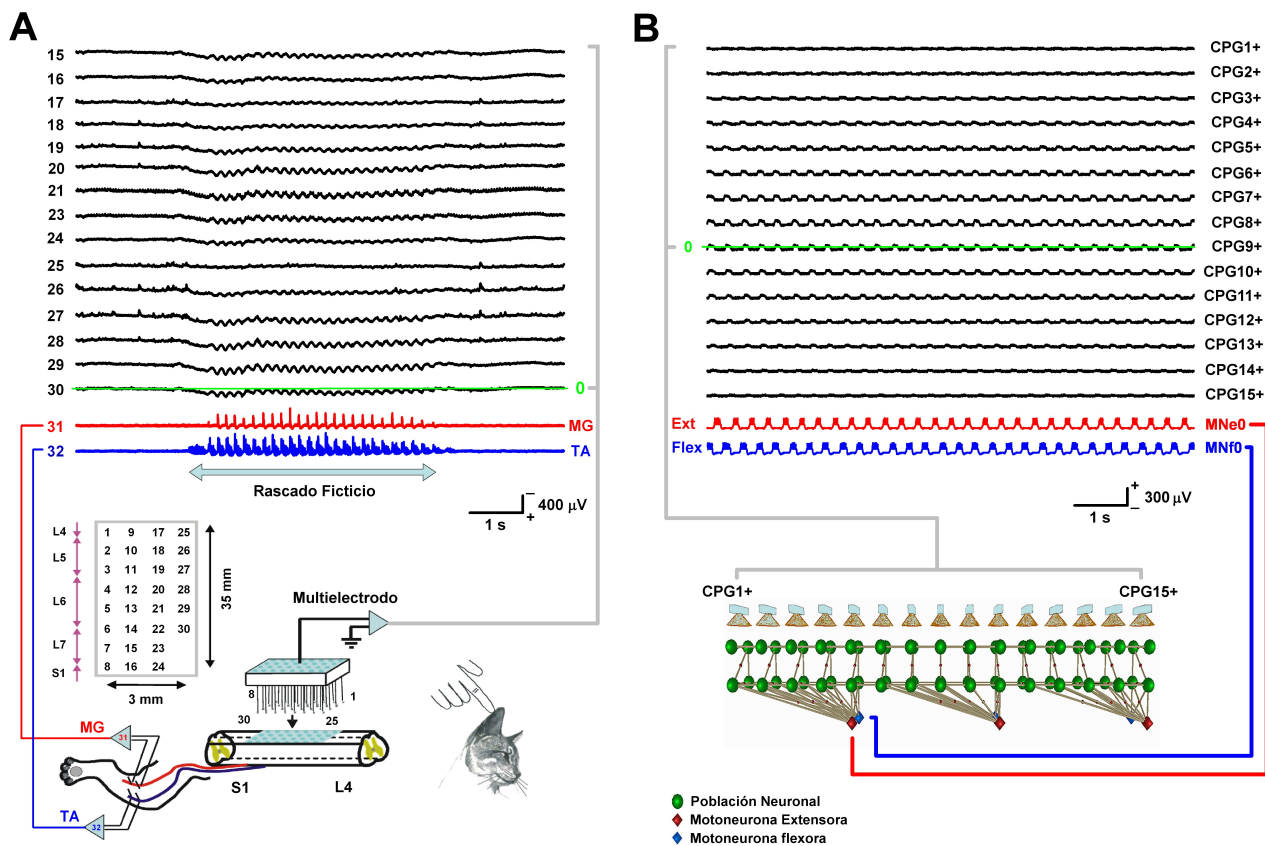


Figura 4.2: Comparación entre PDM, en el gato, y la salida del simulador. En **A** se muestra el arreglo de multielectrodos para el registro simultaneo de PDMs y electroneurogramas de los nervios extensor (GM) y flexor (TA) durante rascado ficticio. El rectángulo indica las posiciones de los electrodos de donde los registros de PDMs (15-30, señales en negro) fueron obtenidos. Las señales roja y azul muestran la actividad electroneurográfica de los músculos MG y TA respectivamente. Note la forma sinusoidal de las ondas del PDM y los potenciales lentos positivos asociados con el disparo tónico inicial de TA durante el episodio de rascado derecho. A todas las señales se les aplicó un filtro pasa-banda a DC-500 Hz. En **B** se muestran las señales correspondientes a los objetos CPG1+ a CPG15+ de la red CPG simulada. Las señales en rojo y azul corresponden al primer par de motoneuronas de la red; una flexora (MNf0) y otra extensora (MNe0). Imagen y texto modificados de (Cuellar et al., 2009).

Para comparar cuantitativamente estas señales primero se calculó la frecuencia de la onda sinusoidal característica de episodios de rascado. La figura 4.3 muestra la comparación del espectro de potencia de las señales reales para episodios de rascado (A) contra el espectro de potencia del objeto "CPG4+" del simulador (B). Ambos espectros muestran un pico de potencia alrededor de los 3 y 4 Hz.

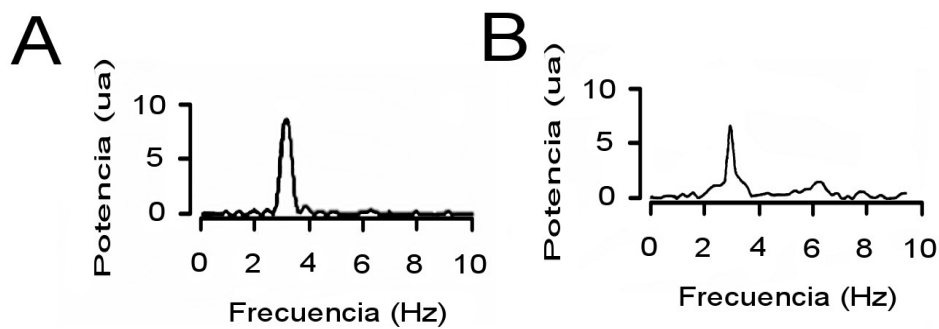


Figura 4.3: Comparación de los espectros de potencia de las señales. **A**, los espectros de potencia promedio de los electrodos 12 o 20 (rascado izquierdo y derecho, respectivamente), **B**, espectro de potencia del objeto “CPG4+”. Texto e imagen modificados de la tesis del M.C. Jesús Ángel Tapia López (Tapia-López, 2008).

La segunda comparación consistió en calcular la relación que existe entre la sinusoide del PDM y la actividad electroneurográfica de los músculos o, en el caso de las simulación, la relación entre las señales de los objetos “CPG+” y las señales de las motoneuronas. Para llevar a cabo las mediciones correspondientes se utilizó el método explicado en la figura 4.4, y los resultados obtenidos, se muestran en la figura 4.5.

Estas mediciones fueron efectuadas, en los registros reales de episodios de rascado, a todas las señales de la matriz de electrodos contra las señales de la actividad electroneurográfica de los nervios flexor y extensor, y en la simulación, a todas las señales de los objetos “CPG+” contra las señales de las motoneuronas en la primera posición.

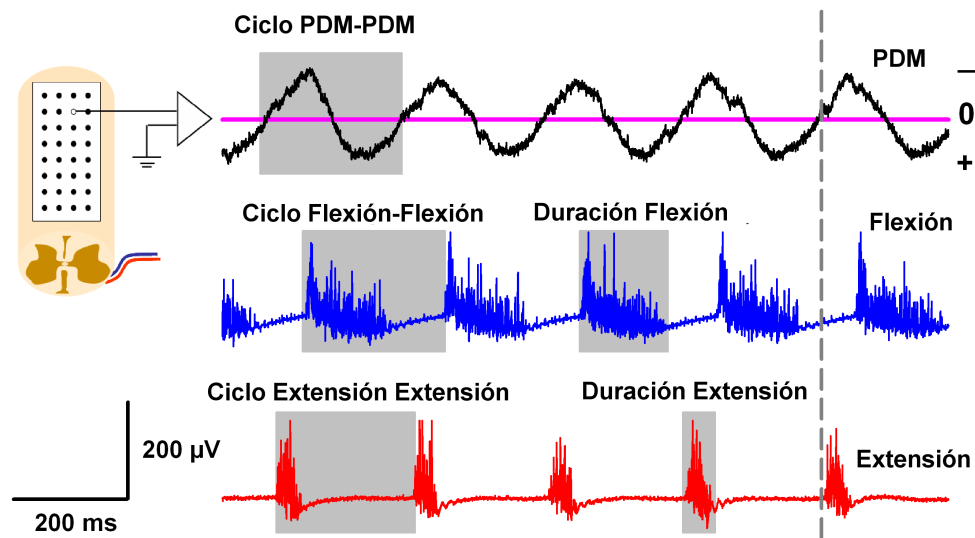


Figura 4.4: Descripción ilustrada de las mediciones efectuadas. La duración del ciclo de las ondas del PDM (ciclo PDM-PDM) fueron medidas como la ocurrencia completa de una onda positiva y otra negativa al cruce por cero. La duración del ciclo de la flexión (ciclo Flexión-Flexión) fue medido desde el inicio de la ráfaga flexora hasta el inicio de las siguiente ráfaga flexora. La duración del ciclo de la extensión (ciclo Extensión-Extensión) fue medido desde el inicio de la ráfaga extensora hasta el inicio de la siguiente ráfaga extensora. La duración de las ráfagas para la flexión y la extensión fueron medidas como se indica, cubriendo sólo la zona con actividad de espigas. Imagen y texto modificados de (Cuellar et al., 2009).

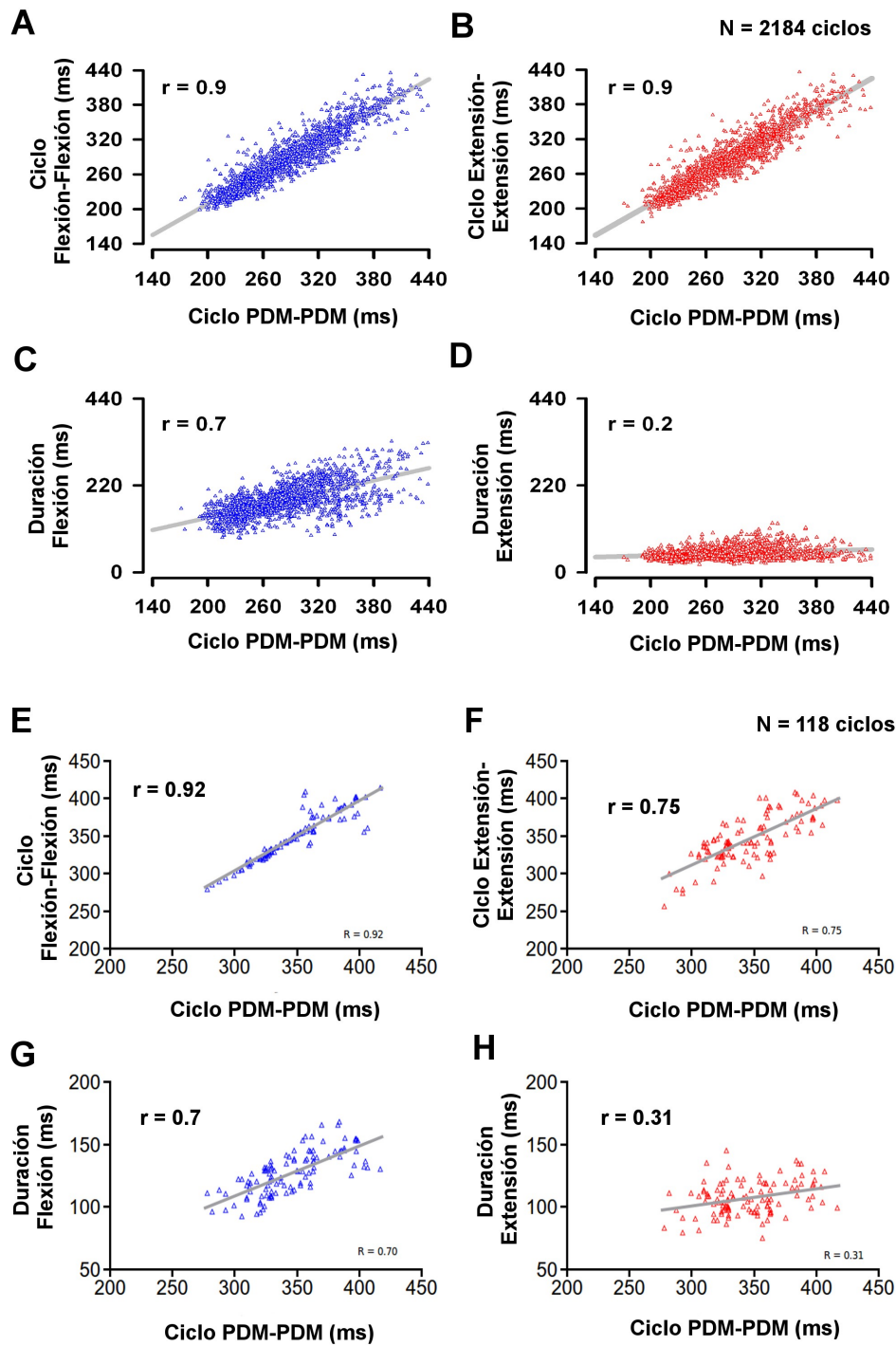


Figura 4.5: Mediciones de correlación entre la duración de los ciclos de las señales del PDM contra la actividad electroencefalográfica de los nervios. **A**, el ciclo PDM-PDM asociado al rascado contra el ciclo Flexión-Flexión del nervio TA. **B**, misma gráfica que en **A** pero para el ciclo Extensión-Extensión del nervio MG. **C**, **D**, las mismas gráficas que en **A** y **B** pero para la duración de la flexión y la duración de la extensión respectivamente. Los datos fueron obtenidos de ~2000 ciclos medidos en episodios de rascado en 22 gatos. A las señales sinusoidales del PDM se les aplicó un filtro pasa-banda de 1-70 Hz. **E**, **F**, **G**, **H** muestran los resultados de las mediciones aplicadas a las señales obtenidas con el simulador; corresponden a las gráficas **A**, **B**, **C**, **D**, respectivamente.

En ambos casos, los gráficos muestran que existe una alta correlación entre el ciclo de la senoide del PDM y los ciclos de la actividad electroneurográfica de los nervios, mientras que al relacionar estos mismos ciclos de la senoide del PDM contra la duración de la descarga de los nervios, la correlación disminuye considerablemente.

El rascado posee un claro comportamiento asimétrico que provoca esta predominancia del hemicentro flexor sobre el extensor. En el caso de la locomoción, los ciclos muestran un mayor equilibrio entre hemisferios, que podría ser causado por una arquitectura simétrica de CPG. Actualmente se está trabajando la posibilidad de reproducir episodios de rascado por medio del simulador; en entregas posteriores de este proyecto se mostrarán las conclusiones de estas pruebas.

4.3 Caso de uso: usando CPG Network

Ejecutar CPG Network por primera vez

La figura 4.6 muestra una imagen de la ventana principal de CPG Network. Una vez ejecutado el programa, esta ventana se mantendrá activa en la mayoría del proceso de modelado y simulación de la red CPG.

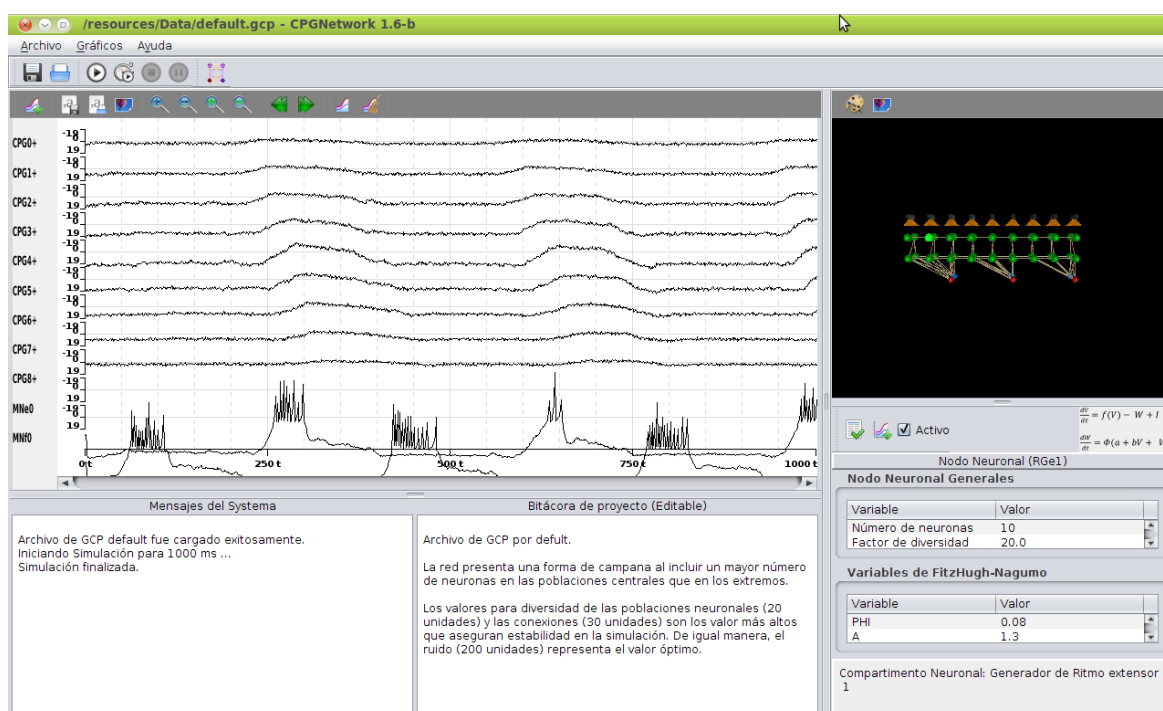


Figura 4.6: Vista de la ventana principal de CPG Network

Como se puede notar, la interfaz principal está compuesta de cuatro paneles centrales, una barra de menús y una caja de herramientas. En cualquier momento, el usuario puede acceder a cualquiera de las funciones que ofrece el programa ya sea por la

barra de menús o por la caja de herramientas. A continuación se explica brevemente el uso de los cuatro paneles:

- El panel de señales (arriba e izquierda) es el encargado de graficar las señales correspondientes a los objetos de la red CPG. Inicialmente se encuentra vacío, pero se puede seleccionar la función Agregar Señales en la caja de herramientas de este panel para escoger las señales que se deseen incluir en la gráfica.
- El panel de texto (abajo e izquierda) mostrará mensajes importantes al usuario. Este panel, en realidad se divide en dos áreas de texto: la de mensajes del sistema y la de bitácora del proyecto. Los mensajes del sistema son alertas que el sistema envía al usuario en distintas situaciones durante el proceso de modelado y simulación de una red. La bitácora del proyecto es un área de texto editable por el usuario; en ella se puede escribir cualquier nota importante que acompañe a la red para entenderla más fácilmente.
- El panel de configuración (abajo y derecha) es el encargado de desplegar información relacionada con los objetos de la red CPG. Por medio de este panel también es posible modificar y actualizar cualquier parámetro tanto de la red como de todos sus objetos.
- El panel de visualización (arriba y derecha) muestra una representación 3D de la red CPG que se esté modelando. Por medio de este panel se puede visualizar el resultado del proceso de modelado interactivamente y en tiempo real. Al seleccionar cualquiera de los objetos en el dibujo 3D, éstos serán enviados al panel de configuración para poder observar o modificar sus parámetros.

Hacer una primera simulación y agregar un conjunto de señales al panel de visualización, hará que la pantalla se vea casi igual a la de la figura 4.6.

- Para simular la red simplemente debe oprimirse el botón Simular la Red en la caja de herramientas principal.
- Para agregar un conjunto de señales al panel de señales debe oprimirse el botón Agregar Señales de la caja de herramientas del panel de señales y oprimir aceptar en la ventana emergente que fue desplegada.

De esta manera podemos rápidamente observar el funcionamiento y las herramientas que ofrece el CPG Network a primera vista.

El proyecto inicial

El proyecto por “default” para el CPG Network es el que se carga siempre al iniciar el programa. Este proyecto se encuentra almacenado en los archivos internos del programa, y jamás podrá ser modificado por un usuario.

La figura 4.6 muestra una vista de este proyecto predefinido, el cual despliega un texto informativo en el panel de bitácora de proyecto que dice lo siguiente:

Archivo de GCP por default.

La red presenta una forma de campana al incluir un mayor número de neuronas en las poblaciones centrales que en los extremos.

Los valores para diversidad de las poblaciones neuronales (20 unidades) y las conexiones (30 unidades) son los valor más altos que aseguran estabilidad en la simulación. De igual manera, el ruido (200 unidades) representa el valor óptimo.

Si se observa el panel de visualización es posible notar notar que la red tiene 8 CPG's conectados longitudinalmente. Lo que el texto del proyecto por "default" nos quiere decir es que cada uno de esos CPG's tiene un número distinto de neuronas en sus poblaciones neuronales; siendo mayor el número para las poblaciones del centro que para las poblaciones de los extremos. Este fenómeno puede observarse más claramente en el panel de señales. Si observamos las señales generadas por los electrodos (tienen nombres como CPG1+), nos daremos cuenta que el electrodo central (CPG4+ en este caso) presenta la señal con valores más grandes en comparación con los demás electrodos.

El segundo párrafo en la bitácora del proyecto nos habla de valores para parámetros específicos de la red. Si el usuario lo desea, se puede aventurar a modificar estos parámetros desde ahora, pero se recomienda leer previamente las demás secciones de este documento antes de empezar.

Ahora bien, si el usuario se encuentra modelando una red y desea guardar los avances que lleva hasta el momento, puede oprimir el botón Guardar Archivo CPG en la caja de herramientas principal y seleccionar un nombre de archivo y ubicación para almacenar el proyecto actual. De la misma manera puede cargar archivos de proyecto (*.gcp) o incluso archivos de señales (*.sig) e imágenes.

Animando la red CPG

Una de las funciones más vistosas que ofrece el CPG Network es la de animar la red. Teniendo una red CPG modelada, y algunas señales añadidas al panel de señales, esta función anima el dibujo del panel de visualización mostrando el comportamiento de cada uno de los objetos de la red. En la figura 4.7 se muestra una imagen de la ventana que se despliega al iniciar la función de animación.

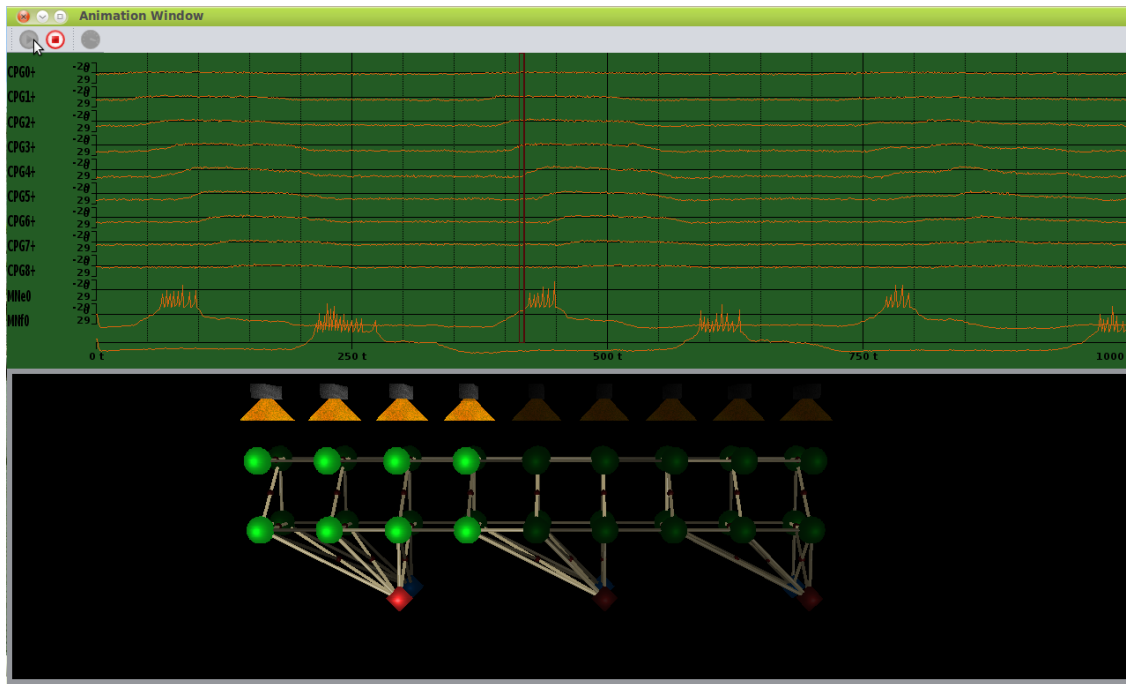


Figura 4.7: Vista de la ventana de animación

Para acceder a esta ventana se debe oprimir el botón Cambiar a Modo Animación en la caja de herramientas principal. De este modo, la ventana principal se ocultará y en su lugar aparecerá la ventana de animación.

Como puede observarse, tanto el panel de señales como el panel de visualización son reutilizados en versiones reducidas para generar esta ventana. Al oprimir en el botón *Iniciar Animación*, en la caja de herramientas de esta ventana aparecerá una línea roja sobre el panel de señales que indicará la posición en el tiempo de la animación en curso. En ese momento, los objetos del panel de visualización comenzarán a cambiar sus colores; los colores más claros representan mayores valores eléctricos de cada objeto, por consiguiente, un nodo neuronal muy oscuro nos dice que el potencial de campo de la población neuronal contenida en este nodo es muy bajo.

Conclusiones

En este trabajo se logró construir una aplicación que permite a científicos del área de neurociencias, en específico del Laboratorio de Neurofisiología Integrativa, modelar y simular las redes neuronales naturales conocidas como Generador Central de Patrones (CPG por sus siglas en inglés).

El programa presentado simula una red CPG de doble capa imitando la arquitectura propuesta por Brownston y Wilson, y organizada longitudinalmente como lo propone la arquitectura de la Unidad Generadora de Ráfagas. De este modo, el programa es capaz de aumentar o disminuir el número de CPG's unitarios, llegando a un máximo de 50 estructuras acopladas entre sí.

Los modelos neuronales aplicados en la red CPG, FitzHugh-Nagumo e Izhikevich, si bien constituyen sistemas de ecuaciones diferenciales ordinarias, ofrecen un modo realista (en comparación a otros modelos más complejos) para llevar a cabo la simulación computacional de neuronas naturales. El programa presentado es capaz de simular hasta 1 000 neuronas con estos modelos en un tiempo razonable. Esto abrió las puertas para diseñar un programa en donde el modelado de la red fuese completamente interactivo.

Para la integración numérica de ecuaciones diferenciales se optó por usar el método Runge-Kutta de cuarto orden. Durante el proceso de diseño se llevaron a cabo pruebas con diversos métodos de integración numérica, las cuales mostraron que el mejor método (en una combinación de velocidad contra precisión) para resolver los modelos utilizados es el de la familia de métodos Runge-Kutta.

Se logró diseñar e implementar una interfaz de usuario intuitiva, que incluye gráficos en dos y tres dimensiones así como otros objetos de interacción. Todos los parámetros, tanto de la red como de los modelos neuronales, pueden ser modificados por el usuario a través de la ventana principal del programa. Mientras el usuario modela la red es capaz de observar una representación en tercera dimensión y en tiempo real del estado de la misma. Y para observar los resultados, se ofrece una ventana que muestra la gráfica (voltaje / tiempo) de todos los objetos que constituyen la red.

Con el programa desarrollado en la presente tesis se logró simular, de manera muy realista, la actividad eléctrica tipo sinusoidal que generan las neuronas de la médula espinal del gato

Considero que este trabajo fue de gran importancia para mi crecimiento profesional. Durante mi servicio social y, posteriormente, en mi estancia de tesis, tuve la oportunidad de sumergirme en el área fascinante de las neurociencias. Gracias a la paciente ayuda de los integrantes del Laboratorio de Neurofisiología Integrativa, aprendí no sólo conceptos y definiciones del área, sino también cómo es el proceso que se debe seguir para llevar a cabo una investigación científica.

De este modo destaco, de manera principal, el concepto de trabajo en equipo como

una herramienta medular para alcanzar buenos resultados. Esta experiencia me hizo notar que los proyectos interdisciplinarios dependen, en especial, de una buena comunicación y del trabajo en equipo, lo cual es fundamental en computación por tratarse de un área inherentemente interdisciplinaria.

Perspectivas

A pesar de la capacidad de configuración que presenta el programa CPG Network, se pueden resaltar algunas deficiencias que posee, sobre todo en el proceso de diseño y modelado de una red CPG. Algunas de estas deficiencias son:

- Sólo se pueden crear redes CPG de dos capas: generador de ritmo y formador de patrones.
- Las conexiones entre poblaciones neuronales están prediseñadas para imitar la arquitectura de doble capa de Brownston y Wilson.
- El modelo por compartimentos que se usa es capaz de representar a una neurona como dos compartimentos: soma y axón.
- Hasta ahora, sólo dos modelos neuronales han sido implementados: FitzHugh-Nagumo e Izhikevich.

Al notar estas deficiencias, se sugieren algunas mejoras que podrían aplicarse al programa para hacerlo más completo y versátil:

- En un intento por generalizar el tipo de redes CPG que el programa es capaz de simular, se podría implementar por separado un editor de redes. Este módulo debería ofrecer al usuario una interfaz adecuada para modelar una red en forma de grafo bidireccional acíclico y almacenarla en un archivo. De este modo el programa principal podría leer esta información, e interpretarla como una arquitectura de red que posteriormente se calibraría en la interfaz que ahora existe.
- Así como otros paquetes de software científico para el modelado de redes neuronales, sería interesante darle al usuario la opción de configurar el número de compartimentos en los cuales una neurona se divide. Esta mejora, probablemente es la que más tiempo de diseño e implementación requeriría, sin embargo, el modelo de compartimentos es muy usado hoy y existe suficiente bibliografía que explica los detalles de su uso e implantación en programas de simulación. Ver el artículo “Simulation of networks of spiking neurons: a review of tools and strategies” (Brette et al., 2007) para mayor información acerca de este tema.
- Sería de vital importancia implementar algunos otros modelos neuronales y aplicarlos a la arquitectura de red propuesta en el programa. Un ejemplo inmediato de algún otro modelo neuronal es el conocido como Hodgkin-Huxley, y como se mencionó previamente, el modelo de FitzHugh-Nagumo es una reducción de éste; programas como NEURON o GENESIS, lo usan por “default” para llevar a cabo las simulaciones. Ya sean modelos más complejos o no, se podría incluirlos en el programa y efectuar una serie de pruebas que arrojen conclusiones sobre los beneficios o desventajas de un modelo sobre otro.

Como cualquier software productivo, el programa que se presenta aquí muestra deficiencias que pueden ser trabajadas en etapas subsecuentes del mantenimiento del mismo. Particularmente, las siguientes versiones del CPG Network podrían ofrecer mejoras como las siguientes:

- Incluir objetos gráficos más complejos. Uno de los factores más importantes para el éxito de un paquete de software es la facilidad de uso. Desde que se comenzó el diseño de este programa, se pensó en ofrecer interfaces intuitivas y amigables, sin embargo, este propósito nunca puede ser cumplido al cien por ciento. Pensando en esto, se tienen algunas ideas de mejoras a los objetos gráficos de la interfaz, como pueden ser: incluir una ventana que muestre todas las conexiones (tanto entrantes como salientes) de un compartimento del programa; agregar funcionamiento de tipo drag-and-drop (recoge y coloca) a todas las ventanas de la interfaz.
- Compatibilidad con otros paquetes de software científico. Dado que el trabajo de los científicos por lo general implica el uso de diversas herramientas, tanto en la etapa de recolección de datos como en la de análisis de los mismos, estas herramientas deben ser capaces de interactuar entre sí. Esta interacción se puede lograr con mejoras, tales como ofrecer gestión de diversos formatos de archivo o crear interfaces en hardware para efectuar recolección y presentación de datos.
- Agregar funciones de análisis de señales digitales integradas en el programa. Por lo general, los científicos usan un número restringido de procedimientos para el análisis de señales. Algunos de los que podrían incluirse en el programa son: mostrar información sobre una función como frecuencia o amplitud máxima y mínima; calcular el espacio de frecuencias de una función; comparar dos funciones por medio de correlación estadística, análisis de covarianza o correlación cruzada.

I. Integración numérica para ecuaciones diferenciales ordinarias

El área de las ecuaciones diferenciales es una de las más estudiadas por matemáticos en los últimos años; su uso compete a casi cualquier área de estudio en las ciencias modernas. Como lo describen textualmente Boyce y DiPrima: “Muchos de los principios y leyes que rigen el comportamiento del mundo natural involucran tasas a las cuales las cosas cambian. Cuando se expresa en términos matemáticos, las relaciones (entre cosas) se convierten en ecuaciones y las tasas de cambio en derivadas. Las ecuaciones que contienen derivadas se conocen como ecuaciones diferenciales” (Boyce & DiPrima, 1998).

Para la resolución de ecuaciones diferenciales existe una gran variedad de métodos analíticos, por medio de los cuales es posible encontrar una solución exacta del sistema propuesto. Sin embargo, la complejidad a la cual pueden llegar algunos sistemas ha hecho necesario el uso de métodos numéricos que sean fácilmente ejecutados en una computadora.

El primer método numérico desarrollado para encontrar soluciones aproximadas de ecuaciones diferenciales fue el propuesto por el matemático suizo Leonhard Paul Euler alrededor del año 1768, y que hoy es conocido por el nombre de método de Euler o método de la línea tangente.

Para explicar este método, considérese el problema con valor inicial siguiente

$$\frac{dy}{dt} = f(t, y) \quad y(t_0) = y_0 \quad a$$

el propósito será conseguir valores aproximados de la función solución $y = \Phi(t)$ a distintos intervalos de t . Para esto, se sabe que la función solución pasa por el punto (t_0, y_0) y que su recta tangente en este punto es $f(t_0, y_0)$. Con esto se puede deducir que la recta tangente de la función solución en el punto (t_0, y_0) está dada por

$$y_1 = y_0 + f(t_0, y_0)(t_1 - t_0) \quad b$$

en donde es el siguiente valor de t definido en los intervalos. Como no se sabe el valor de la función solución en el punto t_1 , se puede simplemente utilizar la aproximación y_1

calculada en el paso anterior para encontrar la aproximación del siguiente punto

$$y_2 = y_1 + f(t_1, y_1)(t_2 - t_1). \quad c$$

Así, se puede reescribir la ecuación 2.b, para encontrar la fórmula general del método de Euler

$$y_{n+1} = y_n + f(t_n, y_n)(t_{n+1} - t_n) \quad n=0,1,2,\dots \quad d$$

Finalmente, si se asume que hay un tamaño de paso uniforme h entre los puntos $t_0, t_1, t_2, \dots$, entonces $t_{n+1} = t_n + h$ para toda h y obtenemos la fórmula

$$y_{n+1} = y_n + f(t_n, y_n)h \quad n=0,1,2,\dots \quad e$$

A partir de esta fórmula es sencillo deducir el algoritmo que pudiera ejecutar este método en una computadora. En la figura 1 se muestra dicho algoritmo y el gráfico correspondiente a un paso en la ejecución del mismo.

Para analizar el rendimiento de este método, y en general de cualquier método numérico, se tienen dos herramientas básicas: el cálculo del error por redondeo y el del error por truncamiento local. El error por redondeo está relacionado con el tipo de computadora y el esquema de aritmética usados al implementar el algoritmo. Por otro lado, el error por truncamiento local está más relacionado con el algoritmo entre sí, y está asociado con el hecho de que cada paso del algoritmo se calcula con un valor aproximado “arrojado” por el paso anterior.

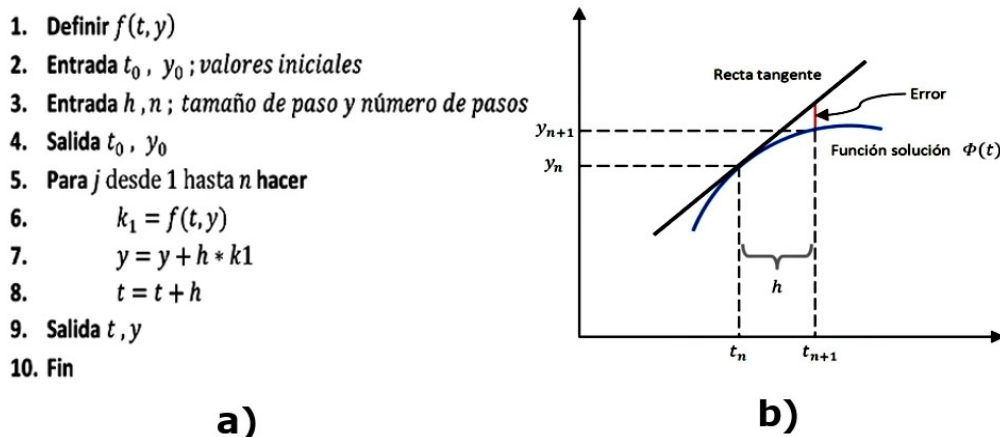


Figura 1. Método de Euler para la resolución numérica de ecuaciones diferenciales. En a) los pasos del algoritmo para efectuar el método. En b) se muestra gráficamente la aplicación del método.

No se ahondará más en el tema del cálculo de los distintos tipos de errores, pero cabe señalar que el método de Euler tiene un error por truncamiento local proporcional a

h^2 . Esto quiere decir que si se disminuye h en un factor proporcional a $1/2$, entonces, el error se verá reducido en $1/4$, y así sucesivamente.

Siendo el método de Euler, el método pionero para la aproximación de soluciones de ecuaciones diferenciales, es también el más simple y menos eficiente. En la actualidad existen diversas mejoras al método de Euler así como muchos otros métodos propuestos con el fin de minimizar el tiempo de ejecución y maximizar la precisión de los algoritmos.

Un ejemplo de estos métodos mejorados es el de Runge-Kutta, nombrado así en honor a sus creadores Carl David Runge y M. Wilhelm Kutta. Este método tiene un error de truncamiento proporcional a h^5 , lo cual indica que es dos ordenes de magnitud más preciso que el método de Euler.

El método de Runge-Kutta hace un promedio de los valores de $f(t, y)$ en diferentes puntos del intervalo $t_n \leq t \leq t_{n+1}$. Está definido por la fórmula

$$y_{n+1} = y_n + h \left(\frac{k_{n1} + 2k_{n2} + 2k_{n3} + k_{n4}}{6} \right), \quad f$$

donde

$$k_{n1} = f(t_n, y_n),$$

$$k_{n2} = f\left(t_n + \frac{1}{2}h, y_n + \frac{1}{2}hk_{n1}\right),$$

$$k_{n3} = f\left(t_n + \frac{1}{2}h, y_n + \frac{1}{2}hk_{n2}\right),$$

g

$$k_{n4} = f\left(t_n + h, y_n + hk_{n3}\right),$$

siendo k_{n1} el valor en el extremo izquierdo del intervalo y k_{n4} el del extremo derecho.

Para implementar este método, simplemente se requiere sustituir los pasos 6 y 7 del algoritmo de Euler (figura 1). Con estas modificaciones, el algoritmo de Runge-Kutta queda como se muestra en la figura 2.

Además de los dos algoritmos mencionados, Euler y Runge-Kutta, existen muchos otros algoritmos diseñados con el propósito de mejorar la precisión de las aproximaciones. Cabe señalar que, tanto estos dos métodos, como todas sus modificaciones (método de Heun, método regresivo de Euler, métodos de Euler y Runge-Kutta con variación de tamaño de paso), pertenecen al grupo llamado métodos de la clase Runge-Kutta (Boyce & DiPrima, 2004).

1. Definir $f(t, y)$
2. Entrada t_0, y_0 ; *valores iniciales*
3. Entrada h, n ; *tamaño de paso y número de pasos*
4. Salida t_0, y_0
5. Para j desde 1 hasta n hacer
 6. $k_1 = f(t, y)$
 7. $k_2 = f(t + 0.5 * h, y + 0.5 * h * k_1)$
 8. $k_3 = f(t + 0.5 * h, y + 0.5 * h * k_2)$
 9. $k_4 = f(t + h, y + h * k_3)$
 10. $y = y + (h/6) * (k_1 + 2 * k_2 + 2 * k_3 + k_4)$
 11. $t = t + h$
12. Salida t, y
13. Fin

Figura 2 Algoritmo de Runge-Kutta. Se reemplazan los pasos 6 y 7 del algoritmo de Euler por los pasos 6-10 de esta imagen.

II. Programación Extrema

La metodología de software Programación Extrema es una de muchas metodologías conocidas como Desarrollo Ágil (Beck et al., 2001), que enfatiza el trabajo en equipo y los grupos colaborativos conjuntos que incluyen tanto a los desarrolladores como a los clientes y administradores. Según Don Wells, autor de la metodología, XP mejora un proyecto de software en cinco aspectos esenciales: la comunicación, la simpleza, la retroalimentación, el respeto y el coraje.

A continuación se listan las seis fases que comprenden el ciclo de vida de un proyecto bajo la metodología XP (Letelier & Penadés, n.d.). La figura 3 muestra una imagen con el diagrama de flujo que explica estas fases.

- **Fase I, Exploración:** los clientes plantean a grandes rasgos las historias de usuario que son de interés para la primera entrega del producto. Al mismo tiempo el equipo de desarrollo se familiariza con las herramientas, tecnologías y prácticas que se utilizarán en el proyecto. Se prueba la tecnología y se exploran las posibilidades de la arquitectura del sistema construyendo un prototipo.
- **Fase II, Planificación de la entrega:** el cliente establece la prioridad de cada historia de usuario, y correspondientemente, los programadores realizan una estimación del esfuerzo innecesario de cada una de ellas. Se toman acuerdos sobre el contenido de la primera entrega y se determina un cronograma en conjunto con el cliente. La planificación se puede realizar basándose en el tiempo o el alcance. La velocidad del proyecto es utilizada para establecer cuántas historias se pueden implementar antes de una fecha determinada o cuánto tiempo tomará implementar un conjunto de historias.
- **Fase III, Iteraciones:** incluye varias iteraciones sobre el sistema antes de ser entregado. El Plan de Entrega está compuesto por iteraciones de no más de tres semanas. En la primera iteración se puede intentar establecer una arquitectura del sistema que pueda ser utilizada durante el resto del proyecto. Al final de la última iteración el sistema estará listo para entrar en producción. Los elementos que deben tomarse en cuenta durante la elaboración del Plan de la Iteración son: historias de usuario no abordadas, velocidad del proyecto, pruebas de aceptación no superadas en la iteración anterior y tareas no terminadas en la iteración anterior.
- **Fase IV, Producción:** requiere de pruebas adicionales y revisiones de rendimiento antes de que el sistema sea trasladado al entorno del cliente. Al mismo tiempo, se deben tomar decisiones sobre la inclusión de nuevas características a la versión actual, debido a cambios durante esta fase.
- **Fase V, Mantenimiento:** mientras la primera versión se encuentra en producción, el proyecto XP debe mantener el sistema en funcionamiento al mismo tiempo que desarrolla nuevas iteraciones. Para realizar esto se requiere de tareas de soporte para el cliente. De esta forma, la velocidad de desarrollo puede bajar después de la puesta del sistema en producción.

- **Fase VI, Muerte del Proyecto** ; es cuando el cliente no tiene más historias para ser incluidas en el sistema. Esto requiere que se satisfagan las necesidades del cliente en otros aspectos como rendimiento y confiabilidad del sistema. Se genera la documentación final del sistema y no se realizan más cambios en la arquitectura.

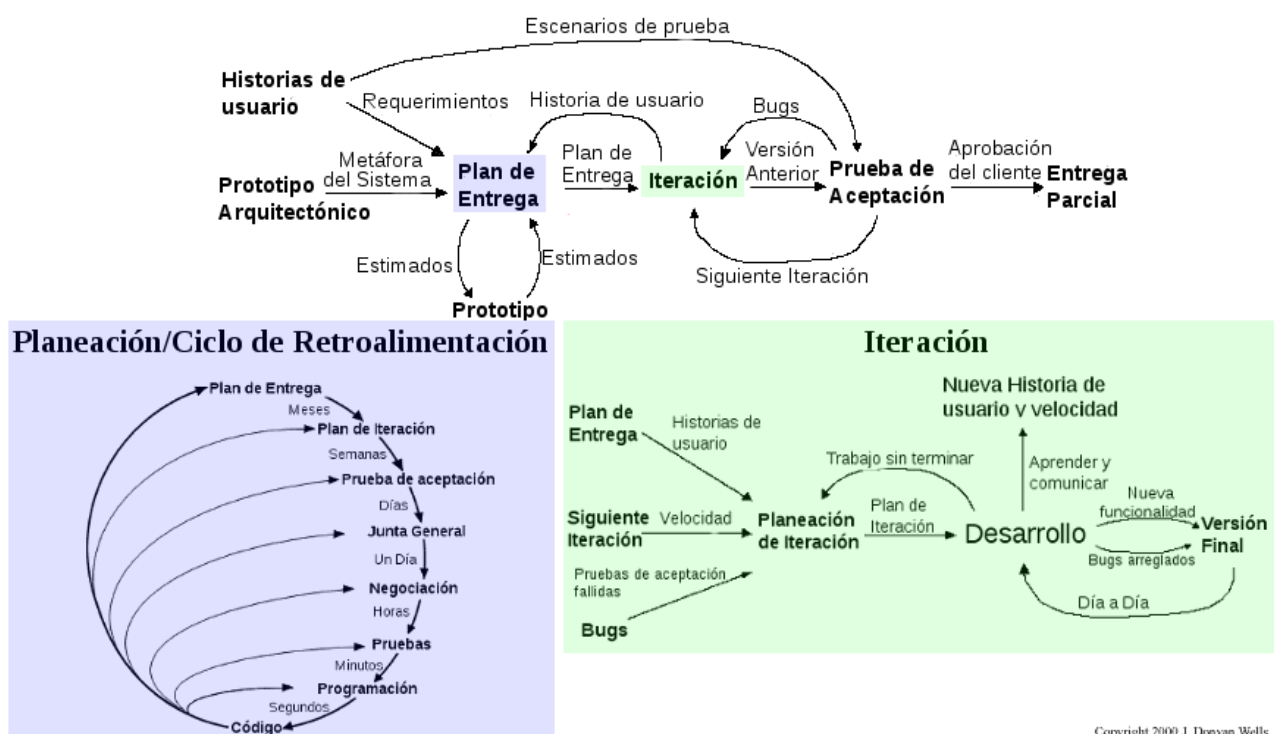


Figura 3. Diagrama de flujo de la metodología Programación Extrema. Imágen modificada y traducida del documento “Extreme Programming: A gentle introduction” (Donovan Wells (Don Wells), 2009).

Bibliografía

- Abbott, L. (1999). Lapicque's introduction of the integrate-and-fire model neuron (1907). *Brain Research Bulletin*, 50(5-6), 303-304. doi: 10.1016/S0361-9230(99)00161-6.
- Banks, J. (1998). *Handbook of Simulation; Principles, Methodology, Advances, Applications, and Practice*. (J. Banks). John Wiley & Sons, Inc.
- Beck, K., Beedle, M., Van Bennekum, A., Cockburn, A., Cunningham, W., Fowler, M., et al. (2001). Manifesto for Agile Software Development. Retrieved from <http://agilemanifesto.org/>.
- Boyce, W. E., & DiPrima, R. C. (1998). *Ecuaciones Diferenciales y Problemas con Valores en la Frontera* (4 ed., p. 751). Mexico D.F.: Editorial Limusa.
- Boyce, W. E., & DiPrima, R. C. (2004). *Elementary Differential Equations and Boundary Value Problems , 8th Edition, with ODE Architect CD*. Wiley. Retrieved from <http://www.amazon.com/Elementary-Differential-Equations-Boundary-Architect/dp/0471433381>.
- Brette, R., Rudolph, M., Carnevale, T., Hines, M., Beeman, D., Bower, J. M., et al. (2007). Simulation of networks of spiking neurons: a review of tools and strategies. *Journal of computational neuroscience*, 23(3), 349-98. doi: 10.1007/s10827-007-0038-6.
- Brown, G. (1911). The Intrinsic Factors in the Act of Progression in the Mammal. *Proceedings of the Royal Society of London. Series B, ume*(572), 308-319. Retrieved from <http://adsabs.harvard.edu/abs/1911RSPSB..84..308B>.
- Brownstone, R. M., & Wilson, J. M. (2008). Strategies for delineating spinal locomotor rhythm generating networks and the possible role of Hb9 interneurons in rhythmogenesis. *Brain Res. Rev.*, 57, 64-76.
- Chung, C. A. (2003). *Simulation Modeling Handbook: A Practical Approach (Industrial and Manufacturing Engineering Series)*. CRC. Retrieved from <http://www.amazon.com/Simulation-Modeling-Handbook-Manufacturing-Engineering/dp/0849312418>.
- Churchland, P. S., Koch, C., & Sejnowski, T. J. (1993). What is computational neuroscience? In E. L. Schwartz, *Computational Neuroscience* (pp. 46-55). Cambridge, MA, USA: MIT Press.
- Cuellar, C. A., Tapia, J. A., Juárez, V., Quevedo, J., Linares, P., Martínez, L., et al. (2009). Propagation of sinusoidal electrical waves along the spinal cord during a fictive motor task. *The Journal of neuroscience : the official journal of the Society for Neuroscience*, 29(3), 798-810. doi: 10.1523/JNEUROSCI.3408-08.2009.
- Destexhe, A., Mainen, Z. F., & Sejnowski, T. J. (1994). An efficient method for computing synaptic conductances based on a kinetic model of receptor binding. *Neural Computation*, 6(1). Retrieved from <http://portal.acm.org/citation.cfm?id=188071>.

Donovan Wells (Don Wells), J. (2009). Extreme Programming: A gentle introduction. Retrieved from <http://www.extremeprogramming.org/>.

Grillner, S. (2006). Biological pattern generation: the cellular and computational logic of networks in motion. *Neuron*, 52(5), 751-66. doi: 10.1016/j.neuron.2006.11.008.

Izhikevich, E. M. (2003). Simple model of spiking neurons. *IEEE transactions on neural networks / a publication of the IEEE Neural Networks Council*, 14(6), 1569-72. doi: 10.1109/TNN.2003.820440.

Izhikevich, E. M. (2006). *Dynamical Systems in Neuroscience: The Geometry of Excitability and Bursting (Computational Neuroscience)*. The MIT Press. Retrieved from <http://www.amazon.com/Dynamical-Systems-Neuroscience-Excitability-Computational/dp/0262090430>.

Kandel, E. R., Schwartz, J. H., & Jessell, T. M. (1996). *Essentials of Neural Science and Behavior*. McGraw-Hill/Appleton & Lange. Retrieved from <http://www.amazon.com/Essentials-Neural-Science-Behavior-Kandel/dp/0838522459>.

Kandel, E., Schwartz, J., & Jessell, T. (2000). *Principles of Neural Science*. McGraw-Hill Medical. Retrieved from <http://www.amazon.com/Principles-Neural-Science-Eric-Kandel/dp/0838577016>.

Keynes, R. D., & Aidley, D. J. (2001). *Nerve and Muscle (Studies in Biology)*. Cambridge University Press. Retrieved from <http://www.amazon.com/Nerve-Muscle-Studies-Biology-Keynes/dp/0521805848>.

Letelier, P., & Penadés, C. (n.d.). Metodologías ágiles para el desarrollo de software : eXtreme Programming (XP). *Development*.

McCrea, D., & Rybak, I. (2008). Organization of mammalian locomotor rhythm and pattern generation. *Brain research reviews*, 57(1), 134-46. doi: 10.1016/j.brainresrev.2007.08.006.

Perkel, D. H. (1993). Computational neuroscience: scope and structure. In E. L. Schwartz, *Computational Neuroscience* (pp. 38-45). Cambridge, MA, USA: MIT Press.

Perkel, D. H., Mulloney, B., & Budelli, R. W. (1981). Quantitative methods for predicting neuronal behavior. *Neuroscience*, 6(5), 823-837. Retrieved from <http://www.sciencedirect.com/science/article/B6T0F-4846J9C-6W/2/2bf517496b2870031eeb7ba58f4a3bdd>.

Perros, H. (2008). *Computer Simulation Techniques: The definitive introduction!*. Computer Science Department NC State University Raleigh, NC. Retrieved from <http://www.csc.ncsu.edu/faculty/perros/simulation.pdf>.

Purves, D., & Williams, S. M. (2001). *Neuroscience*. Sinauer Associates. Retrieved from <http://www.amazon.com/Neuroscience-Book-CD-ROM-Dale-Purves/dp/0878937420>.

Pérez, T., Tapia, J. A., Mirasso, C. R., García-Ojalvo, J., Quevedo, J., Cuellar, C. A., et al. (2009). An intersegmental neuronal architecture for spinal wave propagation under deletions. *The Journal of neuroscience : the official journal of the Society for Neuroscience*, 29(33), 10254-63.

doi: 10.1523/JNEUROSCI.1737-09.2009.

Smith, R. (1999). Simulation: The Engine Behind The Virtual World. *Simulation 200 Series*. Retrieved from <http://www.modelbenders.com/papers/sim2000/SimulationEngine.PDF>.

Tapia-López, J. A. (2006). Generación de potenciales eléctricos espontáneos en neuronas artificiales y su acción sobre un circuito electrónico que simula al reflejo monosináptico de la médula espinal.

Tapia-López, J. A. (2008). Distribución de interneuronas espinales del generador del ritmo y del generador de patrones del rascado ficticio: Resultados experimentales y de modelación.

Thompson, J. R. (1999). *Simulation: A Modeler's Approach (Wiley Series in Probability and Statistics)*. Wiley-Interscience. Retrieved from <http://www.amazon.com/Simulation-Modelers-Approach-Probability-Statistics/dp/0471251844>.

Wolfson, M. M., & Pert, G. J. (1999). *An Introduction to Computer Simulation*. Oxford University Press, USA. Retrieved from <http://www.amazon.com/Introduction-Computer-Simulation-M-Wolfson/dp/019850425X>.