

Tiempos de Disparo y Acoples Sinápticos
en Neuronas Hodgkin Huxley
y Neuronas de Disparo (IF).

JUAN PABLO CALDERÓN A.

Proyecto de Grado para optar al título de Físico
Director: CARLOS ARTURO ÁVILA, PhD.

UNIVERSIDAD DE LOS ANDES
FACULTAD DE CIENCIAS
DEPARTAMENTO DE FÍSICA
BOGOTÁ
JULIO 2002

Índice General

1	Introducción	3
I	Teoría	5
2	Neuronas	6
2.1	Potencial en la membrana	6
2.2	Potencial de acción	8
3	Modelos de Neuronas	10
3.1	Hodgkin y Huxley	10
3.1.1	Primeras Consideraciones	10
3.1.2	Estados de actividad e inactividad	11
3.1.3	Modelo Completo	14
3.2	Modelos de disparo (Integrate and Fire)	15
3.3	Modelo FitzHugh-Nagumo	17
4	Acople entre neuronas	19
4.1	Sinapsis	19
4.2	Red Neuronal	21
II	Simulaciones con neuronas HH e IF.	23
5	Simulación con una neurona HH	24
5.1	Parámetros de la neurona.	24
5.1.1	Parámetros para acople de dos neuronas Hodgkin-Huxley	25
5.2	Entradas de Corriente	26
5.3	Salidas de Voltaje	27

<i>ÍNDICE GENERAL</i>	2
5.4 Respuesta de una neurona HH.	27
5.4.1 Entrada constante y con modulación periódica.	27
5.4.2 Entrada con modulación senoidal	33
5.4.3 Entrada con Modulación Caótica	34
6 Ajuste de una neurona de disparo (IF).	44
7 Acople entre neuronas	50
7.1 Modelo	50
7.2 Acople de dos neuronas Hodgkin Huxley.	52
7.2.1 Comportamiento transiente y estable	53
7.3 Acople de N neuronas.	57
7.3.1 Pesos sinápticos	57
7.3.2 Patrones	59
7.3.3 Recuperación de patrones	60
7.3.4 Recuperación de patrones con una entrada perfecta $\zeta = \xi^{(\mu)}$	61
7.3.5 Recuperación de patrones con entrada imperfecta $\zeta \neq \xi^{(\mu)}$ (ruido)	69
7.3.6 Número de Neuronas activas M	78
7.3.7 Número de neuronas N	82
8 Conclusión	84
A Programas C.	89
A.1 Neurona Hodgkin Huxley	89
A.1.1 Entradas sinápticas	89
A.1.2 Función sináptica.	94
A.1.3 Runge-Kutta de cuarto orden para una neurona HH.	95
A.2 Neurona IF	98
A.3 Acople de Neuronas	108
A.3.1 Acople dos Neuronas	108
A.3.2 Acople 100 Neuronas	112

Capítulo 1

Introducción

Las neuronas en nuestro cerebro son responsables de codificar las características de un estímulo. Últimamente se han hecho muchos estudios teóricos sobre codificación y decodificación de la información en el potencial de acción. Modelos más recientes basados en resultados experimentales, afirman que la rata promedio de disparo en una neurona motora o sensorial depende del estímulo aplicado. Por otro lado hay evidencia científica que muestra que el tiempo preciso y la organización de los potenciales de acción es fundamental para transmitir esta información. Existe mucha controversia acerca de como se lleva a cabo la codificación de la información, por ejemplo, se sabe que el sistema visual humano clasifica patrones en 150 mseg y tiene por lo menos 10 procesos sinápticos desde la retina hasta el cerebro temporal, muy similar a como funciona en los micos macacos[24, 25]. Un proceso como este tendría una frecuencia de disparo menor a 100 Hz, lo que implica que cada neurona contribuye con uno o dos disparos y por lo tanto no se puede calcular una rata promedio de disparo. Esto sugiere que se deben mirar los tiempos precisos de disparo, más no el promedio.

Desde que en 1952 Hodgkin y Huxley crearon un modelo para el potencial de acción en un axón de calamar gigante[2], las propiedades de este modelo se han estudiado bastante y ha sido adoptado para estudios de sistemas biológicos. El modelo de Hodgkin-Huxley es un modelo no lineal de cuatro ecuaciones diferenciales acopladas, por lo que no es sencillo de tratar. Recientemente se han sugerido modelos más sencillos como lo son el modelo de disparo[23] y el modelo de FighzHugh-Nagumo[8, 9].

En este trabajo se estudia la respuesta de una neurona tipo Hodgkin-Huxley con diferentes trenes de pulsos de entrada. Los trenes de pulsos generan entradas

sinápticas de corriente y son modulados en el tiempo tanto determinística como caóticamente. Se estudia que tan factible es reproducir la salida de una neurona Hodgkin-Huxley (HH) con una neurona de disparo (IF).

Desde los trabajos de Hopfield[11] se han estado estudiando las redes neuronales como memorias asociativas utilizando la teoría de la mecánica estadística de Spin-Glasses. Estas teorías son utilizadas para analizar las redes de tipo Hopfield que sirven como modelos de memoria y aprendizaje en el cerebro. El modelo de Hopfield utiliza como variable dinámica la rata de disparo de las neuronas y almacena la información en las conexiones entre la neurona pre-sináptica y post sináptica. Se están estudiando las oscilaciones sincronizadas en las neuronas, experimentos fisiológicos sugieren también que además de la rata promedio de disparos en las neuronas es importante observar la distribución espacio temporal de los disparos. La distribución exacta de los tiempos de disparos al parecer tiene más información que la rata promedio de disparos. Teniendo en cuenta esto para construir una red neuronal, necesariamente necesitamos neuronas que disparen como son las de HH las de FighzHugh-Nagumo o las neuronas IF.

Se implementó una red neuronal tipo Hopfield que funciona como una memoria asociativa de neuronas Hodgkin-Huxley con acoples sinápticos retrasados en el tiempo. En la memoria asociativa los patrones son almacenados en los pesos sinápticos (o conexiones entre las neuronas) de la red, por medio de una regla Hebbiana propuesta por Willshaw[20]. Los patrones almacenados son cadenas binarias expresadas por un vector $\xi^{(\mu)} = \{\xi_j^{(\mu)} \mid j = 1 - N\}$ donde $\mu = 1 - P$ y $\xi_j^{(\mu)} = 0(1)$ refiriéndose a la actividad e inactividad de las neuronas, N es el número de neuronas, y P es el número de patrones que se desea almacenar. Cualquiera de los patrones almacenados puede ser recuperado cuando la red llega a uno de sus estados atractores estables. Para llegar a uno de estos estados se requiere tener como entrada a la red un patrón similar al que se desea recuperar.

Se estudia la capacidad de recuperar exitosamente los patrones almacenados variando el número de neuronas en la red, y cambiando la cantidad de neuronas activas en los patrones, como también la tolerancia al ruido.

Se han hecho muchos estudios de las reglas de aprendizaje de la red, para crear modelos biológicamente plausibles. Se estudia la capacidad de memoria con dos reglas de aprendizaje diferentes.

Parte I

Teoría

Capítulo 2

Neuronas

Las neuronas vienen de diferentes clases y tamaños, por eso los parámetros y propiedades que se describen a continuación pertenecen a un tipo particular de neurona, es complicado hablar de una neurona típica, pero propiedades como la excitabilidad, desarrollo de un potencial de acción, acoples sinápticos y otros son considerados generalmente propiedades de las neuronas.

En la figura 2.1 se puede observar una neurona, con todas sus partes fundamentales. Contiene un núcleo y otros organelos dentro de lo que se denomina el cuerpo de la neurona, también tiene unas dendritas y un axón. Las dendritas reciben la información de otras neuronas, esta información se interpreta en el cuerpo de la célula y luego se transmite por medio de un impulso nervioso a través del axón. El axón tiene unas terminales que se juntan con las dendritas de otra célula y por aquí se transmite el impulso nervioso a la siguiente neurona.

2.1 Potencial en la membrana

Toda la superficie de la célula está forrada por una membrana que tiene aproximadamente 70 \AA de ancho. El interior de la célula tiene un potencial eléctrico menor que afuera, es decir, negativo, si el potencial afuera de la membrana es cero, dentro tenemos aproximadamente -70 mV .

Una propiedad muy importante de la membrana es que es selectivamente permeable a ciertos iones, especialmente a iones de Na^+ , K^+ y Cl^- . Sus concentraciones están dadas en la tabla 2.2. La ultima columna de la tabla 2.2 muestra la diferencia de potencial a través de la membrana debido a la diferencia

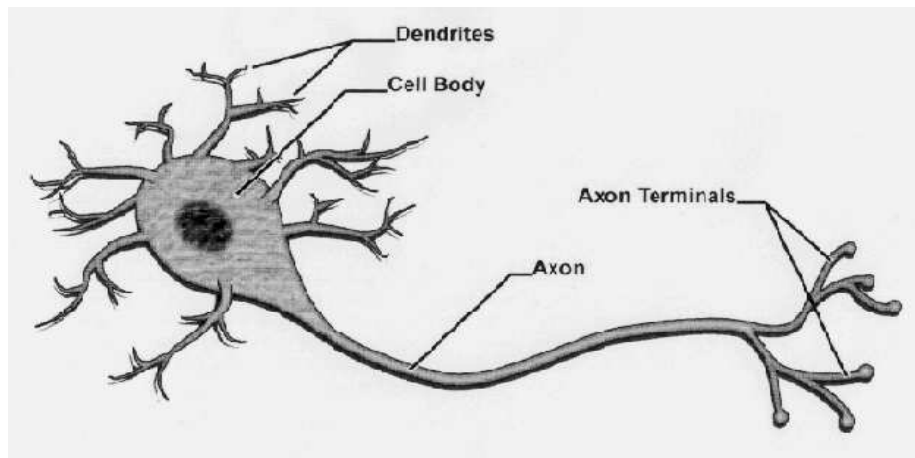


Figura 2.1: Diagrama de una neurona, con sus partes fundamentales.

ION	Concentracion afuera	Concentracion adentro	Potencial de Equilibrio
Na^+	150 mM	~ 15 mM	+60 mV
K^+	5.5 mM	150 mM	-90 mV
Cl^-	125 mM	9 mM	-70 mV

Tabla 2.2: Concentración de iones adentro y afuera de la membrana en una neurona en el estado de reposo,[1]. El potencial de equilibrio es calculado con la ecuación de Nerst.

de concentraciones dentro c_{in} y fuera de la membrana c_{out} . A este potencial también se le denomina potencial de equilibrio, aparece cuando la atracción debido a una diferencia de carga, compensa la atracción debido al gradiente de concentración, está dado por la ecuación de Nerst

$$V = \pm \frac{RT}{F} \ln \frac{c_{out}}{c_{in}} \quad (2.1)$$

donde R es la constante de los gases, F es la constante de Faraday y T la temperatura absoluta.

El potencial de equilibrio de los iones de cloro se acerca al potencial de la membrana en reposo, pero se tiene menos sodio y más potasio del que se necesita para mantener un equilibrio en la membrana. Se observa este desequilibrio por un proceso que se denomina transporte activo de iones.

El potencial de la membrana en el estado de reposo se denomina potencial de reposo y tiene un valor de $V_{rest} = -70$ mV.

Hodgkin y Huxley en 1952 construyeron un modelo[2] donde se describe la corriente a través de la membrana como una superposición de tres corrientes, estas corrientes están explicadas en detalle en la sección 3.1.

2.2 Potencial de acción

El potencial de acción es lo que comúnmente se llama impulso nervioso, sucede cuando la membrana polarizada inversamente se despolariza por pocos milisegundos, volviéndose más positiva en su interior que en su exterior. La forma del potencial de acción se puede apreciar en la figura 2.2.

En el interior de la membrana hay dos capas de lípidos que contienen proteínas, funcionan como bombas y canales de iones y son responsables de la existencia del potencial de reposo y del potencial de acción.

Las bombas de sodio y potasio mueven estos iones en contra del gradiente de concentración, concentra los iones K^+ dentro de la neurona y los iones Na^+ fuera de la neurona.

En ausencia de estímulo, algunos canales de potasio están abiertos, estos canales de iones permiten que iones K^+ escapen de la neurona, dejando atrás una carga neta negativa debido principalmente a la presencia de iones Cl^- , esto es el potencial de reposo.

El potencial de reposo de la neurona puede cambiar por que canales de iones activados por voltaje se abren y se cierran. Esto permite a la neurona

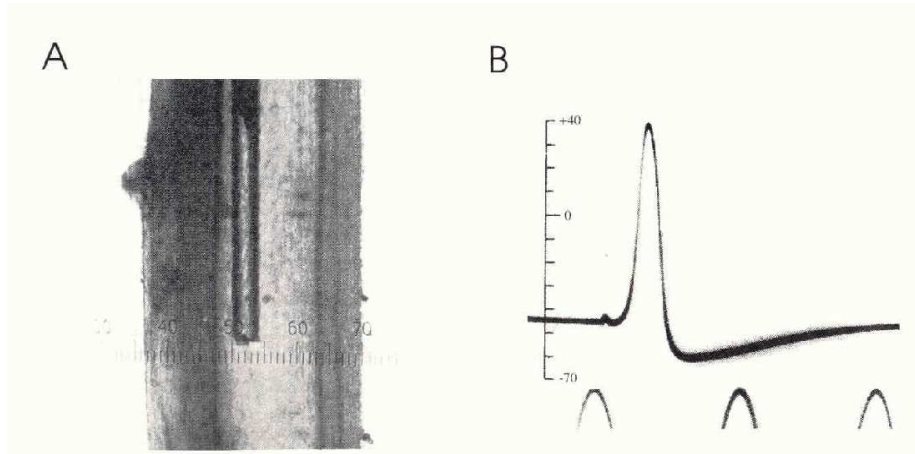


Figura 2.2: A) Medicion intracelular del potencial de acci3n en el ax3n de calamar gigante. B) Potencial de acci3n observado por Hodgkin y Huxley[3], la onda seno abajo tiene un periodo de 2 mseg.[4]

polarizarse o incluso hiper-polarizarse en respuesta a un est3mulo externo. En ausencia de est3mulos algunos canales de potasio est3n abiertos y el potencial de la membrana es el potencial de reposo. En presencia de un est3mulo externo, algunos canales de sodio en la membrana se abren, permitiendo el paso de iones Na^+ lo que despolariza la membrana aumentando su potencial. Cuando el potencial de la membrana llega a un nivel umbral, unos canales de sodio activados por voltaje se abren, la carga de la membrana se invierte r3pidamente produciendo un pico de voltaje, esto es el potencial de acci3n, luego los canales de Na^+ se cierran y los canales de K^+ activados por voltaje se abren, produciendo una r3pida polarizaci3n e incluso una hiper-polarizaci3n de la membrana, a continuaci3n los canales activados por voltaje abiertos se cierran y la membrana vuelve al potencial de reposo.

El potencial de acci3n viaja a trav3s de la neurona gracias a que corrientes internas despolarizan regiones adyacentes del ax3n hasta el nivel umbral, creando potenciales de acci3n, de esta forma el potencial de acci3n se regenera constantemente a trav3s del ax3n.

Para una descripci3n m3s detallada del potencial de acci3n, ver [4, 5, 6].

Capítulo 3

Modelos de Neuronas

3.1 Hodgkin y Huxley

Hodgkin y Huxley en 1952[2] llevaron a cabo varios experimentos con un axón gigante de calamar, el cual medía aproximadamente medio milímetro de diámetro. Bloqueando el paso de ciertos iones usando agentes farmacológicos, lograron obtener una descripción matemática del comportamiento no lineal del potencial en la membrana. El modelo describe la excitación y conducción del impulso nervioso en una neurona. Significó el premio Nobel en 1963.

3.1.1 Primeras Consideraciones

Hodgkin y Huxley representaron su modelo como un circuito eléctrico (ver figura 3.1). Hay dos fuerzas responsables por el flujo de iones a través de la membrana: Una fuerza generada por un gradiente químico, debido a la diferencia de concentración a ambos lados de la membrana, y una fuerza eléctrica generada por un gradiente eléctrico, por la presencia de partículas cargadas a ambos lados de la membrana. El potencial de equilibrio es el resultado de igualar estos dos gradientes. El modelo de Hodgkin y Huxley asume que el voltaje en la membrana se debe principalmente a una corriente capacitiva y a una corriente iónica.

$$I_m(t) = I_{\text{ionica}} + C_m \frac{dV(t)}{dt} \quad (3.1)$$

La corriente debido a las contribuciones iónicas es el resultado de sumar tres corrientes; la primera, generada por el paso de iones de sodio; la segunda, por

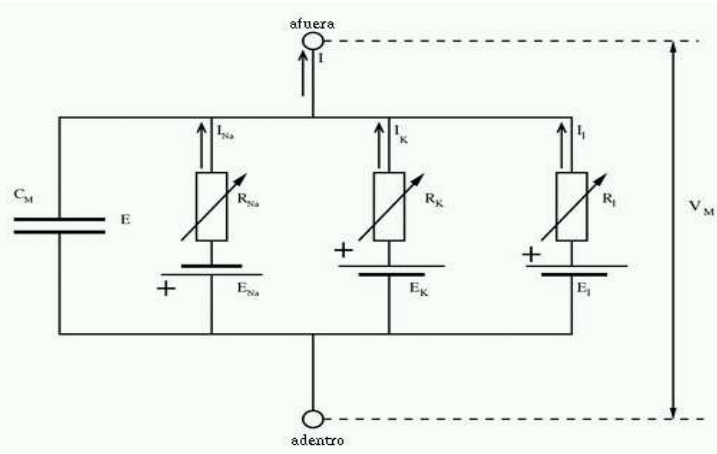


Figura 3.1: Circuito Equivalente para el potencial de la membrana en un axón, modelo de Hodgkin y Huxley.[5]

iones de potasio y la otra debido al resto de iones que atraviesan la membrana, como son clorhídricos y bicarbonatos.

$$I_{ionica} = I_{Na} + I_K + I_{fuga} \quad (3.2)$$

Cada una de las corrientes iónicas obedece la ley de Ohm, es decir, está linealmente relacionada con el potencial de equilibrio E_{Na} , E_K (ecuación 2.1) y el voltaje a través de la membrana V_m .

$$\begin{aligned} I_{Na} &= G_{Na} * (V_m - E_{Na}) \\ I_K &= G_K * (V_m - E_K) \\ I_{fuga} &= G_f * (V_m - V_{rest}) \end{aligned} \quad (3.3)$$

Las tres conductancias son independientes, G_{Na} y G_k dependen del voltaje en la membrana, mientras que G_f no.

3.1.2 Estados de actividad e inactividad

Las conductancias para cada una de las corrientes iónicas fueron expresadas por Hodgkin y Huxley[2] como una conductancia máxima multiplicada por un coeficiente que representa el número de canales abiertos en la membrana. Los

coeficientes son funciones introducidas para reproducir la dinámica observada experimentalmente en las conductancias. Cada uno de los iones tiene sus respectivos canales que se abren y cierran para controlar el paso de iones por la membrana. Los canales se comportan de manera diferente para cada ion y dependen de los respectivos gradientes de concentración.

Hodgkin y Huxley tomaron datos experimentales e introdujeron ciertas funciones dependientes del voltaje describiendo las ratas de activación e inactivación de las compuertas. Estas ratas eran diferentes para cada una de las especies de iones y fueron determinadas ajustando las funciones a resultados experimentales.

Corriente debido a iones de potasio I_K

Hodgkin y Huxley modelaron la conductancia del potasio como:

$$G_k = \overline{G_k} n^4 \quad (3.4)$$

obteniendo una corriente

$$I_k = \overline{G_k} n^4 (V - E_k) \quad (3.5)$$

donde la conductancia máxima es $\overline{G_k} = 36 \text{ mS/cm}^2$ y la fuente del potasio es $E_k = -12 \text{ mV}$ relativo al potencial de equilibrio del axón. n es el coeficiente de activación del canal del potasio, un número adimensional que varía entre 0 y 1. En la ecuación 3.4, n se interpreta como la probabilidad de que la compuerta se encuentre abierta, es decir, activa, la probabilidad de que se encuentre inactiva sería entonces $1-n$. La ecuación 3.4 nos dice que para tener un canal abierto debemos tener cuatro compuertas en estado activo, n también se puede interpretar como la relación que existe entre el número de compuertas abiertas y compuertas cerradas. La corriente de potasio solo puede fluir si cuatro compuertas están abiertas.

Si se considera α_n como la rata de transiciones entre compuertas cerradas y compuertas abiertas, β_n como la rata de transiciones de compuertas abiertas a compuertas cerradas, se llega a la siguiente ecuación diferencial:

$$\frac{dn}{dt} = \alpha_n(1 - n) - \beta_n n \quad (3.6)$$

donde las ratas α_n y β_n son funciones dependientes del voltaje en la membrana, dadas por las siguientes expresiones:

$$\begin{aligned}\alpha_n(V) &= \frac{10 - V}{100(e^{(10-V)/10} - 1)} \\ \beta_n(V) &= 0.125e^{-V/80}\end{aligned}\quad (3.7)$$

donde V es el voltaje en la membrana del axón relativo al potencial de equilibrio, dado en milivoltios.

Hodgkin y Huxley hicieron estas aproximaciones incluyendo los valores numéricos en su artículo en 1952[2].

Corriente debido a iones de sodio I_{Na}

La corriente debido a iones de sodio es más compleja que la corriente de potasio. Hodgkin y Huxley tuvieron que introducir dos coeficientes en este caso, un coeficiente para la activación m y otro para la desactivación h del canal. La conductancia fue modelada como

$$\mathbf{G}_{Na} = \overline{G_{Na}} m^3 h \quad (3.8)$$

La corriente entonces sería

$$I_{Na} = \overline{G_{Na}} m^3 h (V - E_{Na}) \quad (3.9)$$

donde la conductancia máxima es $\overline{G_{Na}} = 120 \text{ mS/cm}^2$ y la fuente del sodio es $E_{Na} = -12 \text{ mV}$ relativo al potencial de equilibrio del axón. m y h son números adimensionales entre 0 y 1. La corriente de sodio depende de cuatro compuertas independientes tres m y una h . La probabilidad de que el canal esté activo es $m^3 h$, notar que h es la probabilidad que la compuerta inactivadora esté en su estado activador. Para que la corriente fluya tiene que haber tres m activos y un h inactivo. Estos coeficientes tienen ecuaciones diferenciales similares a las obtenidas para el coeficiente en la corriente de sodio.

$$\frac{dm}{dt} = \alpha_m(1 - m) - \beta_m m \quad (3.10)$$

$$\frac{dh}{dt} = \alpha_h(1 - h) - \beta_h h \quad (3.11)$$

Estas ratas también se determinaron ajustando los exponentes de los coeficientes a los resultados experimentales.

$$\alpha_m(V) = \frac{25 - V}{10(e^{(25-V)/10} - 1)} \quad (3.12)$$

$$\beta_m(V) = 4e^{-V/18} \quad (3.13)$$

$$\alpha_h(V) = 0.07e^{-V/20} \quad (3.14)$$

$$\beta_h(V) = \frac{1}{e^{(30-V)/10} + 1} \quad (3.15)$$

donde V es el voltaje en la membrana del axón relativo al potencial de equilibrio, dado en milivoltios.

Corriente de fuga I_f

Además de las conductancias de sodio y potasio también tenemos la conductancia de fuga G_f , que no depende del voltaje en la membrana y se mantiene constante durante todo el proceso. El valor medido por Hodgkin-Huxley fue de $G_f = 0.3 \text{ mS/cm}^2$ y corresponde a una resistencia de $R_f = 3333 \Omega * \text{cm}^2$. A esta conductancia también se le asocia un potencial de reversa. La corriente de fuga se debe a todos los iones diferentes a sodio y potasio que cruzan la membrana.

3.1.3 Modelo Completo

Hodgkin-Huxley no midieron el valor de la fuente asociada a la corriente de fuga sino que lo calcularon explícitamente tal que la corriente total a través de la membrana fuera cero en el equilibrio. El valor obtenido por Hodgkin-Huxley fue de $V_{rest} = 10.613 \text{ mV}$.

La capacitancia asociada a la membrana es $C_m = 1 \mu\text{F/cm}^2$.

En el potencial de equilibrio, la resistencia efectiva debido a todas las contribuciones (fuga, potasio, y sodio) suma $857 \Omega * \text{cm}^2$.

La ecuación para todas las corrientes que fluyen a través de la membrana del axón es:

$$C_m \frac{dV}{dt} = \bar{G}_{Na} m^3 h (E_{Na} - V) + \bar{G}_k n^4 (E_k - V) + G_l (V_{rest} - V) + I_{ext}(t) \quad (3.16)$$

donde I_{ext} es una corriente externa que se aplica al axón. Esta ecuación diferencial no lineal junto con las tres ecuaciones diferenciales asociadas a los coeficientes de las conductancias (ecuaciones 3.6, 3.10 y 3.11) también depen-

dientes del voltaje en la membrana describen el modelo en cuatro dimensiones de Hodgkin-Huxley para el potencial de acción en el axón del calamar.

Los valores numéricos descritos en esta sección fueron tomados del libro de Koch[5] pero son muy parecidos a los valores descritos por Hodgkin y Huxley en su artículo[2].

3.2 Modelos de disparo (Integrate and Fire)

Los modelos de disparo (ver Koch[5]) son modelos sencillos que describen los aspectos básicos de excitabilidad en una neurona, se tiene una región donde el voltaje en la membrana es menor que el voltaje umbral, y generación de picos una vez se sobrepasa este umbral.

El modelo consiste en una capacitancia que almacena carga a medida que una entrada externa entrega corriente al interior de la membrana, una vez el voltaje de esta capacitancia llega al nivel umbral V_{th} , hay un disparo, es decir, un pico de voltaje. Existen modelos un poco más complejos con una resistencia conectada a tierra que modela la corriente de fuga a través de la membrana.

El modelo de disparo no reproduce la forma exacta del potencial de acción aunque la relación frecuencia - corriente con parámetros adecuados se puede aproximar a la que reproducirían modelos mucho más complicados.

La ecuación que rige el comportamiento de este modelo en la región antes del disparo es

$$C \frac{dV(t)}{dt} = I(t) \quad (3.17)$$

una vez el voltaje llega al nivel umbral $V(t) = V_{th}$ ocurre un disparo que se modela como una función delta $\delta(t - t_i)$ donde t_i es el instante en que $V(t_i) = V_{th}$. Inmediatamente después de este disparo toda la carga almacenada en la capacitancia se libera por medio de una conexión a tierra (ver figura 6), esto hace que el voltaje en la capacitancia también se vuelva cero. Es posible hallar recursivamente los tiempos en que ocurren los disparos integrando la ecuación 3.17.

$$\int_{t_i}^{t_{i+1}} I(t) dt = CV_{th} \quad (3.18)$$

Entre mayor sea la corriente que entra en el circuito mayor va a ser la frecuencia de disparo, es decir, más rápido se va a almacenar la carga en el

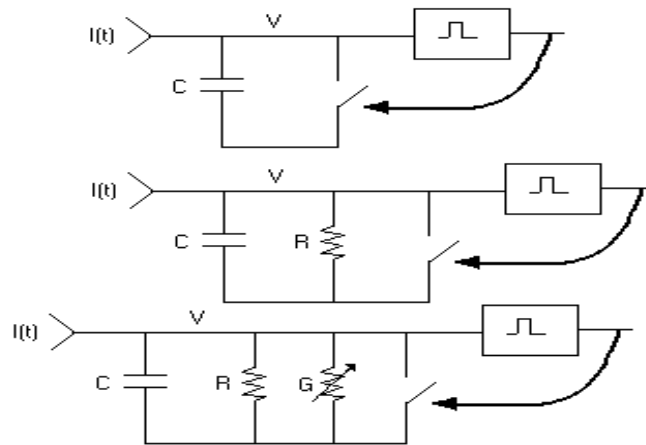


Figura 3.2: Circuito equivalente para los modelos de disparo.

condensador y más rápido el voltaje en la membrana va a llegar a su nivel umbral.

Corrientes pequeñas eventualmente van a causar disparos.

En este tipo de neuronas hace falta el periodo de refracción, la corriente debido a iones de sodio es la responsable de generar los picos y esta corriente necesita un tiempo para recuperarse de la inactivación. La corriente de potasio también afecta el tiempo entre disparos. Este tiempo de refracción se puede modelar con un intervalo de tiempo en el cual no llegue corriente a la capacitancia.

Un modelo más complejo incluye una resistencia de fuga en la región subumbral (ver figura 6). La ecuación diferencial que rige este comportamiento sería:

$$C \frac{dV(t)}{dt} + \frac{V(t)}{R} = I(t) \quad (3.19)$$

Si la corriente entrando a la neurona tiene la forma de un escalón constante en $t=0$, la solución para el voltaje de la membrana está dado por la ecuación

$$V(t) = IR(1 - e^{-t/\tau}) + V(t=0)e^{-t/\tau} \quad (3.20)$$

La neurona va a seguir este comportamiento mientras el voltaje a través de la membrana este por debajo de V_{th} , donde se genera un pico y el voltaje vuelve a cero. La mínima corriente necesaria para que ocurra un disparo sería $I_{th} = V_{th}/R$.

Hasegawa elaboro un modelo de disparo[7], que incluye un tiempo refractario para la neurona y reinicia automáticamente el potencial de la membrana al potencial de reposo.

3.3 Modelo FitzHugh-Nagumo

En la sección 3.1 se describe el modelo de Hodgkin - Huxley para la generación de potenciales de acción en un axón de calamar gigante, este modelo describe la dinámica del potencial de acción en términos de cuatro variables: El potencial en la membrana, y tres variables de estado (n, m y h), que determinan el estado de las conductancias de sodio y potasio responsables del potencial de acción. El modelo de Hodgkin y Huxley reproduce fielmente la realidad pero es un modelo de cuatro ecuaciones diferenciales acopladas, el modelo de FitzHugh[8] y Nagumo[9] reproduce lo esencial del modelo de Hodgkin y Huxley, oscilaciones neuronales periódicas como respuesta a una corriente, y es de segundo orden, en el modelo de FitzHugh-Nagumo se reducen estas cuatro variables a solo dos.

Observando cuidadosamente el comportamiento de las variables en el modelo de Hodgkin y Huxley como respuesta a un escalón de corriente se aprecia que hay una similitud entre el voltaje en la membrana $V(t)$ y la activación del sodio $m(t)$, también se observa una similitud entre la activación del potasio $n(t)$ y la desactivación del sodio $1 - h$. Estas dos primeras variables se unen para formar una sola variable de activación V y las segundas dos variables se unen también para formar una variable W que caracteriza el nivel de acomodamiento del sistema. Este nuevo sistema con solo dos variables se comporta muy parecido al modelo con cuatro variables de Hodgkin Huxley.

Históricamente las ecuaciones de este modelo tiene origen en un trabajo hecho por Vand er Pol (1926)[21], quien formuló un modelo de un oscilador no lineal para aplicarlo a un marca pasos.

Las ecuaciones del modelo son:

$$\begin{aligned}\dot{V} &= V - \frac{V^3}{3} - w + I \\ \dot{W} &= \phi(V + a - bW)\end{aligned}\tag{3.21}$$

donde \dot{V} se refiere a la derivada temporal de V (dV/dt). Los parámetros a, b y ϕ son a -dimensionales y positivos, se ven muchas versiones de estos parámetros[5,

8, 9, 15] pero unos valores que funcionan bastante bien son: $a=0.7$, $b=0.8$ y $\phi = 0.08$ (ver Koch[5]). La amplitud de ϕ determina que tan rápido la variable W cambia con respecto a V . Con un valor de $\phi = 0.08$, V cambia sustancialmente más rápido que W .

Capítulo 4

Acople entre neuronas

Las impresionantes propiedades del sistema nervioso, se deben a que tenemos muchas neuronas interactuando por medio de conexiones entre sí, estas interacciones procesan e integran información, crean comportamientos complejos, manejan conceptos complejos además de aprender y memorizar.

4.1 Sinapsis

Sinapsis es la unión entre neuronas, esta unión permite transmitir mensajes eléctricos o químicos de una neurona a otra, la neurona que transmite el mensaje se denomina neurona pre-sináptica y a la neurona a la que se le transmite el mensaje se le denomina neurona post-sináptica.

Una clásica sinapsis química es la unión neuromuscular, una sinapsis entre una neurona motora y una célula muscular, el neurotransmisor es acetilcolina que causa una despolarización en la membrana post-sináptica. Cuando un potencial de acción llega al borde de un axón se liberan neurotransmisores que se pegan a la membrana post-sináptica causando una despolarización. La membrana pre-sináptica tiene un canal activado por voltaje que no se encuentra en ninguna otra parte del axón, este canal permite el paso de iones Ca^{2+} . Cuando el potencial de acción llega al final del axón los canales de Ca^{2+} se abren, y como la concentración de Ca^{2+} es mayor afuera que adentro de la membrana pre-sináptica, iones de Ca^{2+} entran en la membrana. El incremento en la concentración de Ca^{2+} dentro de la célula pre-sináptica causa que unas vesículas que contienen acetilcolina se fusionen con la membrana y liberen todo su conte-

nido al espacio que hay entre la célula pre-sináptica y la célula post sináptica. Este espacio lleva el nombre de cavidad sináptica. Las moléculas de acetilcolina viajan a través de este espacio y se pegan a receptores en la célula post-sináptica causando que los canales de sodio se abran un poco despolarizandola. Un potencial de acción hace que cien vesículas liberen su contenido lo cual es suficiente para que la célula post-sináptica llegue al umbral y continúe la transmisión del impulso nervioso.

Una neurona puede tener muchas dendritas, por lo que muchos terminales de axones pueden tener sinapsis con estas dendritas y con el cuerpo de la neurona. La membrana pre-sináptica puede almacenar y liberar mas de un tipo de neurotransmisor, igualmente la membrana post-sináptica puede tener receptores para diferentes neurotransmisores, de esta forma recibe diferentes tipos de mensajes químicos. Dependiendo de la reacción de la neurona post-sináptica se dice que el proceso sináptico es excitador o inhibidor. Cuando la membrana post - sináptica se despolariza es excitador como en la unión neuromuscular, y cuando se hiper-polariza es inhibidor.

La liberación de un neurotransmisor en una sinapsis inhibitoria hace más difícil que se dispare un potencial de acción en la célula post-sináptica.

Cada neurona puede recibir más de 1000 entradas sinápticas de las cuales hay unas excitadoras y otras inhibidoras. Individualmente cada célula suma estas entradas y decide si se dispara un potencial de acción. Toda la información contenida en los miles de entradas que recibe la neurona se codifica en la razón de disparo de potenciales de acción. Los potenciales post-sinápticos excitadores e inhibidores se suman tanto temporal como espacialmente.

Además de la sinapsis química existe una sinapsis eléctrica. La separación entre una célula pre - sináptica y una célula post - sináptica es de 2 a 3 nm. Existen unas proteínas llamadas conexones que unen las dos células formando túneles moleculares. Iones y moléculas pequeñas pueden pasar de una célula a otra por medio de estos túneles. La transmisión eléctrica a través de estos túneles es muy rápida, y puede ser en ambas direcciones, es decir, estimulación en cualquiera de las neuronas puede generar un potencial de acción en la otra, mientras que la sinapsis química es más lenta y unidireccional. La sinapsis eléctrica no permite sumar temporalmente las entradas sinápticas, además estos túneles moleculares exigen una gran área de contacto entre las dos células lo que limita el número de entradas sinápticas, tampoco existe una hay eléctrica sinapsis inhibitora. Todo esto hace que la sinapsis eléctrica permita comunicarse más rápidamente pero no es buena para el proceso de integración y aprendizaje.

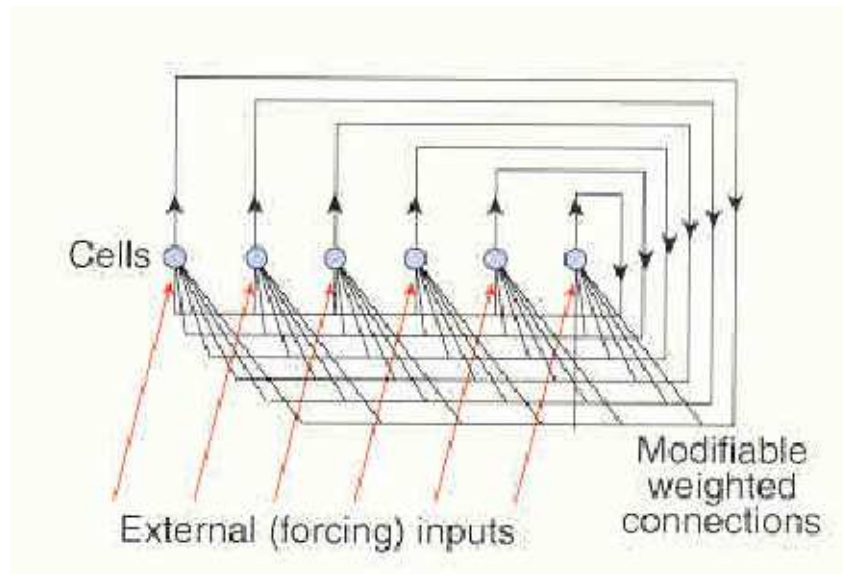


Figura 4.1: Forma generalizada de una red neuronal asociativa, consiste de una capa de células con salidas que se ramifican para proveer realimentación, estas conexiones sinápticas tienen peso, y ninguna célula se realimenta a ella misma, también hay una entrada externa a cada una de las células.[10]

4.2 Red Neuronal

Una red neuronal consiste de varias neuronas conectadas entre sí. Para este trabajo se van a tomar neuronas tipo Hodgkin Huxley con acoples sinápticos, tales como fueron descritos en las secciones y capítulos anteriores, se va a considerar que el impulso nervioso fluye en una sola dirección y se demora un tiempo en atravesar el axón de la neurona. Todas las neuronas están conectadas entre sí como muestra la figura 4.1. La salida de una neurona se realimenta en todas las neuronas excepto en ella misma como un pulso de corriente que se discutirá mas adelante.

En este trabajo se va a trabajar una red tipo Hopfield[11] como memoria asociativa[10], estas redes tienen las siguientes características: Las redes de Hopfield utilizan como variable dinámica la tasa de disparo de las neuronas, y la información está codificada en los tiempos de disparo, esta es almacenada en los acoples sinápticos W_{ij} entre neuronas. Los acoples W_{ij} son calculados mediante reglas sencillas y su valor representa que tan fuerte es la conexión entre la neurona i y la neurona j . Voy a considerar simetría en los acoples es

decir: $W_{ij} = W_{ji}$, pero esto no siempre es así.

Se construyó una red de N neuronas de tipo Hodgkin Huxley en la cual se almacenaban P patrones. Estas neuronas tienen los mismos parámetros que las neuronas utilizadas en el resto de las simulaciones. Están conectadas con acoplos sinápticos con tiempo de retraso.

Una memoria asociativa, asocia un patrón de entrada con un patrón idéntico de salida, calculando los pesos sinápticos con reglas Hebbianas (como se verá más adelante). Yoshioka trabaja una memoria asociativa para neuronas tipo FitzHugh-Nagumo[15], y Hasegawa trabaja una red asociativa con neuronas tipo Hodgkin Huxley[14].

Parte II

Simulaciones con neuronas

HH e IF.

Capítulo 5

Simulación con una neurona

HH

Este capítulo muestra como varía la respuesta de una neurona tipo Hodgkin Huxley a diferentes entradas de corriente. Las entradas son generadas por trenes de impulsos, para los cuales los intervalos entre disparos son modulados en el tiempo de diferentes maneras. El tren de impulsos genera entradas sinápticas de corriente, es decir, se asume que a la neurona entra una corriente que resultó de un proceso sináptico.

5.1 Parámetros de la neurona.

En las simulaciones se utilizan los parámetros dados en el modelo de Hodgkin y Huxley[2], para el cual las conductancias asociadas a las corrientes iónicas toman los siguientes valores:

$$\begin{aligned}g_{Na} &= 120 \text{ mS/cm}^2 \\g_k &= 36 \text{ mS/cm}^2 \\g_f &= 0.3 \text{ mS/cm}^2\end{aligned}\tag{5.1}$$

los potenciales de reposo según Hodgkin y Huxley son:

$$\begin{aligned}
V_{Na} &= 50 \text{ mV} \\
V_k &= -77 \text{ mV} \\
V_f &= -54.5 \text{ mV}
\end{aligned}
\tag{5.2}$$

y la capacitancia de la membrana es de $C = 1 \mu F$.

El valor típico para el máximo potencial a través de la membrana es $V_a = 30 \text{ mV}$, el potencial de la sinapsis es $V_{sin} = -50 \text{ mV}$ y $\tau = 2 \text{ msec}$ es una constante de tiempo relacionada con la función α que describe la sinapsis. Esta función se se discutirá más adelante. Estos valores son los mismos utilizados por Hasegawa en sus simulaciones[12].

Las condiciones iniciales que se utilizaron para las simulaciones del modelo de Hodgkin y Huxley son:

$$\begin{aligned}
V_j &= -65.025 \text{ mV} \\
m_j &= 0.053 \\
h_j &= 0.597 \\
n_j &= 0.317
\end{aligned}
\tag{5.3}$$

son la solución estable para el modelo de Hodgkin y Huxley y se hallaron corriendo el programa (Runge Kutta de cuarto orden) para una neurona tipo Hodgkin-Huxley sin entrada de corriente alguna. Estos parámetros se utilizan tanto para la simulación de una neurona como para la simulación 2, 100 y N neuronas acopladas.

5.1.1 Parámetros para acople de dos neuronas Hodgkin-Huxley

Los parámetros utilizados en las simulaciones son $A_s = g_s(V_a - V_s) = 40 \mu A/cm^2$ para $g_s = 0.5 \text{ mS/cm}^2$. Este valor se va a utilizar en todas las simulaciones, es el mismo valor utilizado por Hasegawa[7, 12, 13, 14], este valor se utiliza en la corriente sináptica.

5.2 Entradas de Corriente

La corriente de entrada I_{ext} que entra a la neurona está dada por la ecuación 3.16 y se asume proviene de una sinapsis. Esta función se compone de dos términos

$$I_{ext} = I_s + I_p \quad (5.4)$$

donde I_s es una corriente DC estática y I_p denota una corriente que es inducida por un tren de pulsos (ecuación 5.5). Se considera un tren de pulsos expresado por la siguiente función tipo delta:

$$U_i(t) = V_a \sum \delta(t - t_{in}) \quad (5.5)$$

El tiempo de disparo t_{in} para un n arbitrario está dado por:

$$t_{in+1} = t_{in} + T_{in}(t_{in}) \quad (5.6)$$

$$t_{i1} = 0 \quad (5.7)$$

Donde T_{in} es el intervalo entre dos disparos consecutivos y es función del tiempo t_{in} en que ocurrió el disparo anterior. En este capítulo se modulan los intervalos entre disparos T_{in} como constante, senoidalmente y finalmente caóticamente utilizando los modelos del atractor de Rossler y el famoso atractor de Lorenz[13].

El tren de pulsos dado por la ecuación 5.5 genera una corriente I_p , dada por

$$I_p(t) = g_{syn} \sum_n \alpha(t - t_{in})(V_a - V_{syn}) \quad (5.8)$$

donde g_{syn} es la conductancia para una sinapsis y V_{syn} es el potencial sináptico.

La función $\alpha(t)$ está definida como:

$$\alpha(t) = (t/\tau)e^{-t/\tau}\theta(t) \quad (5.9)$$

donde τ es la constante de tiempo asociada a la conducción sináptica y $\theta(t)$ es la función escalón. La función alfa tiene un valor máximo de $I_p^{max} = e^{-1}g_{syn}(V_a - V_{syn})$ en $t = t_{in} + \tau$ y tiene un ancho de banda de 2.45τ . El valor típico para el máximo potencial a través de la membrana es $V_a = 30 mV$. El valor para el potencial asociado a la sinapsis es $V_{syn} = -50 mV$ y la constante de tiempo de la

función α es $\tau = 2 \text{ mseg}$. Esta función α como la denominó Hasegawa[12, 13, 14] también es utilizada por Yoshioka[15] y por Koch[5] para describir el proceso sináptico. La función sináptica α está descrita con detalle por Jack[16].

Los valores numéricos son tomados de Hasegawa[13, 14].

5.3 Salidas de Voltaje

El voltaje en la membrana oscila debido a la entrada de corriente dada por la ecuación 5.4. Como nos interesan los tiempos de disparo más que la forma precisa del potencial de acción podemos expresar la salida como un tren de pulsos

$$U_o = V_a \sum_m \delta(t - t_{om}) \quad (5.10)$$

similar a la ecuación 5.5 para el tren de pulsos de entrada. El intervalo entre disparos entonces sería

$$T_{om} = t_{om+1} - t_{om} \quad (5.11)$$

donde t_{om} es el tiempo en que el potencial de la membrana es cero pasando de negativo a positivo. Nos interesa investigar la variación del intervalo T_{om} con respecto a T_{in} , es decir, nos interesa observar como este tren de impulsos de salida cambia dependiendo del tren de impulsos de entrada.

5.4 Respuesta de una neurona HH.

Se utilizó un método de Runge-Kutta de cuarto orden para la integración numérica de los modelos con un paso $dt = 0.01 \text{ mseg}$. Las simulaciones se corrieron por aproximadamente 200 mseg.

Se estudió el comportamiento de una neurona Hodgkin Huxley para diferentes entradas de corriente cuyos intervalos entre disparos están modulados en el tiempo.

5.4.1 Entrada constante y con modulación periódica.

Entrada constante.

Primero se inyectó en una neurona tipo Hodgkin Huxley una corriente constante ($I_p = 0$). Se observa que cuando $I_s < 6.3 \mu A/cm^2$ la neurona permanece

estable, es decir, no presenta disparos. A partir de este valor de I_s se presentan oscilaciones periódicas en las cuales el intervalo entre disparos T_{om} disminuye a medida que I_s aumenta. En el capítulo 6 se pueden encontrar algunos de los periodos de oscilación para diferentes valores en la corriente de entrada.

Modulación periódica.

Se estudió una entrada con $I_s = 0$, esto implica que la oscilación de la membrana se debe exclusivamente al tren de pulsos de entrada. Para la entrada $I_{ext} = I_p$ con periodo $T_{in} = 10 \text{ mseg}$ podemos ver que el tren de pulsos (figura 5.1) genera la corriente periódica I_p (figura 5.2) cuyo máximo valor es $I_p^{max} = 15.3 \mu A/cm^2$.

Una entrada de corriente periódica en una neurona HH tiene como resultado excitaciones en la neurona con intervalos no constantes. Para ciertos valores de corriente estos intervalos varían periódicamente, un ejemplo es el caso de una entrada de periodo $T_i = 10 \text{ mseg}$ para la cual se observa en la salida (figura 5.3) 3 excitaciones cada 40 mseg. Hay tres valores diferentes de intervalos ($T_{om}=11.25$, 12.36 y 16.39 mseg) (figura 5.5), los tres suman 40 mseg y cada uno de ellos se repite cada tres oscilaciones, lo que implica que en 40 mseg ocurren 4 disparos en la entrada y tres en la salida. Hay una razón de 4:3, es decir, por cuatro pulsos de entrada se observan 3 pulsos de salida. Esta razón de pulsos en la entrada contra pulsos en la salida cambia dependiendo del periodo en la entrada. Para el caso de una entrada con periodo $T_i = 5 \text{ mseg}$ tenemos dos valores de intervalos ($T_{om}=10.94$ y 14.06 mseg), juntos suman 25 mseg. La razón es 5:2, es decir, 5 pulsos de entrada por dos de salida. Este fenómeno no ocurre sino para ciertos periodos en la entrada.

Corriente constante + Modulación Periódica.

Se introdujo una corriente periódica sumada a una corriente constante ($I_s = 25 \mu A/cm^2$ valor utilizado por Hasegawa [13, 14]), el intervalo entre picos (periodo) T_{in} (ecuación 5.6) es $T_{in} = 15 \text{ mseg}$.

El tren de pulsos de la entrada (ecuación 5.5) y la entrada generada se observan en las figuras 5.6 y 5.7. En esta entrada la contribución periódica I_p de la corriente alcanza un valor máximo $I_p^{max} = 14.8 \mu A/cm^2$ cuando $t = t_{in} + \tau$.

En la figura 5.8 la línea punteada muestra oscilaciones del potencial en la membrana debido a una entrada constante ($I_s = 25 \mu A/cm^2$ y $I_p = 0$), podemos observar que la oscilación es diferente para el caso de entrada constante $I_p = 0$ (línea punteada) que para el caso en que I_p es periódico (línea continua). En

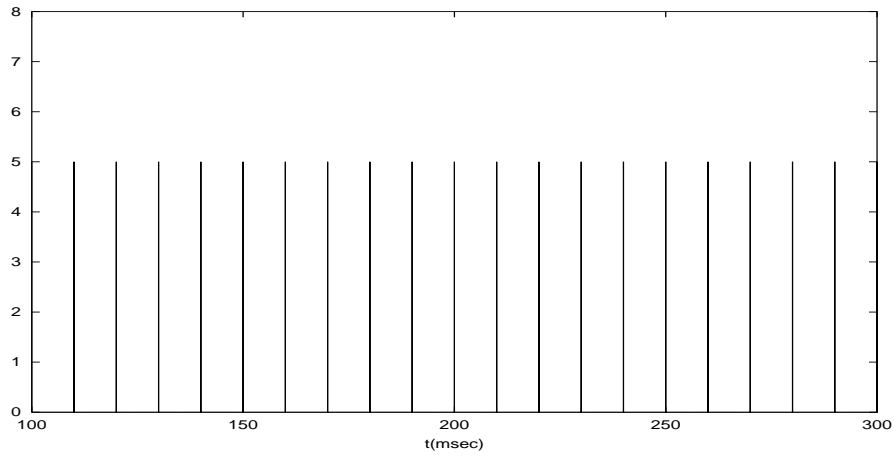


Figura 5.1: Tren de impulsos de entrada U_i para una entrada periódica de periodo 10 mseg.

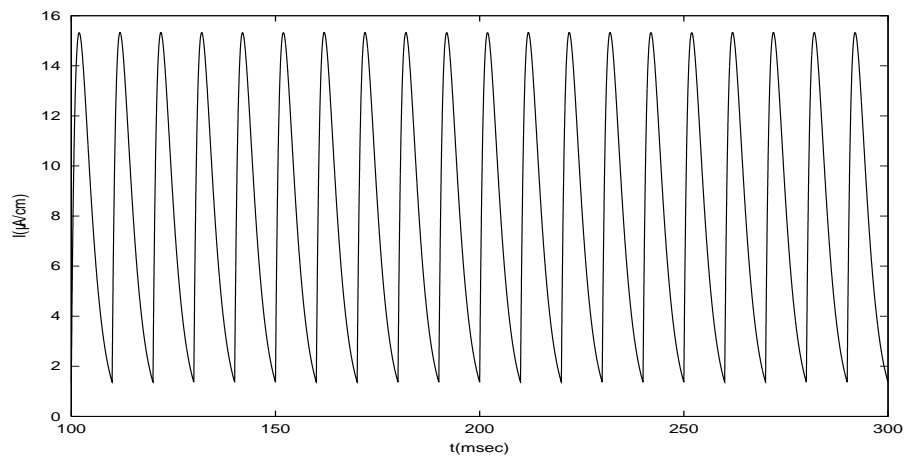


Figura 5.2: Entrada $I_{ext} = I_p$ generada por el tren mostrado en la figura 5.1.

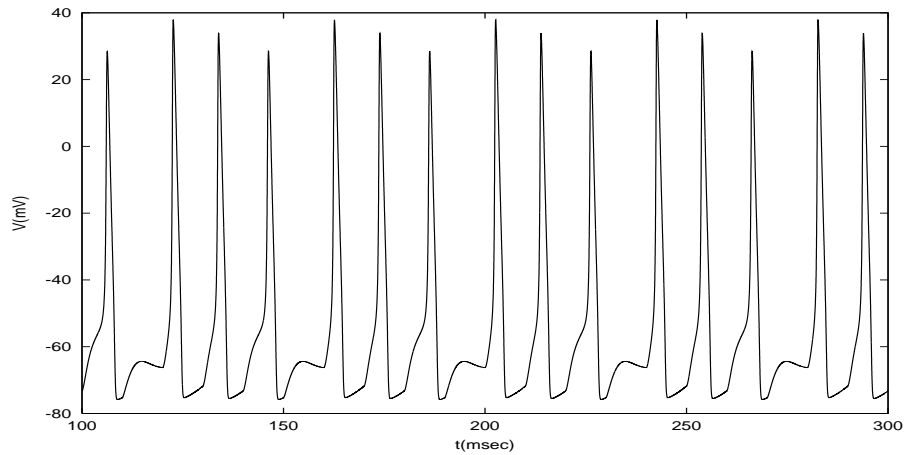


Figura 5.3: Voltaje en la membrana del axón cuando este se estimula con una corriente periódica, con periodo $T_i = 10 \text{ msec}$, como la mostrada en la figura 5.2.

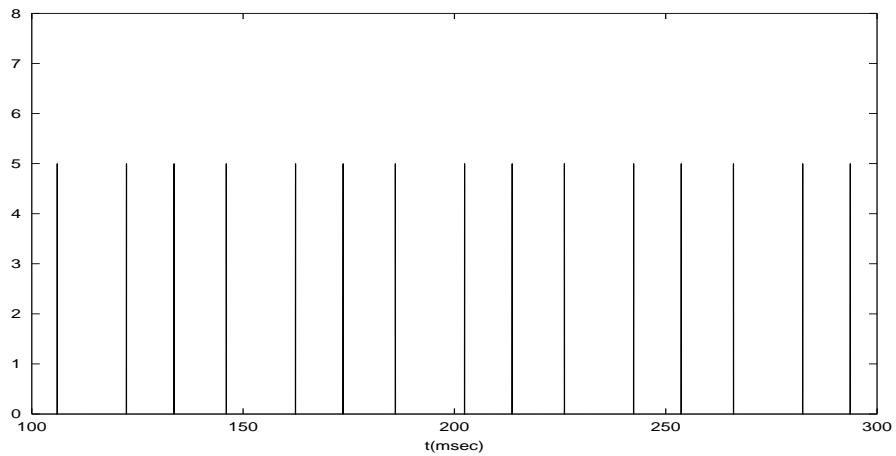
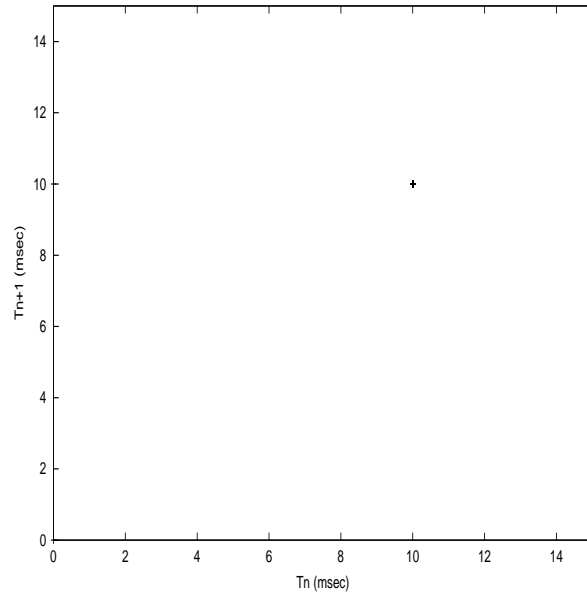
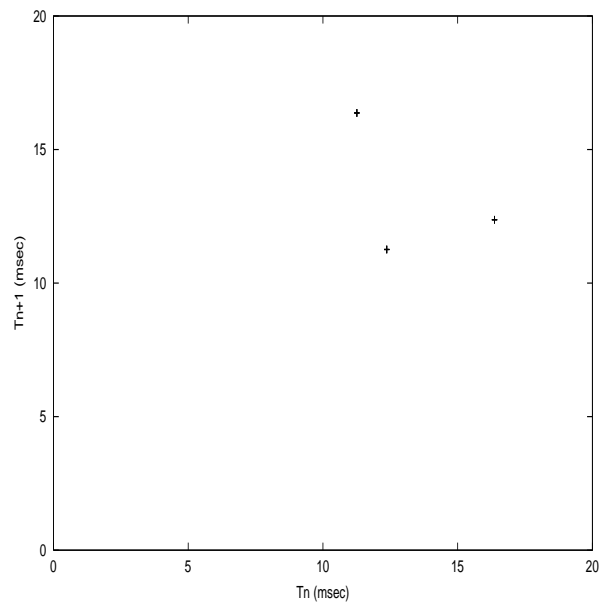


Figura 5.4: Tren de impulsos de salida de voltaje mostrada en la figura 5.3. La entrada es una entrada periódica con periodo $T_i = 10 \text{ msec}$.



(a) Mapa Intervalos Entrada



(b) Mapa Intervalo Salida

Figura 5.5: Diagrama de fase para a) el intervalo entre disparos de la entrada T_{in} y para b) el intervalo entre disparos del voltaje en la membrana T_{om} . Cuando la entrada es una corriente periódica con $I_s = 0$ y periodo 10 msec. Se observan tres valores de intervalos de salida.

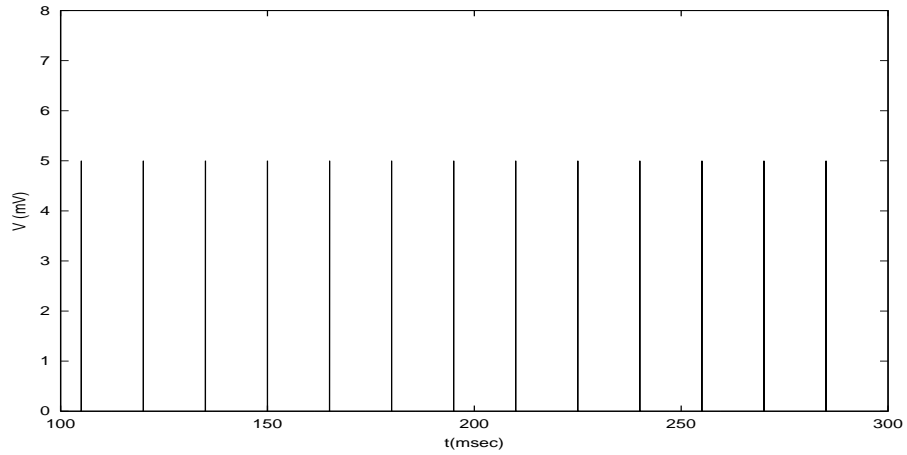


Figura 5.6: Tren de pulsos U_i (ecuación 5.5) con $T_{in} = 15 \text{ msec}$, genera la entrada de corriente I_{ext} mostrada en la figura 5.7.

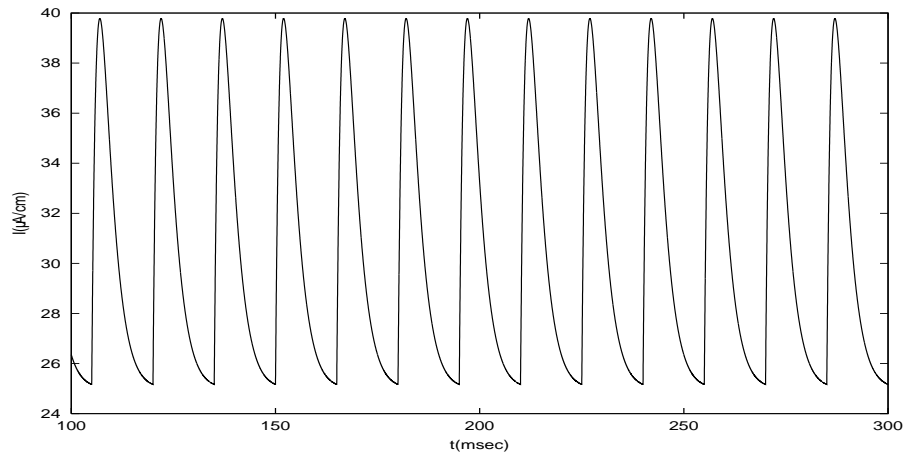


Figura 5.7: Entrada generada por el tren de pulsos mostrado en la gráfica 5.6. Esta entrada es la superposición de una entrada periódica ($T_i = 15 \text{ msec}$) y una corriente constante ($I_s = 25 \mu\text{A}/\text{cm}^2$).

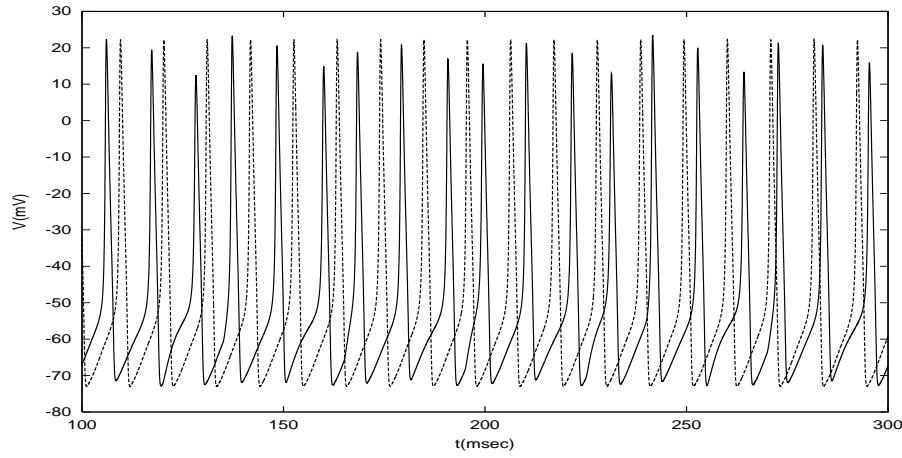


Figura 5.8: Esta figura muestra como varía el voltaje en la membrana de una neurona Hodgkin- Huxley tanto para una entrada constante con $I_s = 25 \mu A/cm^2$ y $I_p = 0$ (línea punteada) como para una entrada periódica de periodo $T_{in} = 15 msec$ y con un nivel DC $I_s = 25 \mu A/cm^2$ (línea continua).

ambos casos podemos observar oscilaciones. Para el caso de entrada constante se observa una salida periódica con periodo $T_{om} = 10.75 msec$, en cambio, cuando se tiene una entrada periódica de periodo $T_{in} = 15 msec$ y una corriente constante DC $I_s = 25 \mu A/cm^2$ los intervalos entre disparos varían entre $T_{om} = 8.36$ y $11.62 msec$ con un promedio $\mu = 10.43 msec$ y desviación estándar $\sigma = 1.12 msec$. En la figura 5.10 se observa que los intervalos de salida T_{om} muestran un comportamiento caótico y se distribuyen en un espacio acotado.

5.4.2 Entrada con modulación senoidal

Se introduce una corriente I_p cuyo intervalo entre picos (ecuación 5.6) es modulado senoidalmente y está dado:

$$T_{in}(t) = d_0 + d_1 \sin(2\pi t/T_p) \quad (5.12)$$

donde T_p es el periodo de la modulación y d_0 y d_1 son parámetros que ajustan la escala de la modulación.

Nos interesa estudiar el efecto de la modulación senoidal, por lo que se escoge un nivel DC $I_s = 0$.

En la figura 5.11 podemos ver que debido a la modulación senoidal los intervalos entre disparos para la entrada de corriente son mucho menores entre

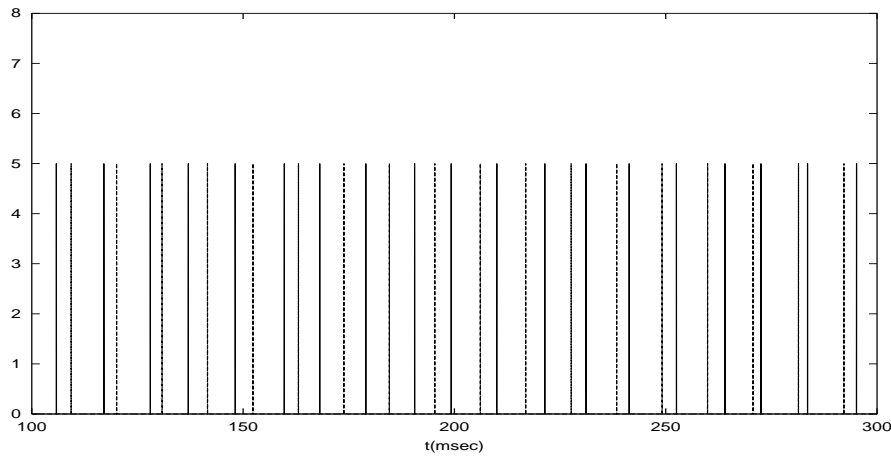


Figura 5.9: Tren de pulsos (ecuación 5.10) correspondiente a las salidas mostradas en la figura 5.8. La entrada consiste en una entrada constante con $I_s = 25 \mu A/cm^2$ y $I_p = 0$ (línea punteada) y una entrada periódica de periodo $T_{in} = 15 \text{ msec}$ con nivel DC de $I_s = 25 \mu A/cm^2$ (línea continua).

$150 \text{ msec} < t < 200 \text{ msec}$ que entre $100 \text{ msec} < t < 150 \text{ msec}$.

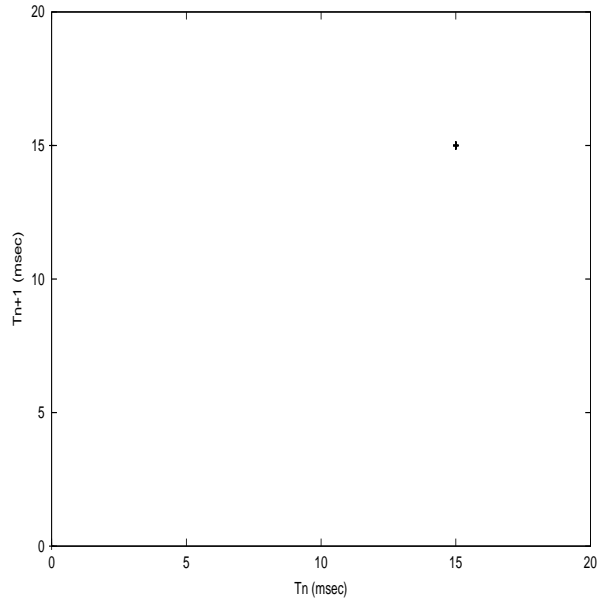
En la figura 5.12 se aprecia el mapa de los intervalos entre picos de la entrada tiene la forma de huevo característica de una modulación senosoidal. La figura 5.14 muestra un comportamiento caótico en el cual los intervalos entre disparos T_{om} están distribuidos entre $11 < T_{om} < 30 \text{ msec}$ casi el mismo rango que los de entrada T_{in} . Los intervalos de la salida tienen promedio $\mu = 17.54$ y desviación estándar $\sigma = 6.94$.

Para el caso de $d_0 = d_1 = 10 \text{ msec}$ con $T_p = 100 \text{ msec}$ se observa que los intervalos entre disparos de la entrada se encuentran distribuidos en $5 < T_{in} < 14.96 \text{ msec}$ y para la salida $11.01 < T_{om} < 19.48 \text{ msec}$ con promedio $\mu = 8.68$ y una desviación estándar de $\sigma = 3.42$.

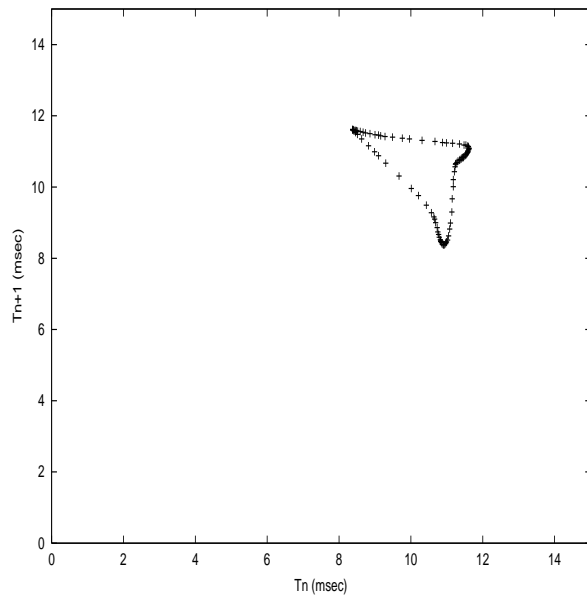
En ninguno de los dos casos; $d_0 = d_1 = 20 \text{ msec}$ y $d_0 = d_1 = 10 \text{ msec}$ hay intervalos entre disparos de salida menores a 11 msec , lo cual se debe al tiempo refractario de la neurona. La neurona tipo Hodgkin Huxley funciona como un filtro pasa altos.

5.4.3 Entrada con Modulación Caótica

Ahora el intervalo entre picos es modulado caóticamente utilizando dos modelos caóticos diferentes.

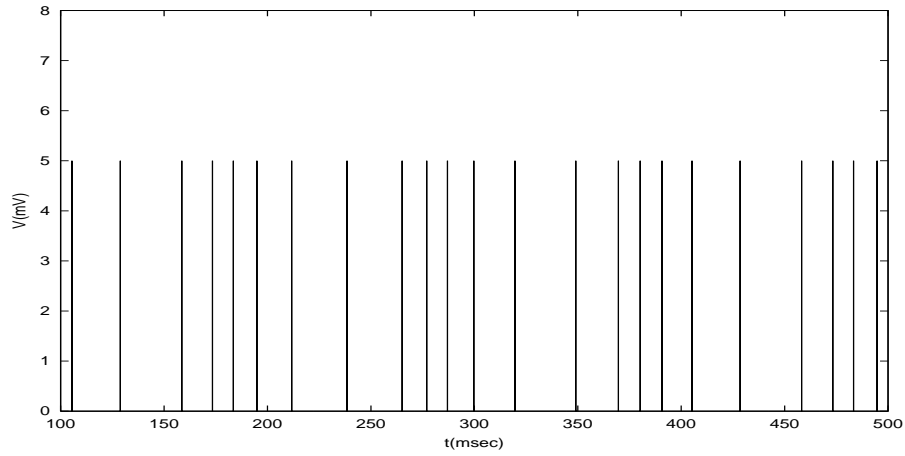
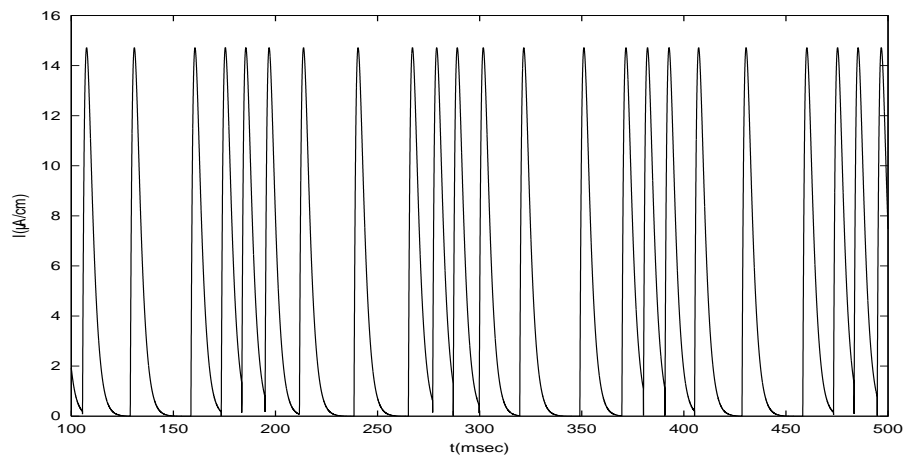


(a) Mapa Intervalos Entrada



(b) Mapa Intervalos Salida

Figura 5.10: Diagrama de Fase para A) Intervalos entre disparos en la entrada, y B) Intervalos entre disparos en la salida. Cuando la neurona se estimula con una entrada de periodo $T_i = 15 \text{ msec}$ y un nivel DC $I_s = 25 \mu A/cm^2$. Acá se observa un comportamiento caótico idéntico al observado por Hasegawa[12].

(a) Tren de pulsos U_i 

(b) Corriente de Entrada

Figura 5.11: Entrada modulada senoidalmente para $d_0 = 2d_1 = 20 \text{ msec}$ y $T_p = 100 \text{ msec}$. La figura A) muestra el tren de impulsos de entrada U_i que genera la corriente $I_{ext} = I_p$ (ecuación 5.4) mostrada en la figura B).

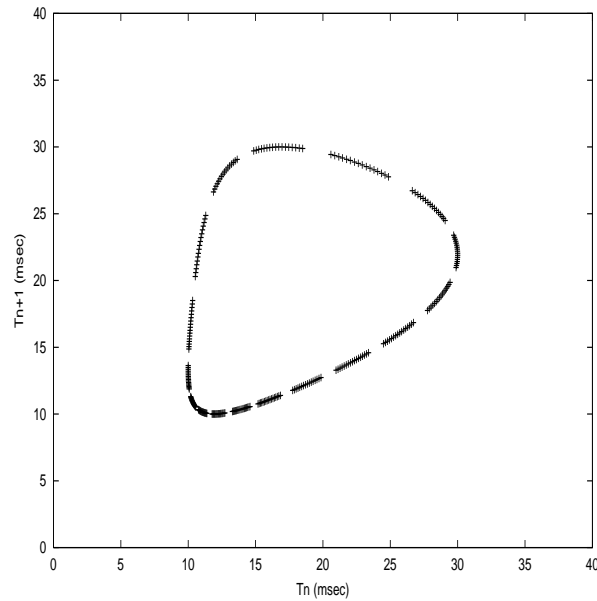


Figura 5.12: La figura muestra el mapa en forma de huevo característico en una modulación senosoidal para el intervalo entre picos de la corriente de entrada I_{ext} con $d_0 = d_1 = 20 \text{ msec}$ y con $T_p = 100 \text{ msec}$.

Modelo de Rossler El modelo de Rossler está dado por el siguiente sistema de tres ecuaciones diferenciales acopladas.

$$\frac{dx}{dt} = y - z \quad (5.13)$$

$$\frac{dy}{dt} = x + ay \quad (5.14)$$

$$\frac{dz}{dt} = bx - cz + xz \quad (5.15)$$

donde $a=0.36$, $b=0.4$ y $c=4.5$. Los valores para a , b y c fueron tomados tal que el sistema se encuentre en una región favorable, es decir, que exista un atractor y las trayectorias que tome el sistema no se alejen demasiado del atractor y así no tener intervalos demasiado grandes.

El intervalo entre picos tiene que ser positivo, por lo que se moduló este intervalo con la variable $x(t)$ con un ajuste de escala. El tiempo entre picos de la ecuación 5.6 está dado por:

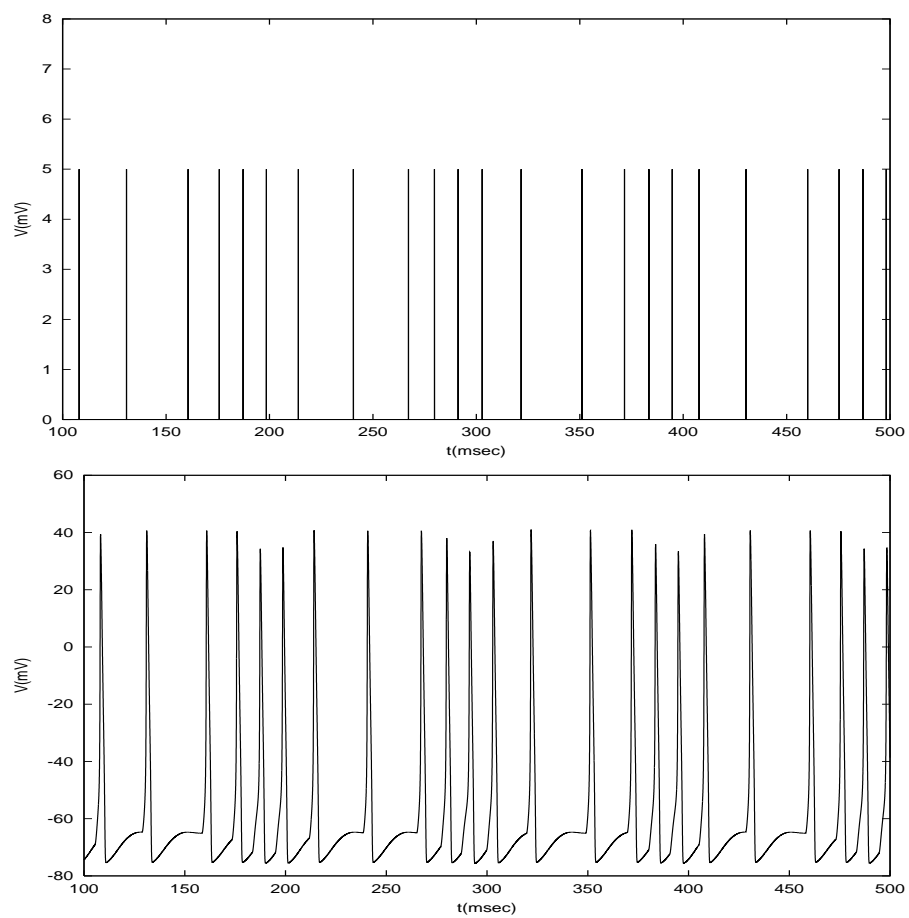


Figura 5.13: Respuesta de una neurona Hodgkin y Huxley a la entrada mostrada en la figura 5.11. Primero se observa U_o que es generado a partir de los máximos en los picos del voltaje.

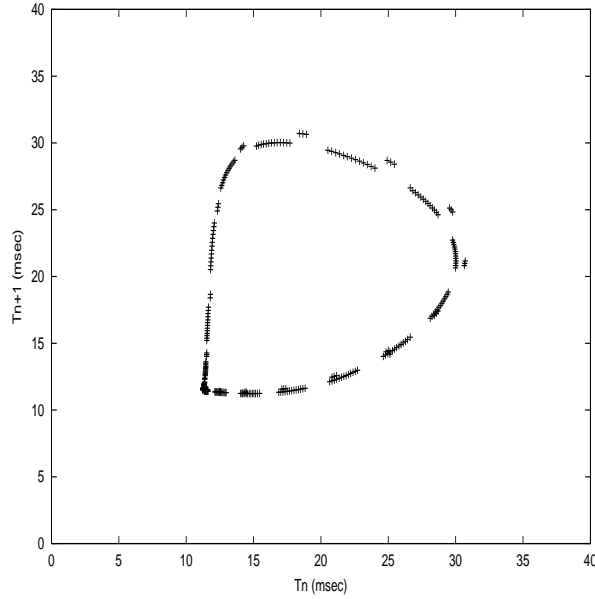


Figura 5.14: Mapa para el intervalo de salida generado por una corriente modulada senoidalmente con $d_0 = 2d_1 = 20 \text{ msec}$.

$$T_{in}(t_{in}) = d_0 + (d_1/10)x(pt_{in}) \quad (5.16)$$

Se hicieron dos ensayos con diferentes parámetros: El primero fue utilizando $d_0 = d_1 = 10 \text{ msec}$ y $p = 1/10$, el segundo fue con $d_0 = d_1 = 20 \text{ msec}$ y $p = 1/20$.

Modelo de Lorenz También se consideró una modulación utilizando el atractor de Lorenz dado por el siguiente sistema de ecuaciones diferenciales:

$$\frac{dx}{dt} = d(y - x) \quad (5.17)$$

$$\frac{dy}{dt} = ex - y - xz \quad (5.18)$$

$$\frac{dz}{dt} = -fz + xy \quad (5.19)$$

con $d=10$, $e=28$ y $f=8/3$. Valores escogidos con el mismo criterio que en el modelo de Rossler. Para el intervalo entre picos utilicé la variable z . T_{in} en la ecuación 5.6 está dado por:

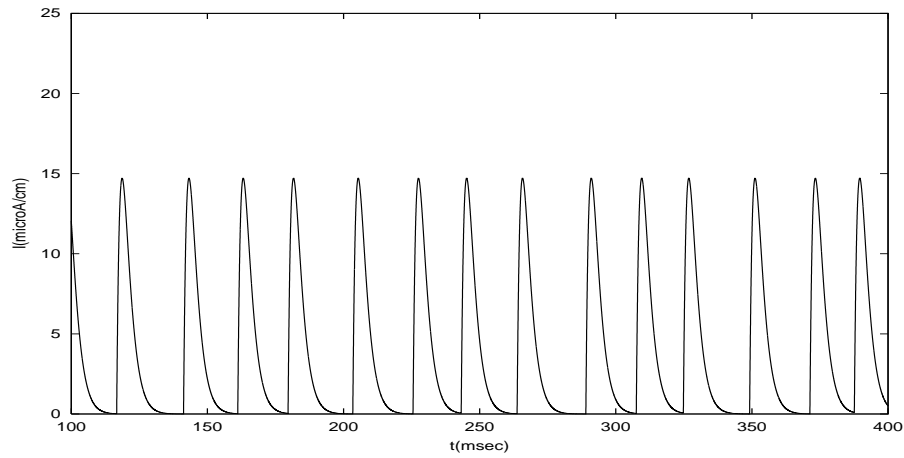


Figura 5.15: Corriente generada a partir del tren de pulsos (figura 5.16) cuya modulación es generada por el atractor de Lorenz.

$$T_{in}(t_{in}) = d_0 + (d_1/25)(z(pt_{in}) - 25) \quad (5.20)$$

donde $d_0 = d_1 = 20$ y $p=1/100$.

Variando los parámetros del modelo de Lorenz se pueden tener una gran cantidad de valores para los intervalos entre disparos. Estos parámetros se escogieron para que los intervalos entre disparos estuvieran distribuidos entre 5 y 30 msec. Los intervalos entre los disparos en el potencial de la membrana muestran un comportamiento caótico por lo que se distribuyen en un espacio acotado. Por cuestiones de tiempo, la simulación se corrió por 100 seg. con un $dt = 0.01$ msec, puede ser interesante correr la simulación por más tiempo y así buscar más estructura en los intervalos entre disparos de la salida. También cambiando los parámetros del modelo se puede encontrar algo interesante.

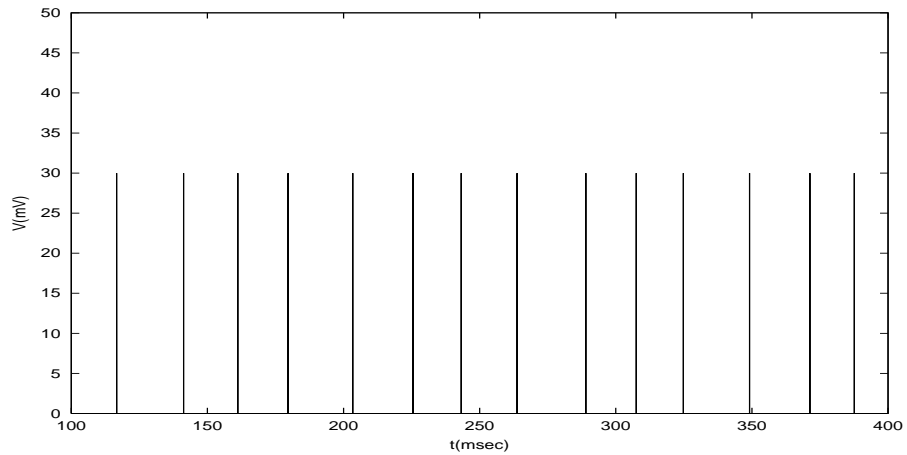


Figura 5.16: Los Intervalos entre disparos son modulados con el modelo de Lorenz para los parámetros descritos anteriormente.

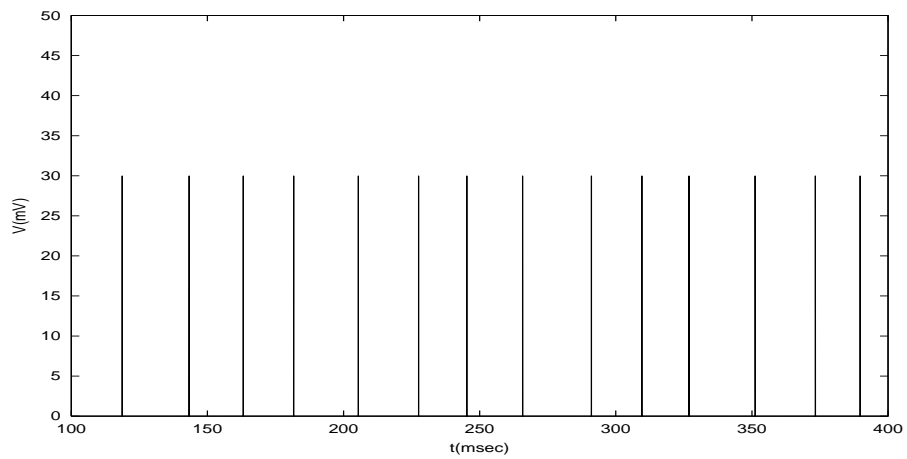


Figura 5.17: Tren de pulsos generado por una Neurona Hodgkin Huxley cuando los intervalos entre disparos de la entrada son modulados caóticamente con el atractor de Lorenz.

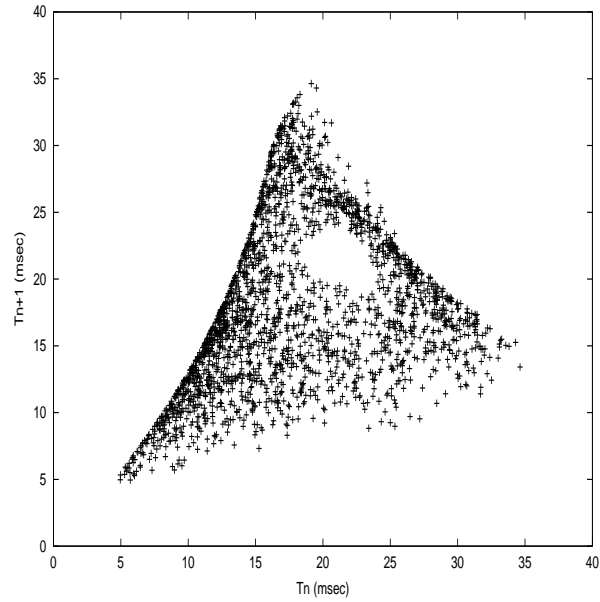
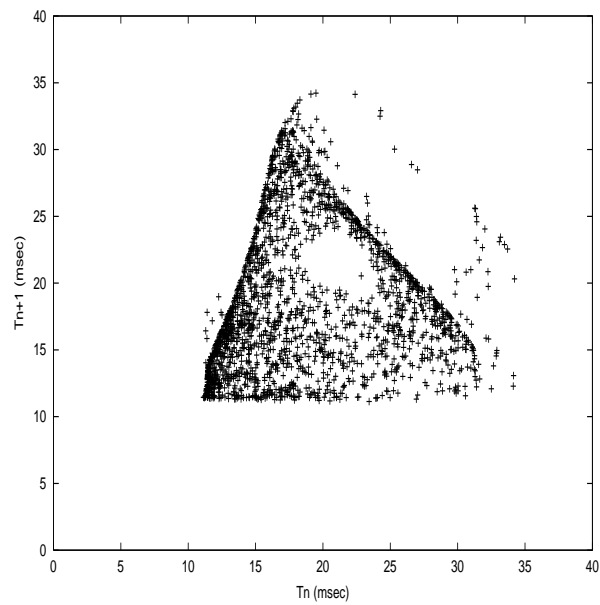
(a) Intervalos entre disparos T_{in} (b) Intervalos para la salida T_{om}

Figura 5.18: Diagrama de fase a) para el intervalo entre disparos de la entrada T_{in} y b) para el intervalo entre disparos en el voltaje a través de la membrana T_{om} . Los intervalos entre disparos en la entrada son modulados con el atractor de Lorenz.

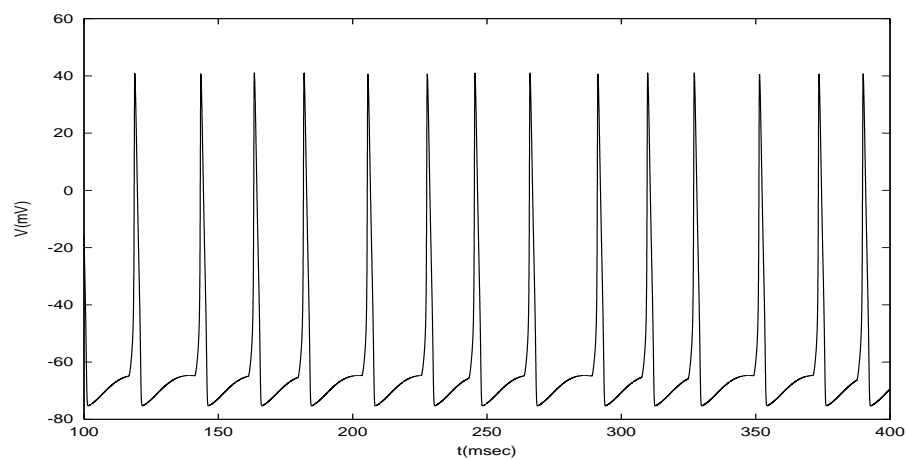


Figura 5.19: Respuesta de una neurona Hodgkin Huxley a una entrada modulada con el atractor de Lorenz.

Capítulo 6

Ajuste de una neurona de disparo (IF).

En este capítulo se intenta reproducir la salida de una neurona tipo Hodgkin Huxley ajustando los parámetros de una neurona de disparo. Como ya se ha mencionado anteriormente la forma de los picos de voltaje (potencial de acción) en la salida no es relevante, la información se encuentra codificada en la frecuencia y distribución de los disparos de la neurona.

Utilicé un tipo de neurona de disparo sencillo; un condensador C que almacena la carga que se introduce a la neurona (I_{ext}), cuando el voltaje en el condensador llega a cierto umbral V_{th} , se genera un impulso de voltaje V_i , y un instante después se libera la carga de forma exponencial con una constante de tiempo grande, i.e. se descarga lentamente.

La ecuación diferencial que rige la carga del condensador es:

$$\frac{dV}{dt} + \frac{(V - V_r)}{\tau} = \frac{I_{ext}}{C} \quad (6.1)$$

donde V_r es el voltaje en reposo de la neurona y τ es el tiempo de carga del condensador. El tiempo refractario de la neurona es el tiempo en que se demora en llegar desde este pico de voltaje hasta el voltaje de reposo V_r , en este intervalo el voltaje se rige por la siguiente ecuación diferencial.

$$\frac{dV}{dt} + \frac{1}{\tau_r}(V - V_r) = 0 \quad (6.2)$$

I_{ext} (μA)	Periodo (mseg)	Tiempo de carga τ (mseg)
10	14.66	16.406
15	12.73	6.782
20	11.57	4.490
25	10.76	3.420
30	10.14	2.774

Tabla 6.1: Tiempo de carga τ para ajustar una neurona de disparo con $C = 4 \mu F$ con el fin de reproducir la salida de una neurona tipo Hodgkin Huxley a una entrada de corriente constante I_{ext} .

I_{ext} (μA)	Periodo (mseg)	C (μF) para $\tau = 15 \text{ mseg}$	C (μF) para $\tau = 20 \text{ mseg}$
10	14.66	3.940	4.194
15	12.73	5.302	5.566
20	11.57	6.552	6.865
25	10.76	7.724	8.008
30	10.14	8.790	9.170

Tabla 6.2: Capacitancia C para ajustar una neurona de disparo con $\tau = 15 \text{ mseg}$ y $\tau = 20 \text{ mseg}$ con el fin de reproducir la salida de una neurona tipo Hodgkin Huxley a una entrada de corriente constante I_{ext} .

Parámetros de la neurona de disparo.

El voltaje umbral se escogió como $V_{th} = -55 \text{ mV}$, valor observado en las gráficas obtenidas para el modelo de Hodgkin-Huxley. El voltaje de reposo es el mismo utilizado por Hasegawa[12, 13, 7, 14] $V_r = -75 \text{ mV}$. El tiempo de carga y la capacitancia del condensador se consideran variables. El tiempo de descarga para la parte refractiva se tomo como $\tau_r = 100/\tau$.

Ajuste para una entrada de corriente constante.

Como se observa en el capítulo anterior una entrada de corriente I_{ext} constante genera oscilaciones periódicas en el potencial de la membrana. El periodo de oscilación de una neurona Hodgkin y Huxley depende de la magnitud de esta entrada. Es posible ajustar la neurona de disparo para reproducir estas oscilaciones el único problema es que se tiene que ajustar para cada entrada. Los valores de los parámetros encontrados se encuentran en las tablas 6.1 y 6.2.

Es posible reproducir la respuesta de una neurona Hodgkin Huxley a una entrada I_{ext} constante. Los parámetros que ajustan la neurona IF para una entrada constante especifica se encuentran en las tablas 6.1 y 6.2. En las figuras

a continuación se puede observar como se comporta una neurona cuando uno de los dos parámetros está fijo. Como era de esperarse son relaciones exponenciales.

Para una capacitancia constante la gráfica de τ contra el periodo de oscilación se desplaza cuando se cambia la magnitud de la corriente de entrada I_{ext} , si se aumenta esta corriente de entrada se obtiene una curva más cercana a los ejes de la gráfica. Cuando se tiene un tiempo de carga fijo τ , la corriente de entrada I_{ext} reduce la pendiente de la gráfica C contra Periodo (figura 6.2), es decir el periodo es más sensible a los cambios en la capacitancia del modelo.

Entrada periódica.

Se estudió una entrada periódica con periodo $T = 15 \text{ mseg}$ ver figura 6.3.

Se observó que una neurona de disparo puede reproducir una salida de una neurona tipo Hodgkin Huxley, sólo se tienen que ajustar los parámetros de la neurona de disparo. El problema de esto es que la neurona de disparo es capaz de reproducir la salida para una entrada específica, al cambiar de entrada tenemos que volver a ajustar la neurona.

Se hizo un intento de hacer un análisis de correlación entre las separaciones entre picos de la respuesta obtenida por el modelo de Hodgkin y Huxley y por el modelo de disparo, la idea inicial era hacer un programa que variara el tiempo de carga de la neurona de disparo y para cada valor de τ calcular un coeficiente de correlación. No se encontró correlación alguna.

Se observó que los intervalos entre disparos en la salida cuando la entrada es periódica son mucho más complicados de reproducir que cuando la entrada es constante, se puede ajustar para que se parezca bastante pero es un proceso lento y de muchas iteraciones.

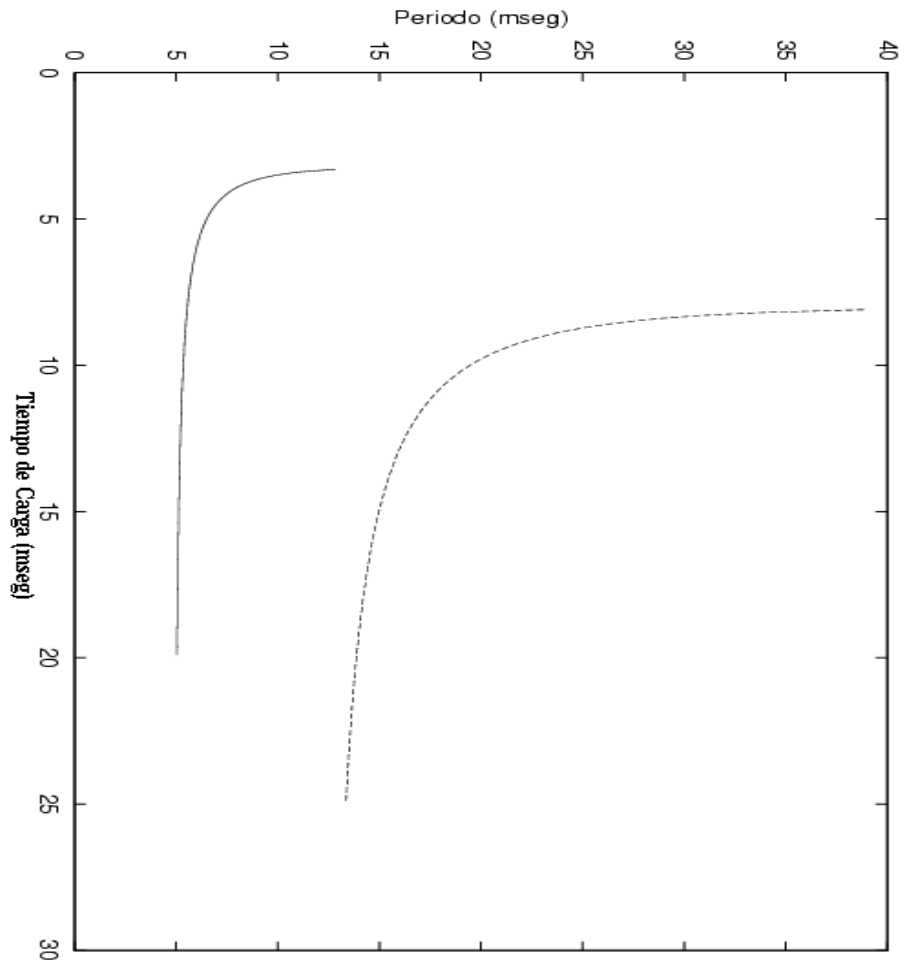


Figura 6.1: Variación del periodo de oscilación con relación al tiempo de carga τ , para una neurona de disparo con $C = 4 \mu F$ cuando se tiene una entrada de corriente $I_{ext} = 10 \mu A$ (línea punteada) y $I_{ext} = 25 \mu A$ (línea continua).

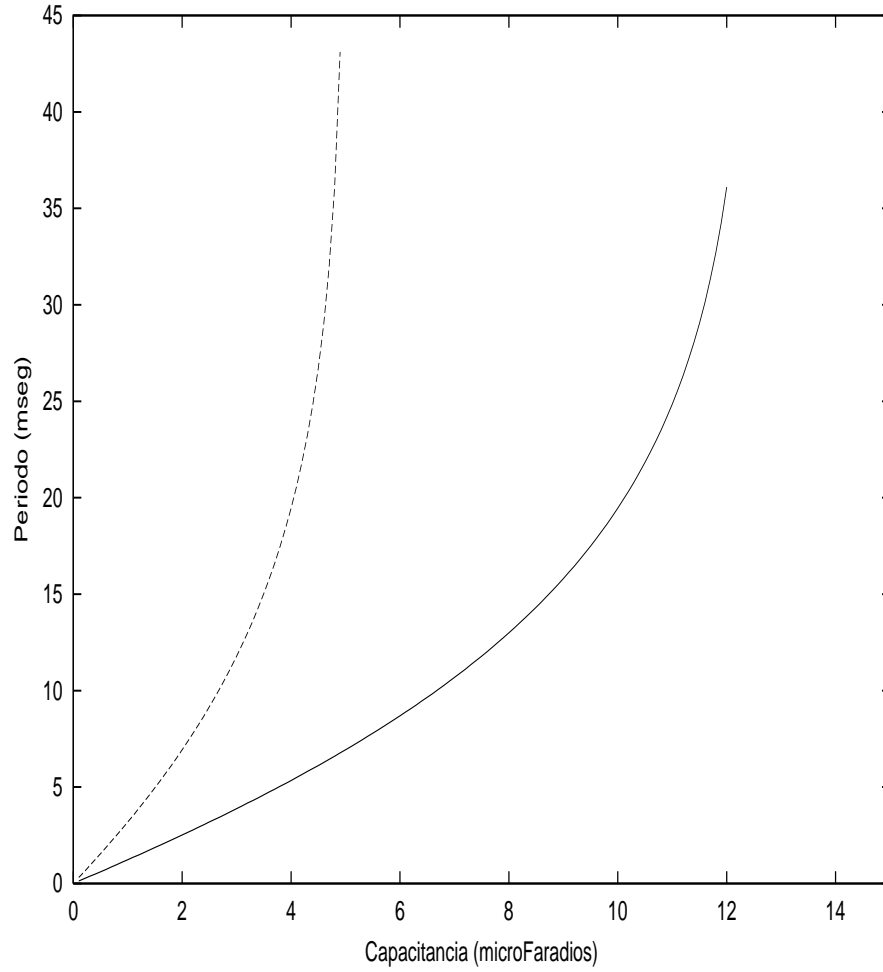
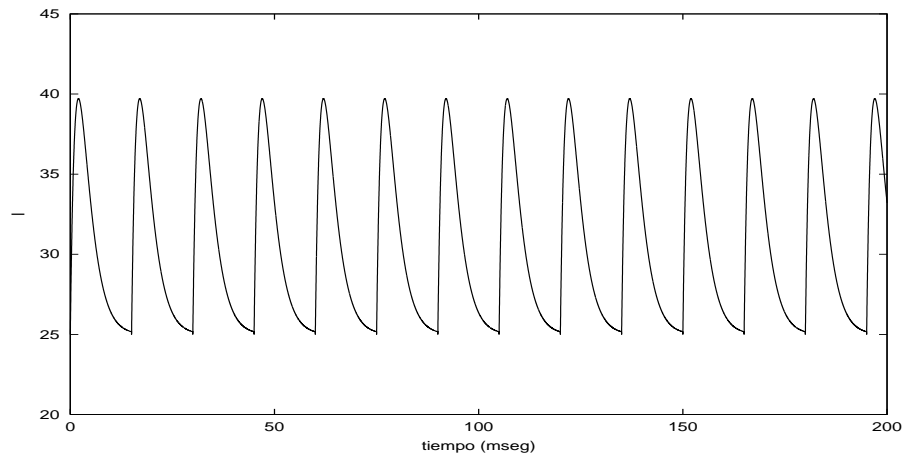
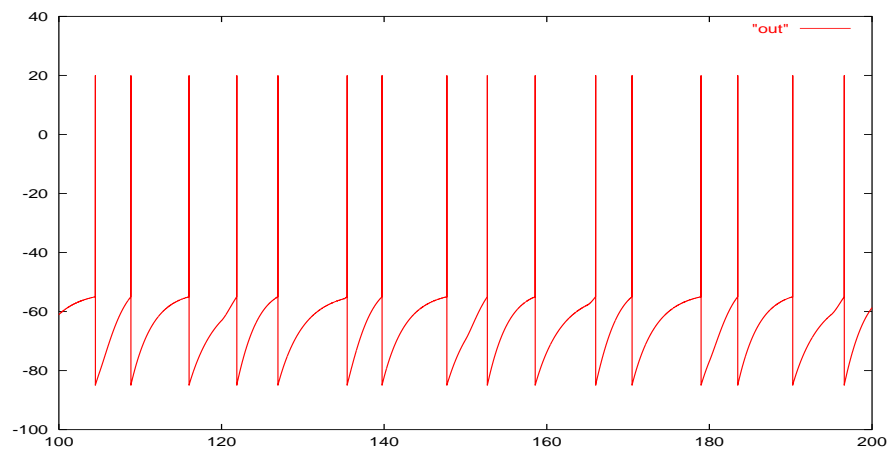


Figura 6.2: Variación del periodo de oscilación con relación a la capacitancia C , para una neurona de disparo con $\tau = 10 \text{ mseg}$ cuando se tiene una entrada de corriente $I_{ext} = 10 \mu A$ (línea punteada) y para $I_{ext} = 25 \mu A$ (línea continua).

Figura 6.3: Entrada Periódica $T = 15 \text{ mseg}$.Figura 6.4: Salida de una neurona de disparo con $C = 4 \mu F$ y $\tau = 3.5 \text{ mseg}$ para una entrada periódica $T = 15 \text{ mseg}$ (ver figura 6.3).

Capítulo 7

Acople entre neuronas

En este capítulo se van a acoplar neuronas tipo Hodgkin Huxley por medio de acoples sinápticos. En la primera parte se acoplan dos neuronas, tal que el tren de impulsos de salida de la neurona uno genera una entrada sináptica para la neurona dos y viceversa. Existe un tiempo de retraso para que la información pase de una neurona a otra, este tiempo es lo que demora el impulso nervioso en atravesar el axón y las dendritas.

Para la segunda parte de el capítulo se construye una red neuronal de N neuronas acopladas sinápticamente con tiempo de retraso. La red es del tipo Hopfield[11], es decir, la salida de una neurona genera una entrada en todas las otras neuronas menos en ella misma, las conexiones entre las neuronas se calculan con una regla Hebbiana (que se discutirá a continuación). Esta red funciona como una memoria asociativa, como se explica más adelante.

7.1 Modelo

El comportamiento de una neurona j está dado por el modelo de Hodgkin Huxley[2], es el mismo modelo descrito anteriormente, adicionalmente se incluye una corriente de interacción entre neuronas. La dinámica del potencial a través de la membrana V_j para una neurona j está dado por:

$$C \frac{dV_j(t)}{dt} = -I_j^{ion}(V_j, m_j, h_j, n_j) + I_j^{ext} + I_j^{int} \quad (7.1)$$

donde $C = 1 \mu F/cm^2$ es la capacitancia de la neurona. El primer término de la ecuación 7.1 describe la corriente debido a los iones de sodio, potasio y la

corriente de fuga:

$$I_j^{ion}(V_j, m_j, h_j, n_j) = g_{Na} m_j^3 h_j (V_j - V_{Na}) + g_k n_j^4 (V_j - V_k) + g_f (V_j - V_f) \quad (7.2)$$

Todas las neuronas tienen los mismos parámetros descritos en la sección 5.1. Las variables m_j , h_j y n_j son las mismas que están descritas por las ecuaciones dadas en la sección 3.1. El segundo término de la ecuación 7.1 denota la corriente externa aplicada a la neurona j y está dada por:

$$I_j^{ext} = g_{sin}(V_a - V_c) \sum_n \alpha(t - t_{in}), \quad (7.3)$$

la cual es inducida por el tren de pulsos pre-sináptico ecuación 5.5.

$$U_i(t) = V_a \sum_n \delta(t - t_{in}) \quad (7.4)$$

En las ecuaciones 7.3 y 7.4 t_{in} se refiere al n -ésimo tiempo de disparo del tren de pulsos de entrada U_i , g_{sin} es la conductancia asociada al proceso sináptico, V_c es el potencial de equilibrio, V_a es el valor típico para el máximo de potencial en la membrana y τ_s la constante de tiempo relevante a la conducción sináptica (ver valores en la sección 5.1). La función α es la misma función descrita por la ecuación 5.9 cambiando τ por τ_s .

Cuando el potencial en la membrana $V_j(t)$ de la neurona j oscila, se crea un tren de pulsos de salida similar al de la ecuación 5.10 dado por:

$$U_{oj}(t) = V_a \sum_m \delta(t - t_{ojm}) \quad (7.5)$$

el tiempo de disparo t_{ojm} es el tiempo en que una neurona j dispara por m -ésima vez, es decir su potencial de membrana pasa por $V = 0.0 mV$ de abajo hacia arriba.

El tercer término de la ecuación 7.1 describe la corriente de interacción entre neuronas

$$I_j^{int} = G \left(\sum_{k \neq j} \sum_m (g_{jk}^{exc} - g_{jk}^{inh}) (V_a - V_c) \alpha(t - \tau_{jk} - t_{okm}) \right) \quad (7.6)$$

donde

$$g_{jk}^{exc} = g_{exc} W_{jk} \quad (7.7)$$

$$W_{jk} = \theta \left(\sum_{\mu=1}^P \xi_j^{(\mu)} \xi_k^{(\mu)} \right) \quad (7.8)$$

$$G(x) = x\theta(x) \quad (7.9)$$

En las ecuaciones 7.6 - 7.9 no se permite la interacción mutua ($j = k$), g_{jk}^{exc} es la conductancia de la sináptica excitatoria y g_{jk}^{inh} es la conductancia de la parte inhibitoria. τ_{jk} es el tiempo que se demora una corriente en pasar a través del axón y la dendrita para llegar de una neurona j a una neurona k . W_{jk} es la matriz de pesos para las conexiones entre las neuronas, dada por la regla de Willshaw (ec. 7.8) dice que tan fuerte es la conexión entre una neurona i y una neurona j . La forma de la ecuación 7.7 se discute en la sección 7.3.

Los acoples son excitativos, pero se incluye una parte inhibitoria para reducir ruido. Este término fue incluido por Hasegawa en su modelo[14]. La corriente de interacción siempre es positiva debido a la naturaleza de la función $G(x)$.

7.2 Acople de dos neuronas Hodgkin Huxley.

Para el caso de dos neuronas acopladas la ecuación 7.6 se reduce a

$$I_j^{int} = \sum_{k \neq j} \sum_m C_{jk} \alpha(t - \tau_{jk} - t_{okm}) \quad (7.10)$$

donde $C_{jk} = |C_{12}| = |C_{21}|$ representa el acople entre las dos neuronas.

$$C_{jk} = g_s (V_a - V_s) \quad (7.11)$$

donde g_s y V_s son la conductancia y el potencial de equilibrio relativo al proceso sináptico. $\tau_{12} = \tau_{21} = \tau_d$ es el tiempo de retraso.

En el modelo se utilizan tres parámetros T_i y τ_d que se van a tratar como parámetros libres ya que los tiempos de retraso y los intervalos entre picos varían mucho entre los sistemas biológicos.

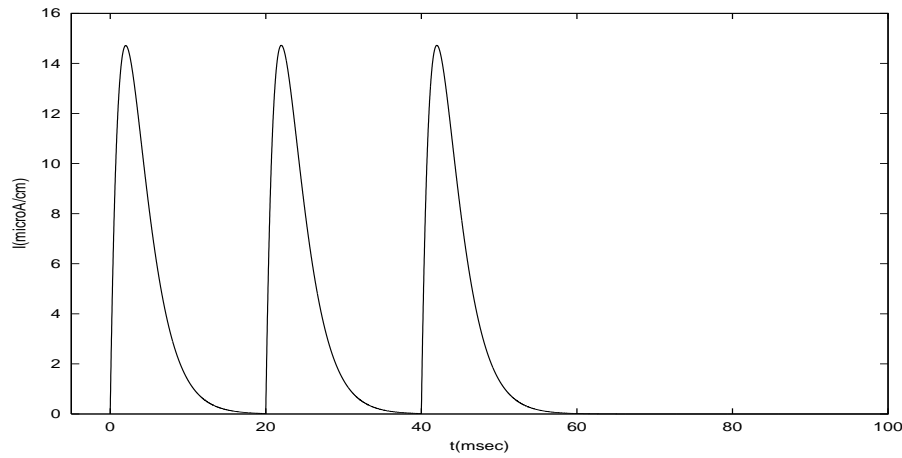


Figura 7.1: Entrada de corriente con $M=3$ pulsos e intervalos entre ellos de $T_i = 20$ mseg.

7.2.1 Comportamiento transiente y estable

Se estudió la respuesta transiente de dos neuronas acopladas cuya entrada consiste de M pulsos de corriente separados por un tiempo T_i , la entrada se aplicó a una de las dos neuronas.

Se hicieron pruebas variando M de 2-5 y con intervalos entre picos $T_i = 5, 10, 20$ mseg. Las gráficas fueron hechas con $M=3$, $T_i = 20$ mseg, $\tau_d = 10$ mseg.

Para una neurona solitaria la respuesta a una entrada de M pulsos depende del intervalo entre los pulsos, para un intervalo $T_i = 20$ mseg la salida muestra el mismo número de picos e igualmente espaciados. Cuando $T_i = 5$ mseg o $T_i = 10$ mseg el intervalo entre picos en la salida es generalmente mayor al de entrada y el número de picos no es necesariamente igual.

Dos neuronas acopladas funcionan como un oscilador, cuando la neurona 1 recibe una entrada genera un pulso de salida, que se convierte en una entrada sináptica para la neurona 2, τ_d milisegundos después, esta neurona 2 se excita y genera otro pulso que vuelve a la neurona 1.

En la figuras 7.2 y 7.4 se muestran el tren de pulsos de entrada U_i (ecuación 5.5) y el tren de pulsos de salida U_o cuando tenemos dos neuronas acopladas. Aunque tenemos una entrada transiente en una de las dos neuronas, se continúa observando oscilación en ambas neuronas, inclusive después de haber terminado de entrar corriente en la neurona.

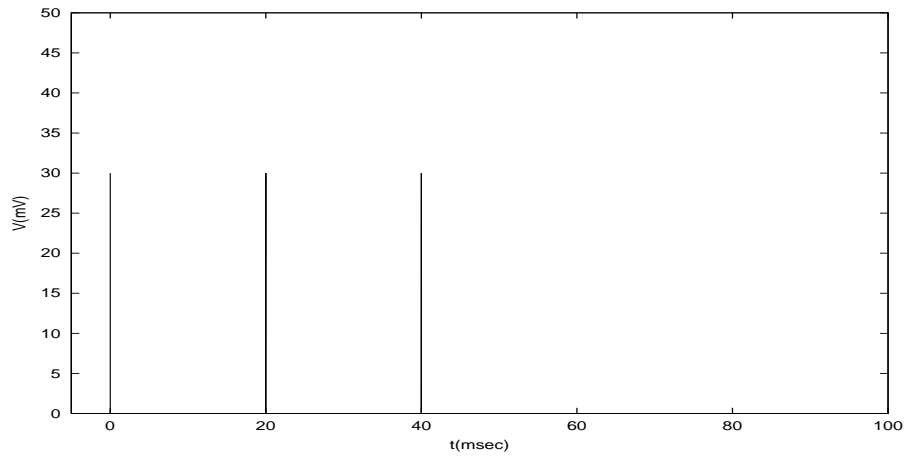


Figura 7.2: Tren de pulsos U_i que genera la entrada de la figura 7.1..

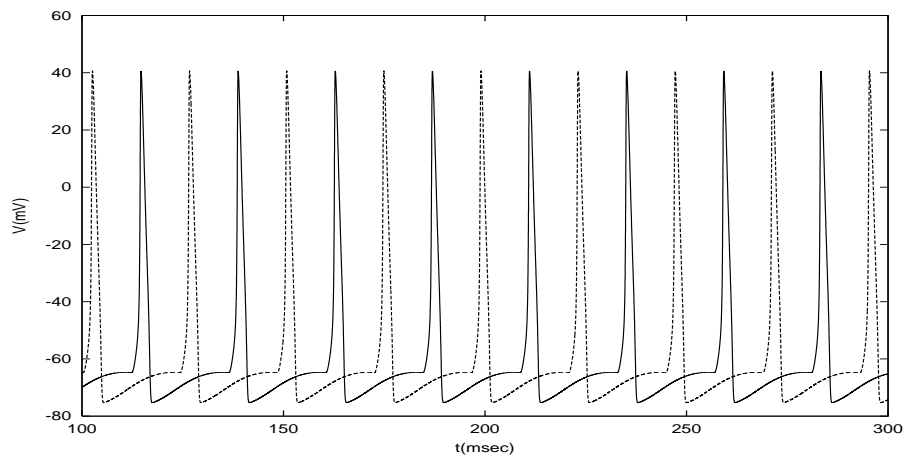


Figura 7.3: Voltaje en la membrana V_j para la entrada mostrada en la figura 7.1, la neurona 2 (línea punteada) muestra un desfase de $\tau = \tau_d + \tau_r$ con respecto a la neurona 1 (línea continua), τ_d es el tiempo de retraso y τ_r el tiempo de reacción de la neurona.

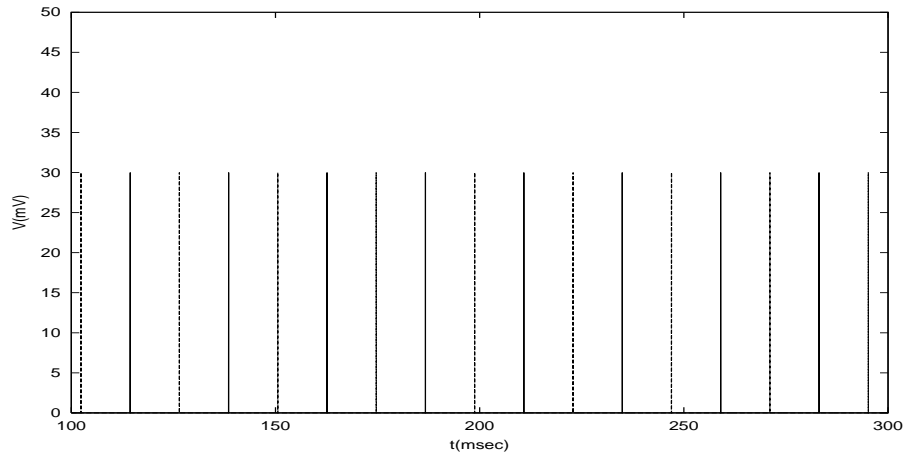


Figura 7.4: Tren de pulsos de salida U_{oj} en la membrana para la entrada mostrada en la figura 7.1, y generado a partir de la salida (figura 7.3). U_{o1} (línea continua) y U_{o2} (línea punteada).

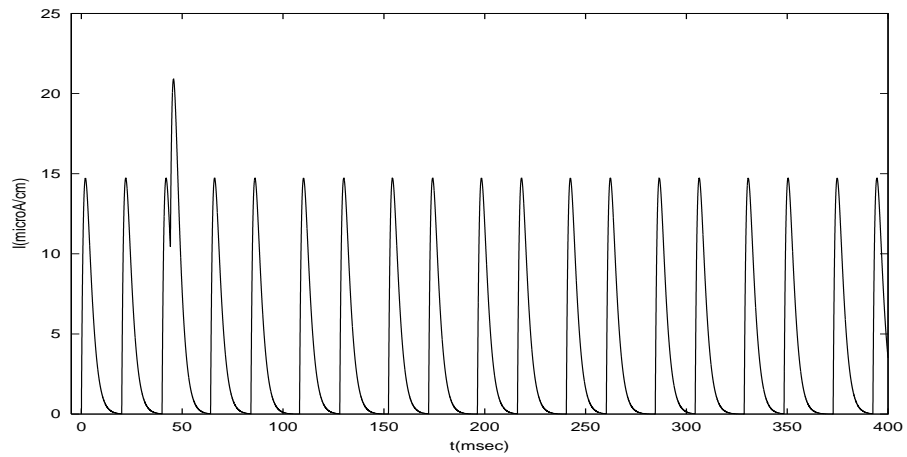


Figura 7.5: Corriente de acople I_1^{int} para la entrada mostrada en la figura 7.1, esta es la corriente que recibe la neurona 1 debido al tren de pulsos de entrada y a la excitación de la neurona 2.

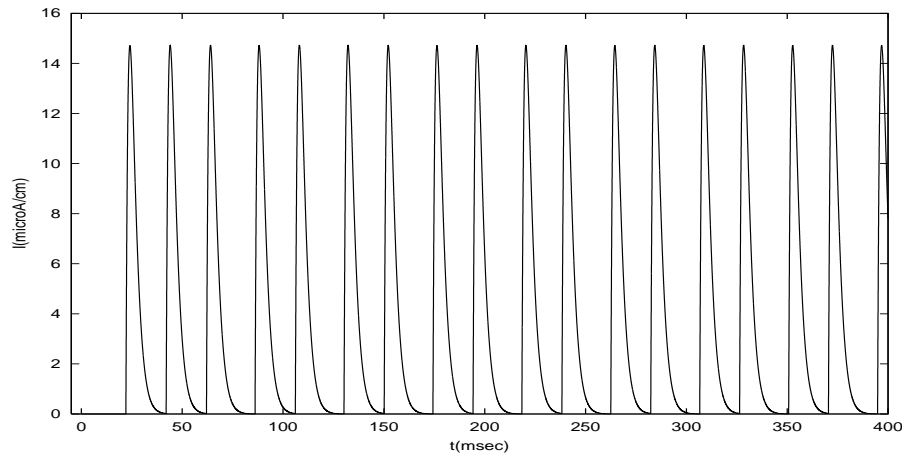


Figura 7.6: Corriente de acople I_2^{int} para la entrada mostrada en la figura 7.1, esta es la corriente que recibe la neurona 2 debido a la excitación de la neurona 1.

Las figuras 7.5 y 7.6 muestran la corriente post-sináptica ($I_j = I_j^{ext} + I_j^{int}$) que entra en las neuronas (ecuaciones 7.6 y 7.3), para la segunda neurona se observa excitación después de $\tau = \tau_d + \tau_r = 12.45 \text{ msec}$, el tiempo de retraso más el tiempo de reacción de la neurona. Para la neurona 1 se observa una superposición de la entrada externa y la corriente procedente de la neurona 2, después de 60 msec la entrada ya está apagada y la corriente que se observa se debe plenamente a la excitación de la segunda neurona.

La figura 7.3 muestra el potencial a través de ambas membranas V , el primer pulso insertado en la neurona 1 hace que esta reaccione generando otro pulso $\tau_r \sim 2 \text{ msec}$. Este pulso generado por la neurona 1 se propaga a lo largo del axón y llega $\tau_d = 10 \text{ msec}$, después a la neurona 2, la cual a su vez se demora $\tau_r \sim 2 \text{ msec}$ en disparar, otra vez el impulso tarda $\tau_d = 10 \text{ msec}$ en volver a la neurona 1. Un pulso de entrada causa una oscilación en el potencial de la membrana con un periodo $t_o = 24.10 \text{ msec}$.

Cuando se estudió una neurona Hodgkin-Huxley (capítulo 3.1) con una entrada de intervalos de $T_i = 20 \text{ msec}$, la salida tenía intervalos entre disparos de $T_o = 20 \text{ msec}$ y el número de pulsos saliendo era el mismo número de pulsos entrando. Cuando $T_i = 5 \text{ msec}$ y $T_i = 10 \text{ msec}$ los intervalos entre picos de salida eran generalmente más grandes que los de entrada y el número de pulsos saliendo no es igual al número de pulsos que entran.

Para el caso de dos neuronas acopladas con un tiempo de retraso $\tau_d =$

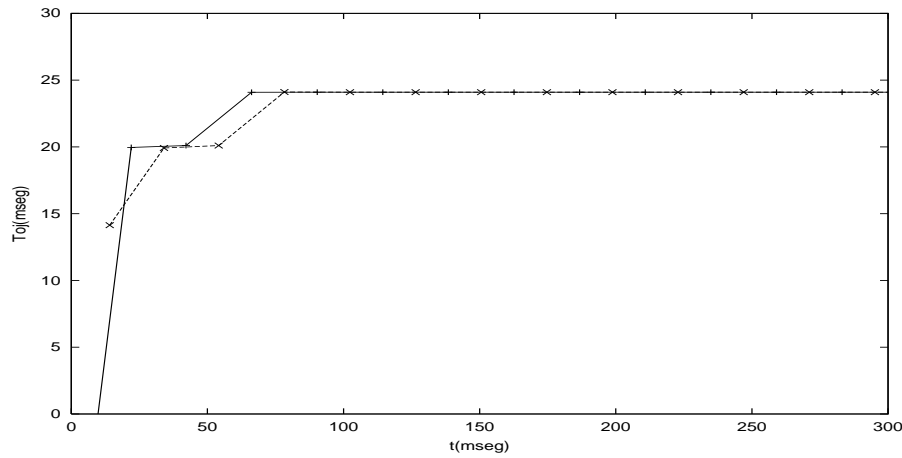


Figura 7.7: Intervalos entre picos T_o del tren de pulsos de salida, para la neurona 1 (línea continua) y neurona 2 (línea punteada) con tiempo de retraso $\tau_d = 10$ mseg, y la entrada tiene un intervalo entre disparos de $T_i = 20$ mseg.

10 mseg (figura 7.7) podemos ver que el intervalo entre picos T_o no es 24.10 mseg desde el comienzo sino que oscila alrededor de este valor, comienza con valores de $T_{o1} = 20.00$ mseg y $T_{o2} = 19.96$ mseg y rápidamente se acerca a $T_o = 24.10$ mseg.

En las figuras 7.8 y 7.9 vemos como cambiar el tiempo de retraso afecta la convergencia del intervalo entre disparos T_o de la salida. En el primer caso (figura 7.8) se tiene un tiempo de retraso de $\tau_d = 13.75$ mseg, para el cual el intervalo entre picos de salida T_o oscila entre $T_o = 20.00$ mseg y $T_o = 12.36$ mseg, finalmente converge a $T_o = 15.93$ mseg.

Para el caso de $\tau_d = 20$ mseg el intervalo entre picos diverge lentamente como se puede observar en la figura 7.9.

7.3 Acople de N neuronas.

7.3.1 Pesos sinápticos

El peso sináptico entre una neurona pre-sináptica j y una neurona post-sináptica k usualmente se asume para el modelo de Hopfield[11] de una memoria asociativa como:

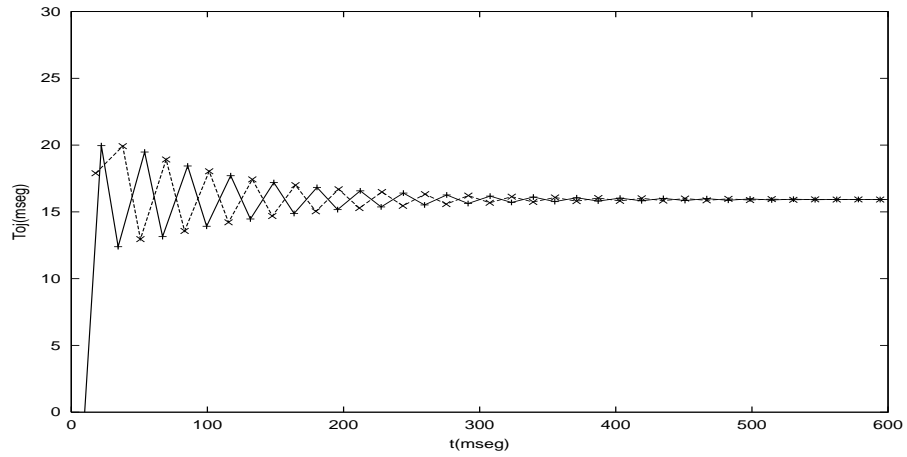


Figura 7.8: Intervalos entre picos T_o del tren de pulsos de salida, para la neurona 1 (línea continua) y neurona 2 (línea punteada) con tiempo de retraso $\tau_d = 13.75 \text{ mseg}$, y la entrada tiene un intervalo entre disparos de $T_i = 20 \text{ mseg}$.

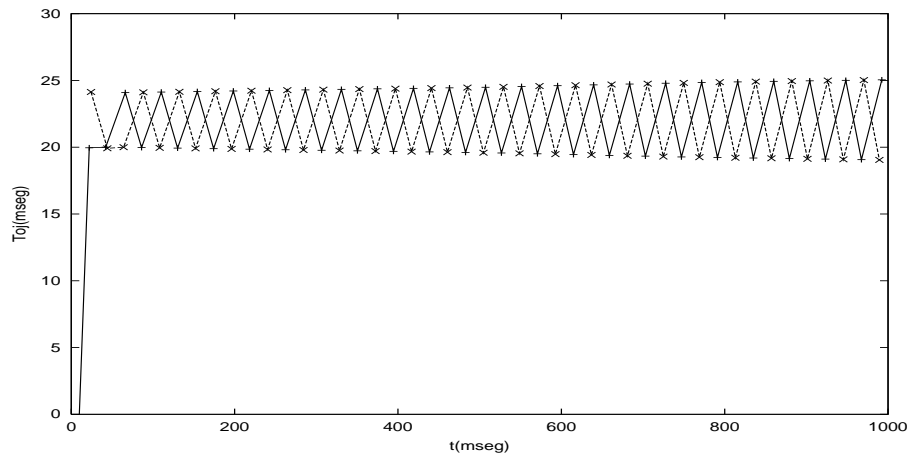


Figura 7.9: Intervalos entre picos T_o del tren de pulsos de salida, para la neurona 1 (línea continua) y neurona 2 (línea punteada) con tiempo de retraso $\tau_d = 20 \text{ mseg}$, y la entrada tiene un intervalo entre disparos de $T_i = 20 \text{ mseg}$.

$$W_{jk} = \sum_{\mu=1}^P (\xi_j^{(\mu)} - a)(\xi_k^{(\mu)} - b) \quad (7.12)$$

donde P es el número de patrones que se van a almacenar, $\xi_j^{(\mu)} = 0/1$ denota el estado (callado o activo) de la neurona j . a y b están dados por la actividad promedio, $a = b \equiv \langle \xi_j^{(\mu)} \rangle$.

Se ha experimentado con reglas más sencillas como

$$W_{jk} = \sum_{\mu=1}^P \xi_j^{(\mu)} \xi_k^{(\mu)} \quad (7.13)$$

aunque no han demostrado ser muy eficientes recaudando patrones memorizados[18], se adoptó la siguiente regla para la memorización de patrones, la misma usada por Hasegawa[14].

$$W_{jk} = \theta\left(\sum_{\mu=1}^P \xi_j^{(\mu)} \xi_k^{(\mu)}\right) \quad (7.14)$$

θ es la función escalón. Biológicamente esta regla no es justificable, salvo en pocos casos, ha sido demostrado en varios artículos que esta regla es más eficiente en la memorización de patrones para redes neurales del tipo Hopfield[11]. El peso sináptico dado por la ecuación 7.14 puede ser 0 o 1 mientras los pesos dados por 7.12 y 7.13 tienen un rango mucho más grande.

7.3.2 Patrones

Los patrones que se van a almacenar, se expresan en un vector $\xi^{(\mu)} = \{\xi_j^{(\mu)} \mid j = 1 - N\}$ donde $\mu = 1 - P$ y $\xi_j^{(\mu)} = 0(1)$ refiriéndose al estado de la neurona j (callada o activa), P es el número de patrones que se van a memorizar.

Los patrones fueron creados aleatoriamente tal que cumplieran la condición $M = fN$, donde

$$M = M^{(\mu)} = \sum_j \xi_j^{(\mu)} \quad (7.15)$$

representa el número de neuronas prendidas y f es el porcentaje de neuronas prendidas. Para facilitar el análisis se escogió para el primer patrón ($\mu = 1$) que:

$$\begin{aligned}\xi_j^{(\mu)} &= 1, & j &\leq M \\ \eta_j(t) &= 0, & j &> M\end{aligned}\quad (7.16)$$

Para crear los patrones se utilizó el generador de números aleatorios de Numerical Recipes “ran1.c”[19].

Estos patrones quedan almacenados en los pesos sinápticos, entre más patrones se almacenen habrá más conexiones, es decir, $W_{ij} \neq 0$. Una neurona j y una neurona k que tengan $\xi_j^{(\mu)} = \xi_k^{(\mu)} = 1$, van a tener una conexión $W_{jk} = 1$.

7.3.3 Recuperación de patrones

Una vez la red tiene memorizados los patrones, la entrada I_j^{ext} (ecuación 7.3) va a ser la responsable de la recuperación de los patrones.

Para recuperar un patrón se necesita otro vector $\zeta = \{\zeta_j \mid j = 1 - N\}$, este vector dice cuales neuronas van a recibir una entrada $I_j^{ext} \neq 0$ y cuales no. Una neurona j que tengan $\zeta_j = 1$ recibe un pulso de corriente dado por la ecuación 7.3 con $T_{in} = 0$, y una neurona j tal que $\zeta_j = 0$ recibe una entrada $I_j^{ext} = 0$.

De esta forma para recuperar el patrón μ se debe tener que $\zeta = \xi^{(\mu)}$. Es decir para recuperar el primer patrón $\mu = 1$ debo introducir un pulso de corriente dado por la ecuación 7.3 a las primeras M neuronas, y una corriente $I_j^{ext} = 0$ para las neuronas con $j > M$.

Cuando una neurona j recibe un pulso de corriente se excita y transmite otro pulso de corriente a todas las neuronas con las que tenga establecida una conexión ($W_{jk} = 1$). Estas neuronas también se van a excitar y otra vez transmitirán el pulso a las neuronas con quienes están conectadas, de esta forma oscilan únicamente las neuronas que están conectadas entre si. Cada patrón almacenado funciona como un atractor, una vez todas las neuronas activas de uno de los patrones almacenados estén oscilando, el sistema va a tender a quedarse en ese estado.

Para medir que tan exacto se están recuperando los patrones utilizamos una cantidad m_i dada por

$$m_i = \frac{\sum_j [2\xi_j^{(\mu)} - 1][2\zeta_j - 1]}{N}\quad (7.17)$$

se puede ver que $m_i = 1$ cuando $\xi_j^{(\mu)} = \zeta_j$, la evolución en el tiempo de esta cantidad m la puedo expresar como

$$m(t) = \frac{1}{N} \sum_j [2\xi_j^{(\mu)} - 1][2\eta_j(t) - 1] \quad (7.18)$$

donde $\eta_j(t)$ se encuentra como

$$\begin{aligned} \eta_j(t) &= 1, & \text{si } U_{o,j} &= 1 \text{ en } t \in [t_{o,jm} - \delta t, t_{o,jm} + \delta t] \\ \eta_j(t) &= 0 & \text{de lo contrario} \end{aligned} \quad (7.19)$$

$t_{o,jm}$ es el m -ésimo tiempo de disparo de la neurona j y $\delta t = 2.45\tau_{sin} \sim 5 \text{ mseg}$ que es el ancho medio de la función α dada en la ecuación 5.9.

$m(t)$ se calcula cada $\tau_d + \tau_r = 12.45 \text{ mseg}$ empezando en $t=2.45 \text{ mseg}$, en estos instantes se espera obtener disparos de las neuronas en la red. Las neuronas que reciben el impulso inicial disparan $\tau_r = 2.45 \text{ mseg}$ después, este impulso se demora $\tau_d = 10 \text{ mseg}$ en llegar a las siguientes neuronas, sólo las neuronas con las que se halla establecido conexión alguna, van a recibir este impulso. Los primeros disparos se van a observar a los 2.45 mseg , y luego cada 12.45 mseg se va a observar disparos de las neuronas que estén conectadas.

Se pueden observar disparos en cualquier instante de la simulación pero sólo los que se observen en los tiempos descritos arriba van a ser parte del patrón que se desea recuperar.

7.3.4 Recuperación de patrones con una entrada perfecta

$$\zeta = \xi^{(\mu)}$$

La recuperación exitosa de un patrón μ depende de cuantos patrones P se hallan almacenado y del número de neuronas activas M que tenga cada uno de los patrones almacenados. La recuperación de un patrón se considera exitosa si $m(t) = 1.0$ después de 500 mseg (ver ecuación 7.18), para algunos casos se consideró hasta un segundo de tiempo de simulación, pero puede llegar a ser muy demorado. Entre más patrones se almacenen en la red mayor va a ser el número de conexiones en la matriz de pesos sinápticos W_{jk} (ecuación 7.14), de la misma manera influye el número de neuronas prendidas M que tenga cada patrón, entre más neuronas activas M mayor número de conexiones. Como hay más conexiones es más probable que una neurona que no es parte del patrón reciba un estímulo de otra que si es parte y comience a oscilar, por eso existe un límite de patrones P^* donde la recuperación va a ser exitosa sin importar el patrón que se esté buscando.

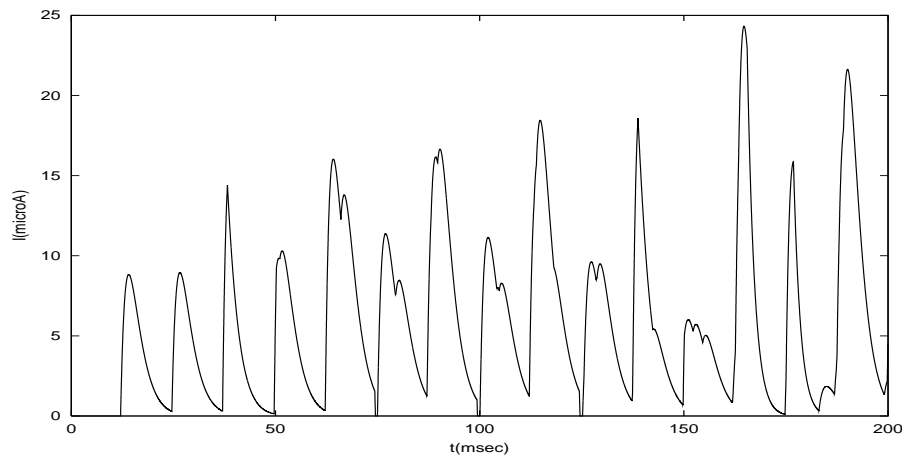


Figura 7.10: Corriente de acople (ecuación 7.6) durante la recuperación de un patrón, en una red de $N=100$, se almacenaron 60 patrones y $M=10$ neuronas activas.

La figura 7.10 muestra una corriente de acople I_{int} (ecuación 7.6) durante la recuperación de un patrón, se observa que comienza a llegar corriente a la neurona después de 12.45 msec, este es el tiempo de retraso sumado al tiempo de respuesta de la neurona. También se observa que esta corriente es una superposición de todas las corrientes provenientes de todas las neuronas con las cuales existe una conexión. La corriente de acople muestra picos periódicos con un periodo de 12.45 msec, esto se debe a que todas las neuronas se demoran 2.45 msec. en responder a una entrada y que el impulso se demora $\tau_d = 10$ msec en atravesar el axón.

La figura 7.11 muestra la recuperación del patrón $\mu = 1$ cuando se han almacenado 50 patrones aleatorios con el 10% de las neuronas prendidas, en este caso la recuperación es exitosa. También se añadieron nuevos patrones con el mismo número de neuronas prendidas para encontrar el número máximo de patrones P^* que podía almacenar la red. En ese caso $P^* = 55$ patrones, se puede notar que este valor puede cambiar ya que los patrones son generados aleatoriamente. Memorizar un menor número de patrones significa siempre una exitosa recuperación. Para cualquier número de patrones almacenados menor a 55 se puede recuperar perfectamente un patrón durante 1 segundo. Se tomo 1 segundo como tiempo de simulación por razones computacionales y por que se consideró que era un tiempo suficientemente largo para simular cualquier proceso biológico que se pueda estar llevando a cabo en el cerebro, de todos

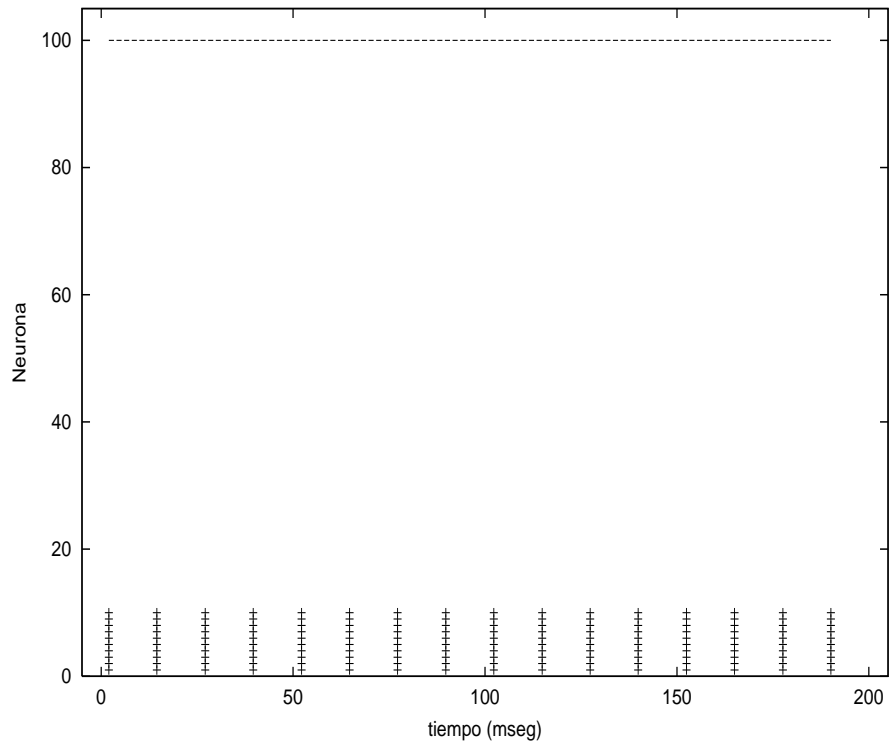


Figura 7.11: Muestra la evolución en el tiempo de la variable $m(t)$ (línea punteada) y la variable η_j (cruces para $\eta_j = 1$), para las 100 neuronas de la red, el patrón que se recupera es el primer patrón $\mu = 1$ de $P=50$ almacenados, consiste en las 10 primeras neuronas prendidas ($M=10$).

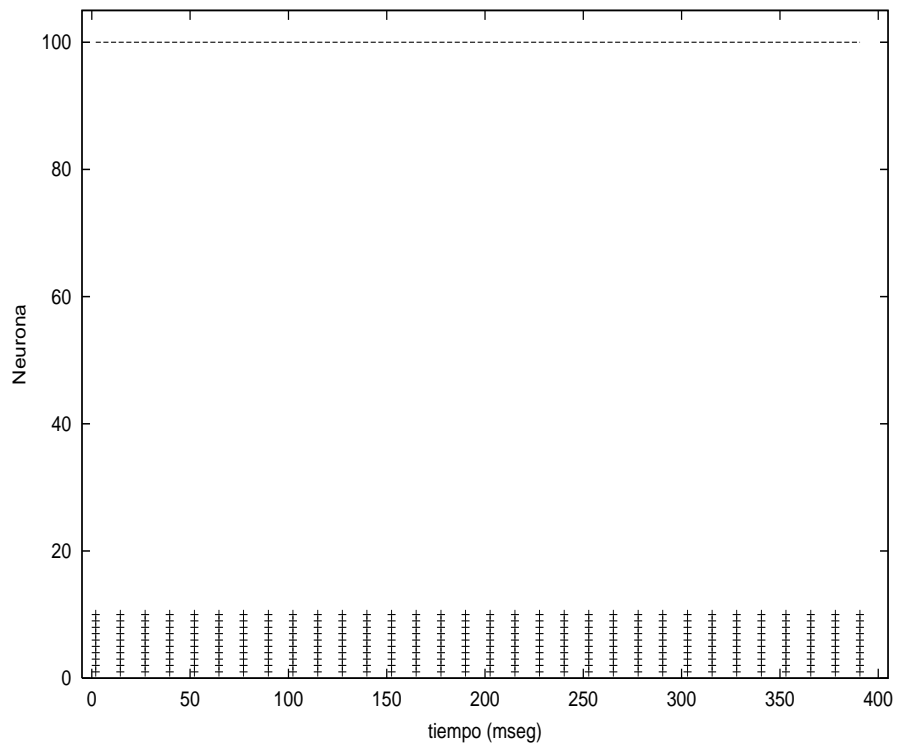


Figura 7.12: Muestra la evolución en el tiempo de la variable $m(t)$ (línea punteada), y la variable η_j (cruces para $\eta_j = 1$), para las 100 neuronas de la red, el patrón que se recupera es el primer patrón de 55 almacenados que consiste en las 10 primeras neuronas prendidas ($M=10$), la simulación se corrió por 1 segundo sin mostrar decaimiento.

modos la memoria no puede ser eterna.

La recuperación exitosa de un patrón no depende del índice del patrón que se desea recuperar, (figura 7.13 y figura 7.15). La memorización es independiente del patrón, esto se debe a la forma en que esta organizada la red. Existe una cierta simetría en la red tal que todas las neuronas comparten las mismas conexiones, esto hace que todos los patrones sean equivalentes.

Para el caso de $P = P^* + 1$ (56 patrones almacenados en el caso mencionado anteriormente, figura 7.14), se observa una solución estable, mas no perfecta. Después de la primera oscilación comienza a oscilar una neurona (77) que no hace parte del patrón, en la cuarta oscilación la neurona 31 dispara y vuelve a disparar en la sexta oscilación. Luego en la séptima oscilación la neurona 33 comienza a oscilar y al parecer se llega a una configuración estable. Aunque

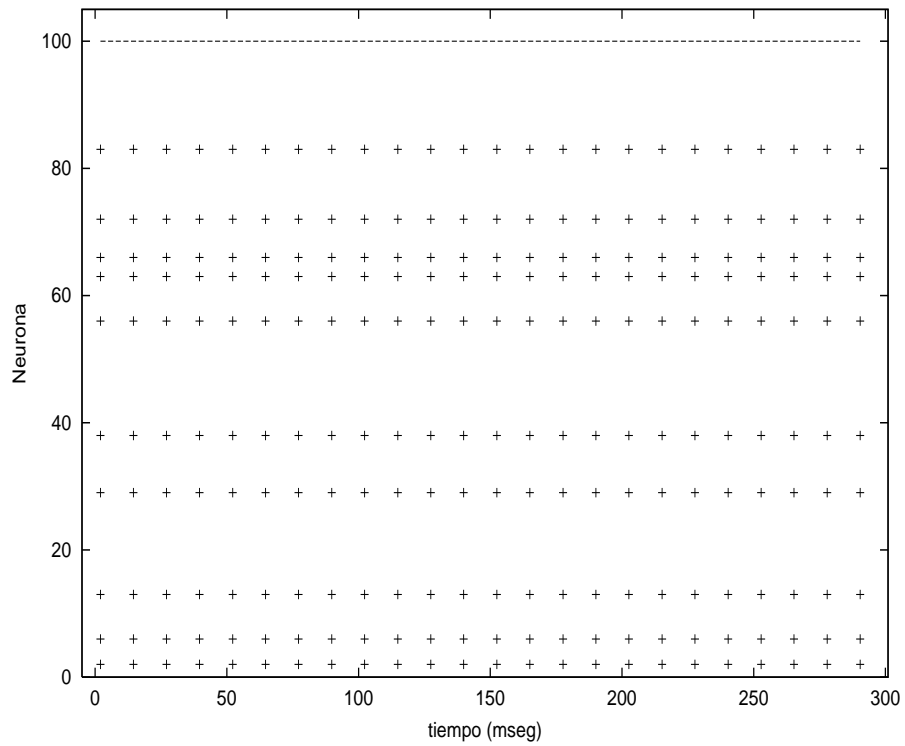


Figura 7.13: Recuperación de un patrón cualquiera para una red con 55 patrones almacenados, $M=10$, esta simulación se corrió por 1 segundo sin mostrar decaimiento.

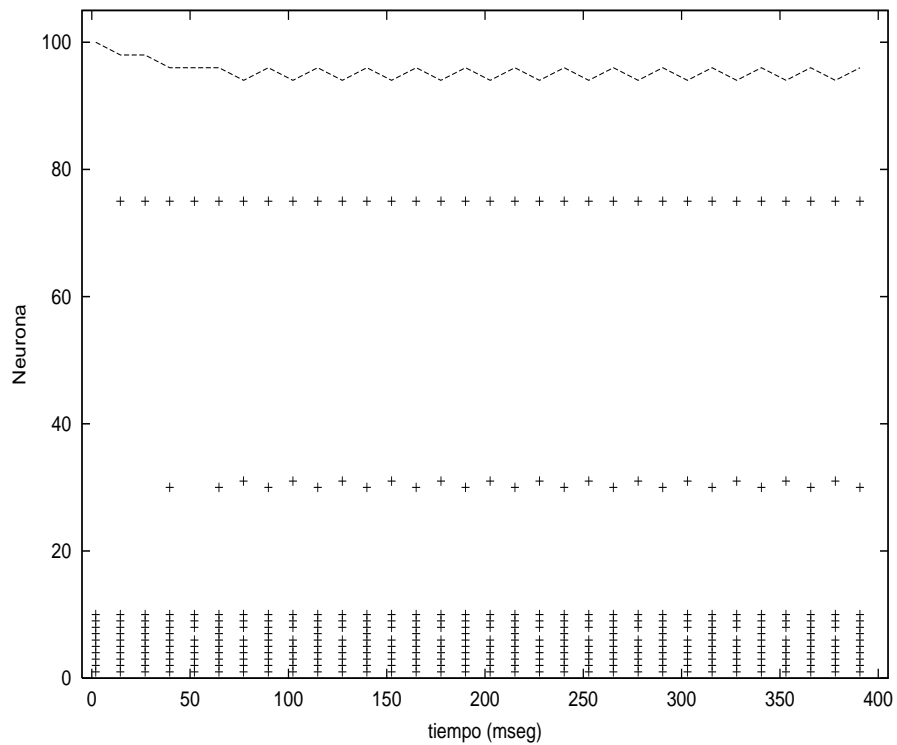


Figura 7.14: Recuperación del patrón $\mu = 1$ con $M=10$ y $P = P^* + 1 = 56$ patrones almacenados. Se observa estabilidad aunque el patrón no se puede recuperar a la perfección. La simulación se corrió para 1 segundo sin mostrar cambio.

el patrón no se puede recuperar perfectamente, parte del patrón es recuperable por lo menos durante un segundo, tiempo que duró la simulación.

El mismo comportamiento se observa cuando se almacenan otros patrones. La red es capaz de almacenar hasta un cierto número de patrones P^* de manera estable, a partir de este número P^* cualquier patrón adicional que se almacene vuelve inestable la recuperación de cualquiera de los patrones almacenados previamente.

El número P^* en muchos casos se encuentra cerca a 55 para $M=10$ y disminuye a medida que aumenta M .

En las figuras 7.16 y 7.17 podemos ver la recuperación del primer patrón almacenado $\xi^{(1)}$ con $M=10$, cuando $P > P^*$. Se utilizaron los mismos patrones de los casos anteriores, en el caso de $P = P^* + 3 = 58$ después de los 100 mseg ya tres neuronas que no pertenecen al patrón están prendidas, se pueden observar

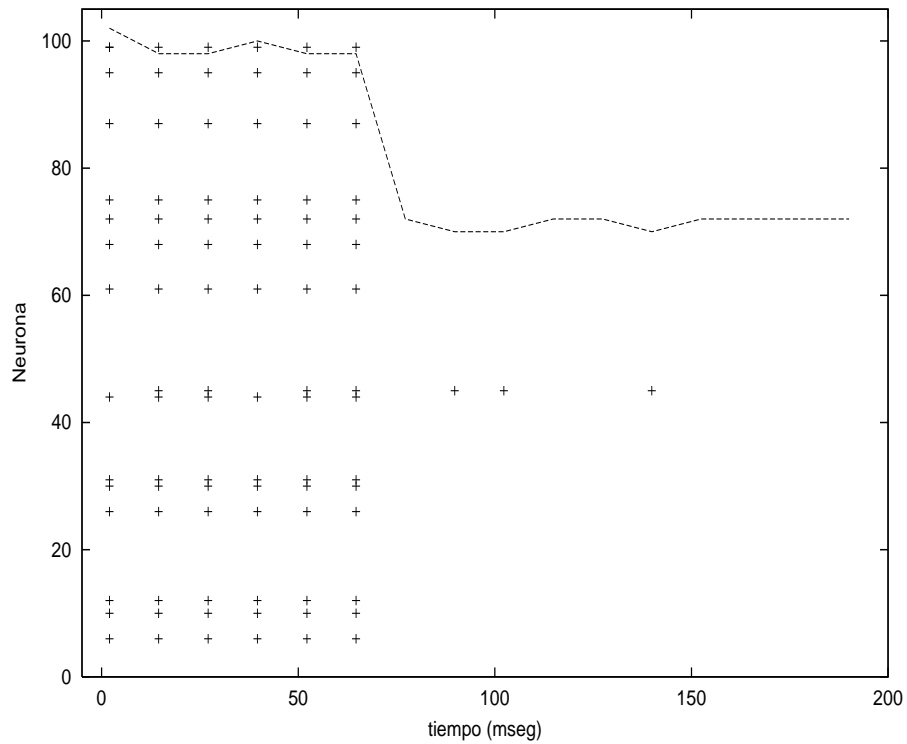


Figura 7.15: Esta figura muestra la evolución de la recuperación del patrón número 34, en una red con 56 patrones almacenados, $M=10$ (diez neuronas prendidas). Se observa recuperación parcial durante los primeros 50 mseg, luego es imposible recuperar la información, podemos ver que la neurona 44 comienza a oscilar desde la segunda oscilación. Esta neurona oscilando fuera de tiempo es la que daña la recuperación.

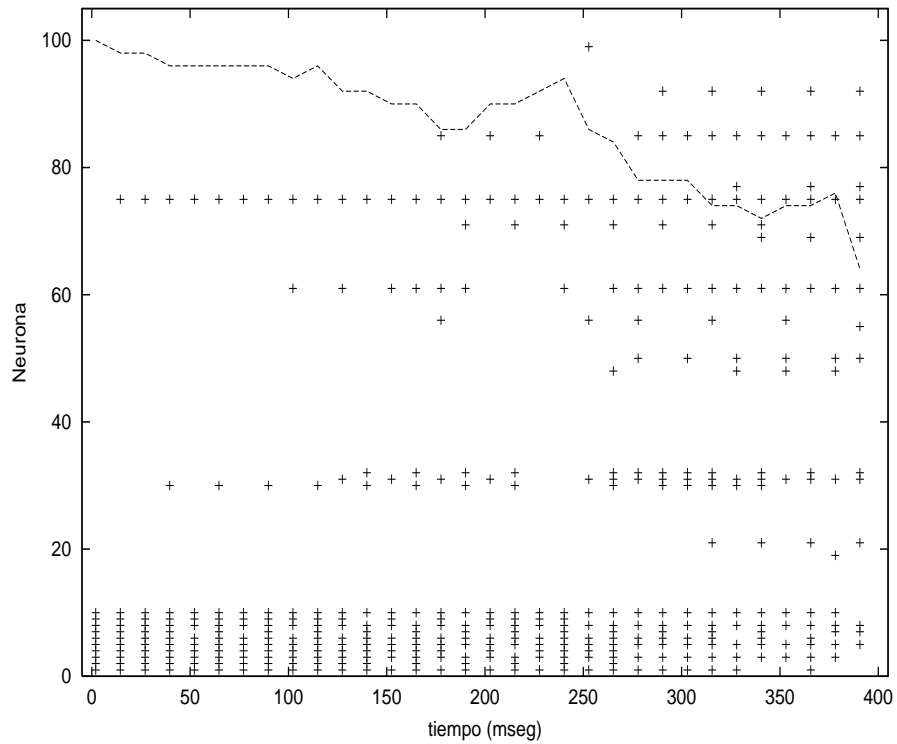


Figura 7.16: Muestra la evolución en el tiempo de la variable $m(t)$ (línea punteada), y la variable η_j (cruces para $\eta_j = 1$). Para las 100 neuronas de la red, el patrón que se recupera es el primer patrón $\xi^{(1)}$ que consiste en las 10 primeras neuronas prendidas ($M=10$) de 58 almacenados.

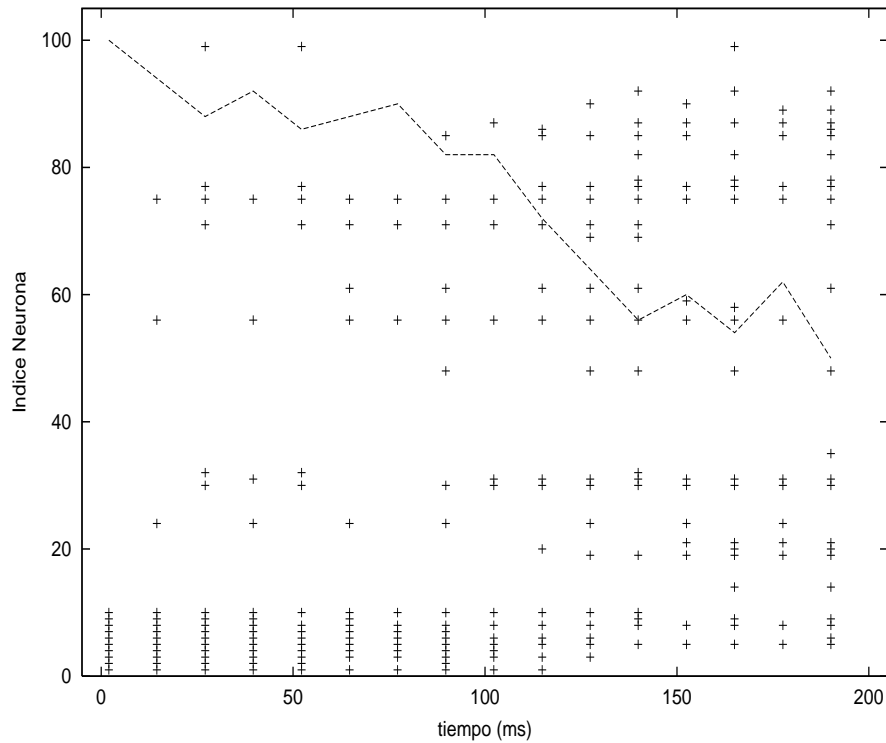


Figura 7.17: Muestra la evolución en el tiempo de la variable $m(t)$ (línea punteada) y la variable η_j (cruces para $\eta_j = 1$). Para las 100 neuronas de la red, el patrón que se recupera es el primer patrón $\xi^{(1)}$ de 70 almacenados que consisten en las 10 primeras neuronas prendidas ($M=10$).

rastros del patrón hasta 350 mseg, después es irreconocible. El caso de $P=70$ es más radical después de 100 mseg ya no se reconoce patrón alguno.

7.3.5 Recuperación de patrones con entrada imperfecta $\zeta \neq \xi^{(\mu)}$ (ruido)

Para estudiar la recuperación de los patrones con una entrada con ruido, se utilizaron los mismos patrones que se almacenaron para generar las figuras 7.12 y 7.13. La matriz de pesos sinápticos es la misma, ahora la diferencia es que para recuperar un patrón se utiliza un vector ζ ligeramente diferente. La idea es observar cual es el efecto del ruido en la recuperación del patrón en la red.

Se consideró el caso en que $\zeta \neq \xi^{(\mu)}$ para la obtención del patrón $\xi^{(\mu)}$ (ver sección 7.3.3). En este caso se utilizó el patrón $\xi^{(1)}$ con $M=10$ que consiste

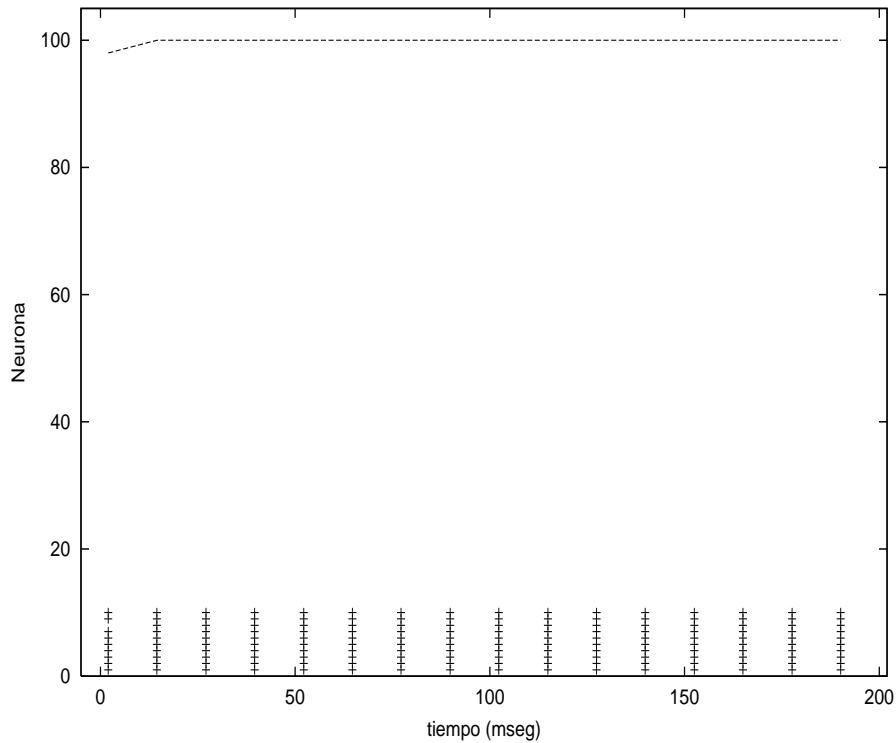


Figura 7.18: Recuperación del patrón $\xi^{(1)}$ con un vector de entrada $\zeta \neq \xi^{(1)}$. Se almacenaron 55 patrones con $M=10$. Se apagó la neurona 8 que en el patrón original estaba prendida. En la recuperación $\zeta_8 = 0$ y debería ser 1.

en las primeras 10 neuronas prendidas y el resto apagadas. En el vector ζ se apagaron ciertas neuronas que deberían estar activas y se prendieron otras que deberían estar apagadas y así observar si el patrón era recuperable. En muchos casos la recuperación fue estable, en otros no.

Apagar una neurona no muestra problema alguno en la recuperación del patrón (figura 7.18). Cuando se apagó la neurona número 8, en la primera oscilación $m(t) < 1$ pero ya en la segunda oscilación $m(t) = 1.0$. A partir de aquí el patrón recuperado es el patrón almacenado y es estable durante todo el resto de la simulación (un segundo).

En la figura 7.19 se apagan las neuronas 5 y 8 que deberían estar prendidas. El efecto es similar al caso anterior, las neuronas 5 y 8 tienen conexiones con las otras neuronas que hacen parte del patrón $\xi^{(1)}$. Las neuronas que reciben el estímulo inicial ζ , hacen que se activen 12.45 mseg después las neuronas 5 y 8.

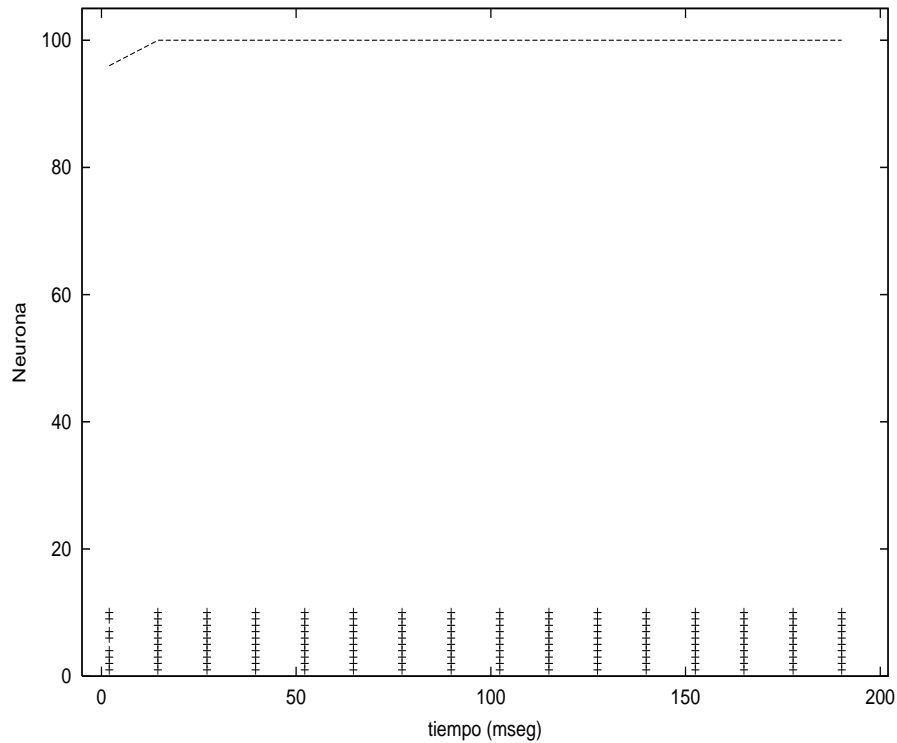


Figura 7.19: Recuperación del patrón $\xi^{(1)}$ con un vector de entrada $\zeta \neq \xi^{(1)}$. Se almacenaron 55 patrones con $M=10$. Para la recuperación en el vector de entrada $\zeta_8 = 0$ y $\zeta_5 = 0$ deberían ser 1.

La figura 7.20 muestra el caso en que se tienen 5 neuronas del patrón apagadas, es decir se intenta recuperar un patrón con solo 50% del patrón almacenado. El patrón se observa por durante 100 mseg, aunque no es una recuperación perfecta, después de los 100 mseg no hay patrón alguno observado.

En la figura 7.21 se observa el caso en que una neurona fuera del patrón se prende, en otras palabras el vector de entrada tiene las 10 primeras neuronas prendidas y adicionalmente se prende otra neurona, en este caso es la neurona número 58. Se observa que la recuperación del patrón es satisfactoria, el patrón no es 100% recuperable durante un pequeño transiente, luego se estabiliza y se recupera en un 100%.

Para el caso de una recuperación con dos neuronas prendidas adicionales a las del patrón original (figura 7.22), se puede reconocer el patrón por casi 200 mseg, después de esto ya no es recuperable. En el caso de tres neuronas

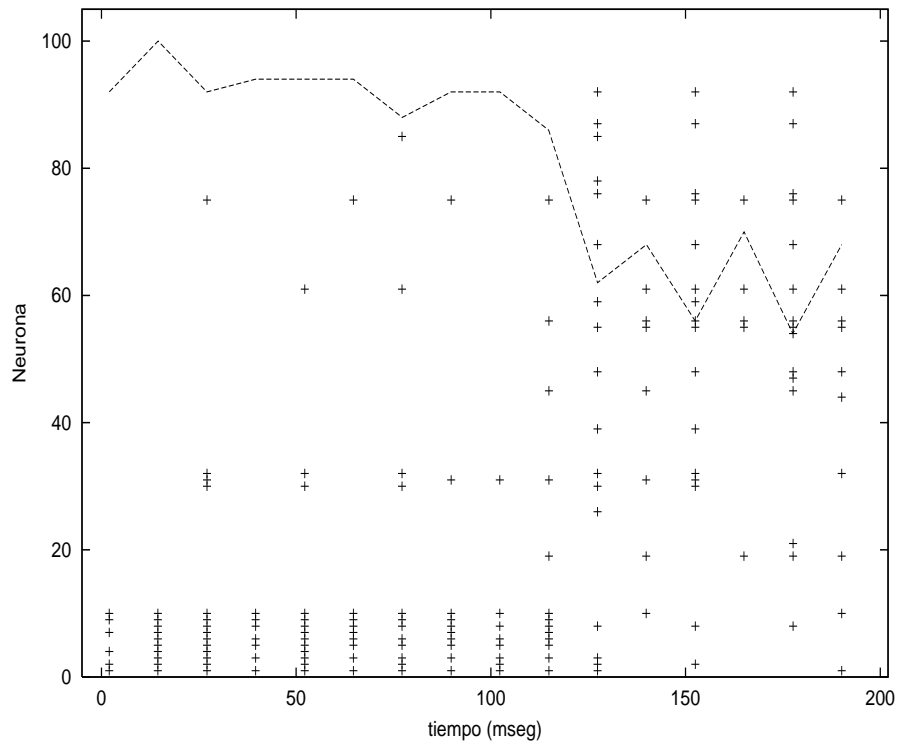


Figura 7.20: Recuperación del patrón $\xi^{(1)}$ con solo el 50% del patrón. Se almacenaron 55 patrones con $M=10$.

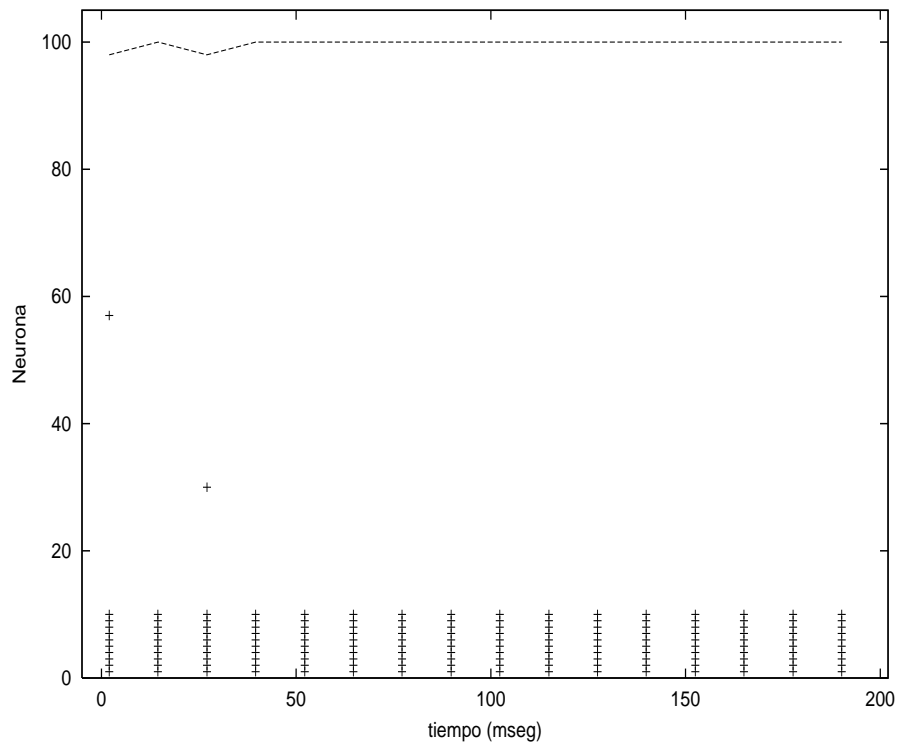


Figura 7.21: Muestra la recuperación del patrón $\xi^{(1)}$ para una entrada con ruido. En la entrada se prende la neurona 58 que no hace parte del patrón. En la red hay almacenadas 55 patrones con $M=10$.

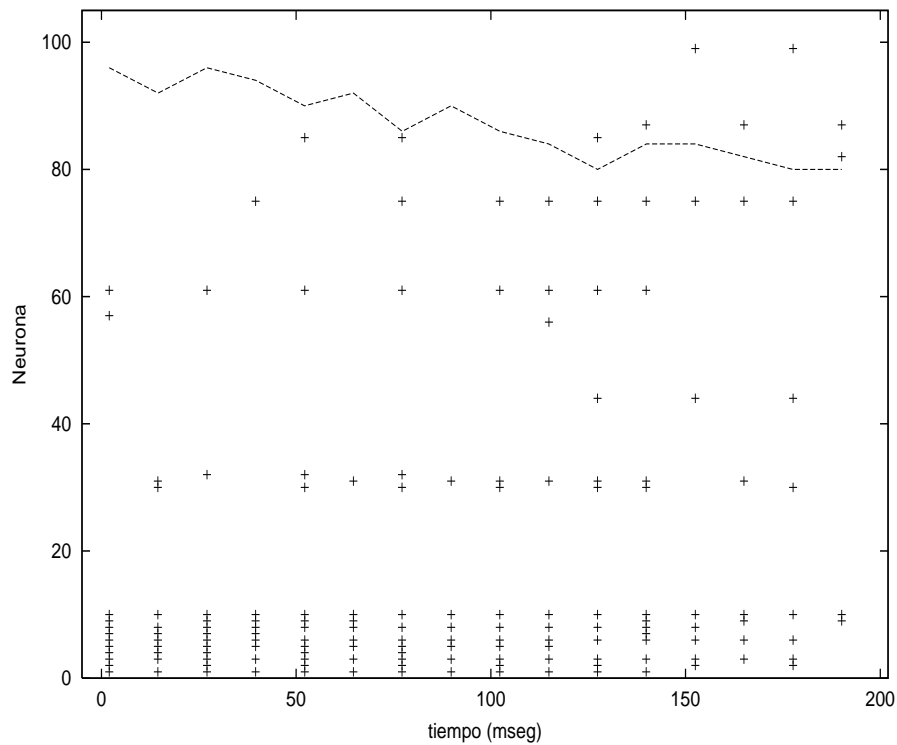


Figura 7.22: Recuperación del patrón $\xi^{(1)}$ con dos neuronas prendidas adicionales al patrón original. En la red hay almacenadas 55 patrones con $M=10$.

prendidas adicionales a las del patrón original no se logra diferenciar por mucho tiempo el patrón, la recuperación es deficiente (figura 7.23).

En la figura 7.24 se observa la recuperación del mismo patrón con otro ruido, en este caso se apagó una neurona del patrón y se prendió otra que no pertenecía al patrón original. El patrón observado es una buena aproximación del patrón almacenado, pero es inestable. Después de 200 mseg se apaga, antes de los 200 mseg se observan oscilaciones de $m(t)$, en cada iteración se prenden o se apagan 1 o 2 neuronas, pero en los 200 mseg se apagan todas. El patrón es recuperable aunque no en su totalidad por los primeros 200 mseg.

En la figura 7.25 se observa un intento de recuperar el patrón $\xi^{(1)}$ cuando en la entrada se apagan dos de las neuronas del patrón original y se prenden otras dos que no hacen parte de este patrón, lo observado es que el patrón recuperado se parece a $\xi^{(1)}$ en las primeras oscilaciones, pero después de un tiempo comienza a parecerse a $\xi^{(34)}$ y finalmente desaparece. El patrón $\xi^{(34)}$ tiene un $M=15$

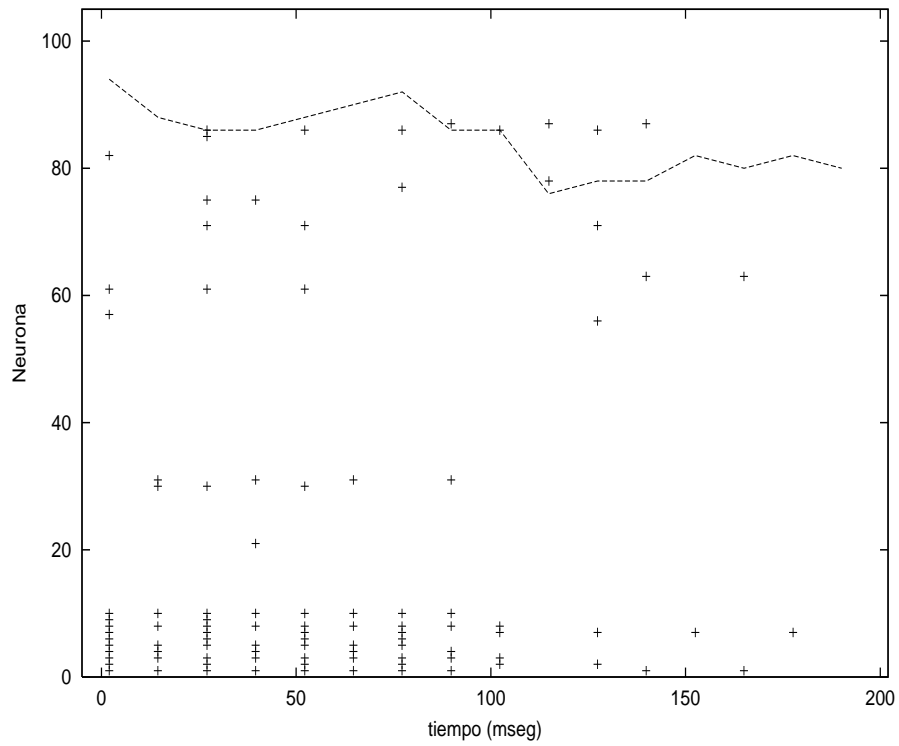


Figura 7.23: Recuperación del patrón $\xi^{(1)}$ con tres neuronas prendidas adicionales al patrón original. En la red se almacenaron 55 patrones con $M=10$.

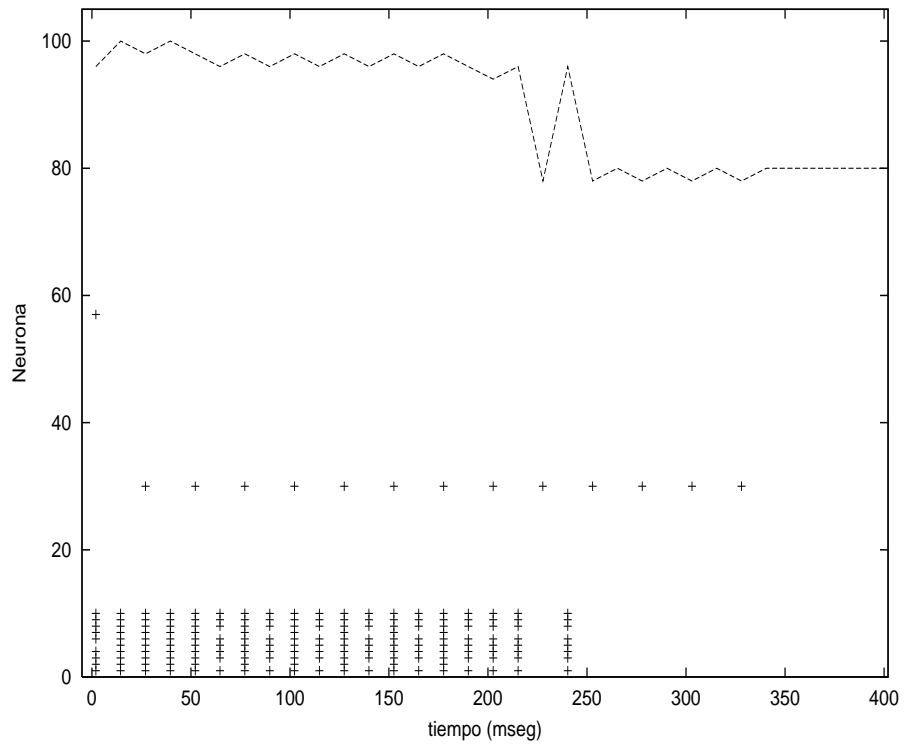


Figura 7.24: Recuperación del patrón $\xi^{(1)}$ con ruido, se prendió la neurona 58 y se apagó la neurona 5. En la red están almacenados 55 patrones con $M=10$.

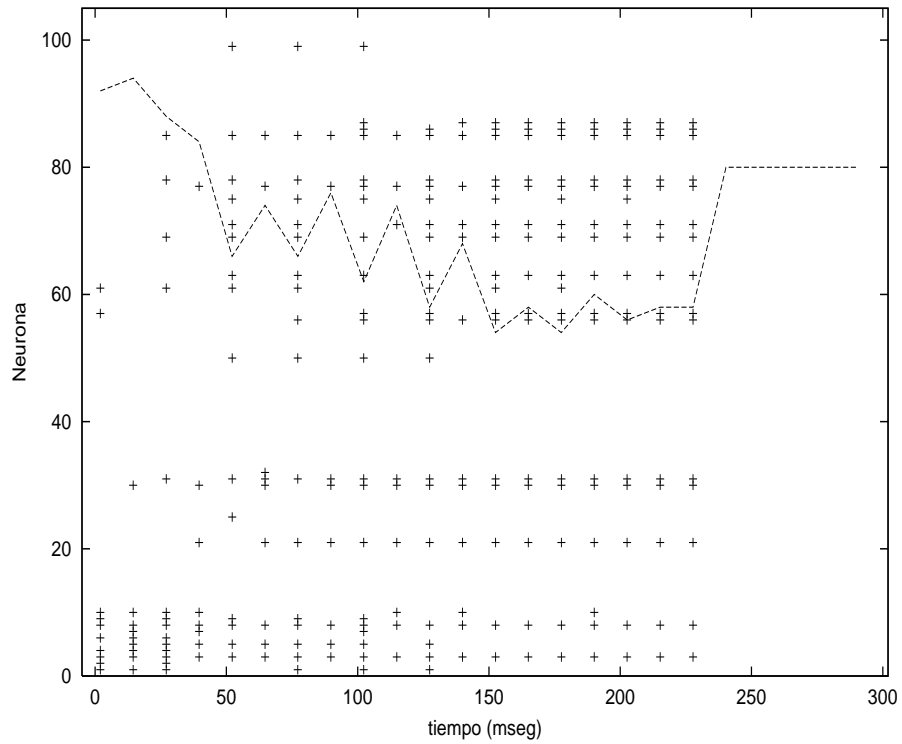


Figura 7.25: Recuperación del patrón $\xi^{(1)}$ con una entrada ζ con ruido. El ruido consiste en apagar dos de las neuronas pertenecientes al patrón y prender dos neuronas adicionales. En la red se almacenaron 55 patrones con $M=10$.

(15 neuronas activas), es un caso extraordinario ya que debería tener solo 10 neuronas prendidas, debido a que los patrones se generan aleatoriamente esto es posible, posiblemente con patrones más homogéneos no se hubiera dado este caso. Es posible pasar de un patrón a otro como se observa en este ejemplo. Con una entrada arbitraria las neuronas estimuladas comienzan a oscilar y estimulan a otras con las que halla una conexión establecida, es probable que se termine en uno de los patrones almacenados inicialmente. Los patrones almacenados son estados estables y atractores, cuando la red llega a uno de estos estados tiende a quedarse ahí.

7.3.6 Número de Neuronas activas M

Al aumentar el número de neuronas prendidas M y con el mismo número de patrones $P = 55$ almacenados, se observa que la red es capaz de almacenar un menor número de patrones $P^* < P = 55$ perfectamente. En las figuras 7.26 y 7.27 se observa la recuperación del primer patrón almacenado $\xi^{(1)}$, en el caso de $M=11$ se puede distinguir el patrón por los primeros 100 mseg, a partir de acá la recuperación se degenera rápidamente. En el caso de $M=15$ es mucho más rápida la degeneración del patrón, en menos de 50 mseg ya es irreconocible. Podemos ver que una neurona prendida puede volver inestable la memoria, una memoria que era capaz de memorizar 55 patrones durante un segundo con 10 neuronas prendidas, ahora con 11 neuronas prendidas se alcanza a distinguir un patrón con dificultad por solo 100 mseg.

Se utilizó una búsqueda binaria para encontrar cuantos patrones se podían almacenar para cada valor de M (número de neuronas activas en cada patrón). Primero se generaban patrones para un M específico y se calculaba la matriz de pesos sinápticos. Luego se corría la simulación para recuperar uno de los patrones, si la recuperación era exitosa se añadían más patrones y se comenzaba de nuevo, si no era exitosa se eliminaban patrones y se comenzaba nuevamente hasta encontrar el número de patrones P^* que se podían almacenar para cada M .

Este procedimiento se llevó a cabo con dos reglas de aprendizaje diferentes, los pesos sinápticos se calcularon utilizando las dos reglas dadas por Willshaw[20] (ec. 7.13 y ec. 7.14). Para elaborar un promedio se corrieron varias simulaciones en cada caso.

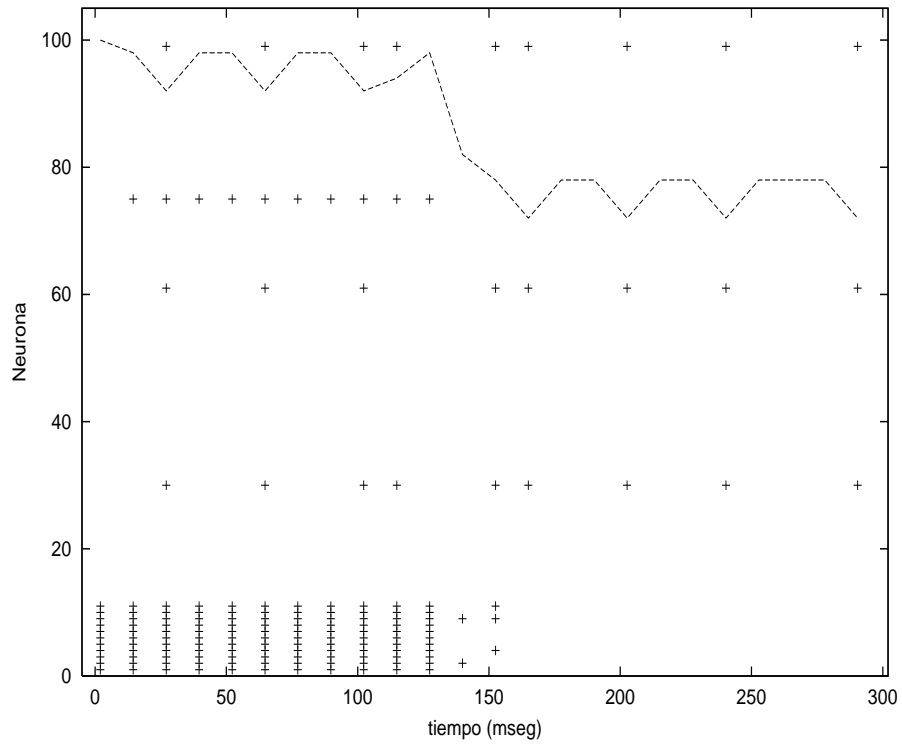


Figura 7.26: Recuperación del patrón $\xi^{(1)}$ con $M=11$ y $P=55$ patrones almacenados. Se observa una recuperación aceptable del patrón hasta 120 mseg a partir de aquí el patrón es irrecuperable.

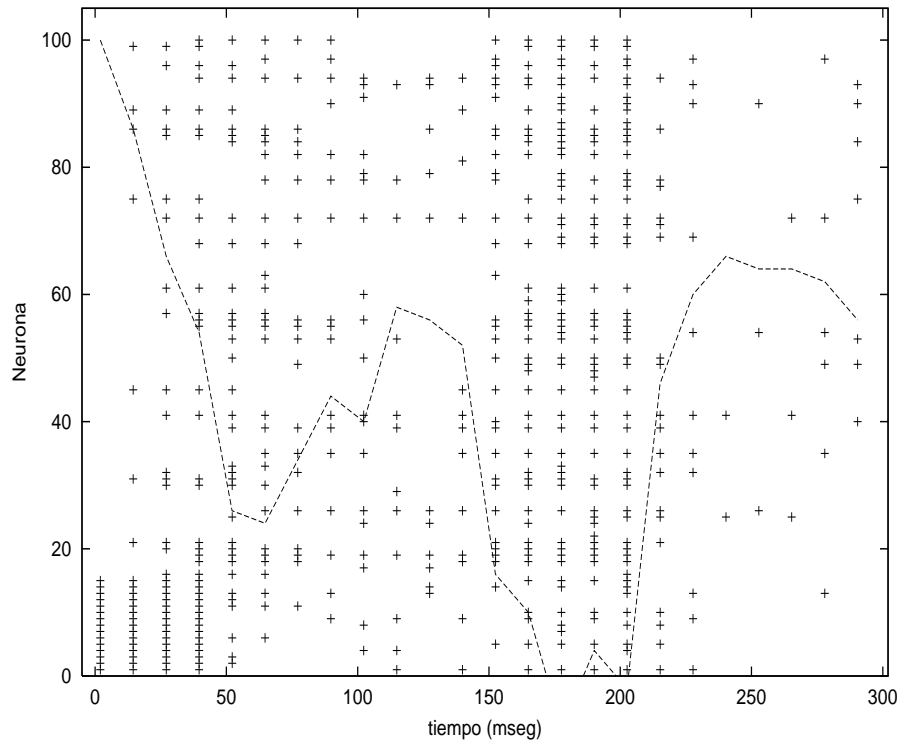


Figura 7.27: Recuperación del patrón $\xi^{(1)}$ con $M=15$ y $P=55$ patrones almacenados. Se observa una recuperación aceptable del patrón hasta por 30 mseg, a partir de aquí el patrón es irrecuperable.

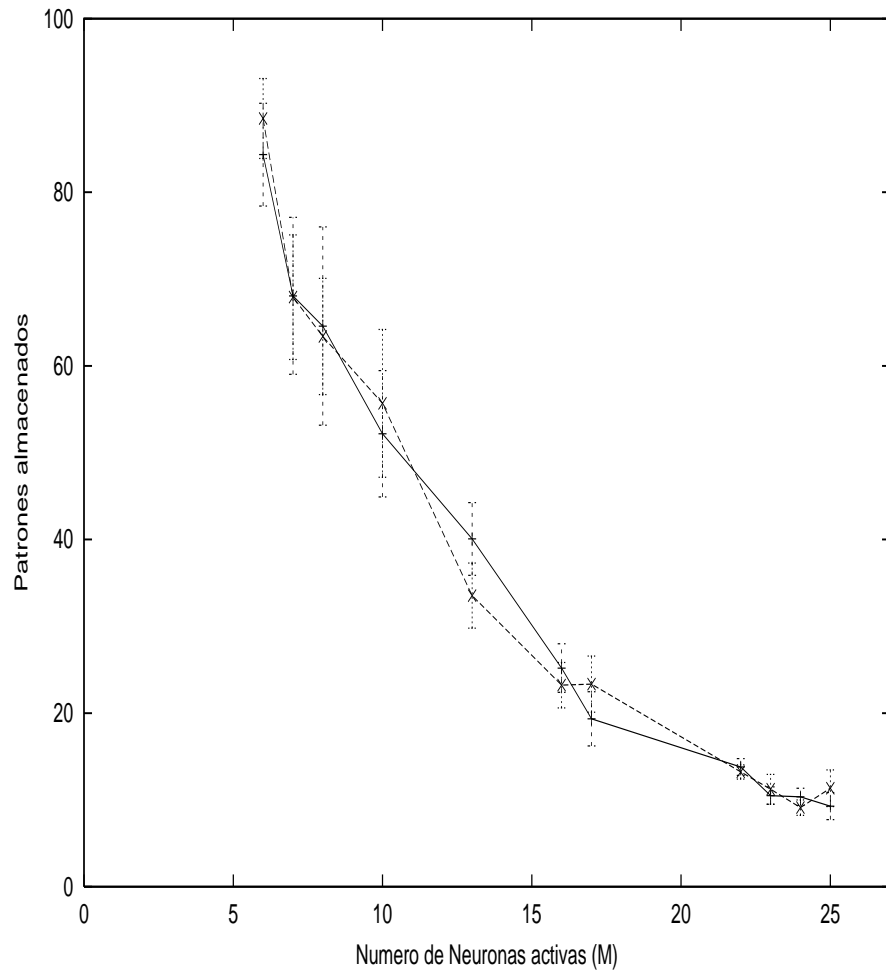


Figura 7.28: Gráfica M contra P^* para una red neuronal de 100 neuronas. Los puntos en la gráfica son el promedio de 15 simulaciones y las barras de error, la desviación estándar.

7.3.7 Número de neuronas N

Para encontrar como influye el número de neuronas en la capacidad de almacenar patrones, se utilizó el método de búsqueda binaria cambiando el número de patrones almacenados para una red con N neuronas. Este procedimiento fue demorado, se implementó para las dos reglas de aprendizaje de Willshaw (ec. 7.13 y ec. 7.14) y se corrieron 15 casos por cada N . En la figura.7.29 los puntos en la gráfica son el promedio de estos 15 casos y las barras de error, la desviación estandar.

Con el resultado obtenido no se puede decir cual de las dos reglas de aprendizaje es más eficiente, lo único que se puede decir es que son muy similares. Parece que la capacidad de almacenar patrones tiene una dependencia lineal con el número de neuronas. Al aumentar el número de neuronas aumenta la capacidad de memorización como era de esperarse. Las ecuaciones de las rectas para las reglas de aprendizaje dadas por las ec. 7.13 y 7.14 son respectivamente:

$$P^* = 1.0342N - 41.088 \quad (7.20)$$

$$P^* = 1.0237N - 42.73 \quad (7.21)$$

Utilizando esta información encontramos que para almacenar 1 patrón de 10 neuronas activas exitosamente se necesita una red de 41 neuronas cuando utilizamos la regla de aprendizaje dada por la ec. 7.13 y se necesitan 43 neuronas para la regla de aprendizaje dada por la ec. 7.14. Sin tener en cuenta los errores obtenidos en la gráfica se puede decir que la regla de aprendizaje dada por la ec. 7.13 es ligeramente mejor por que tiene mayor pendiente, es decir, la capacidad de almacenar patrones aumenta más rápidamente que con la otra regla de aprendizaje cuando se cambia el número de neuronas en la red.

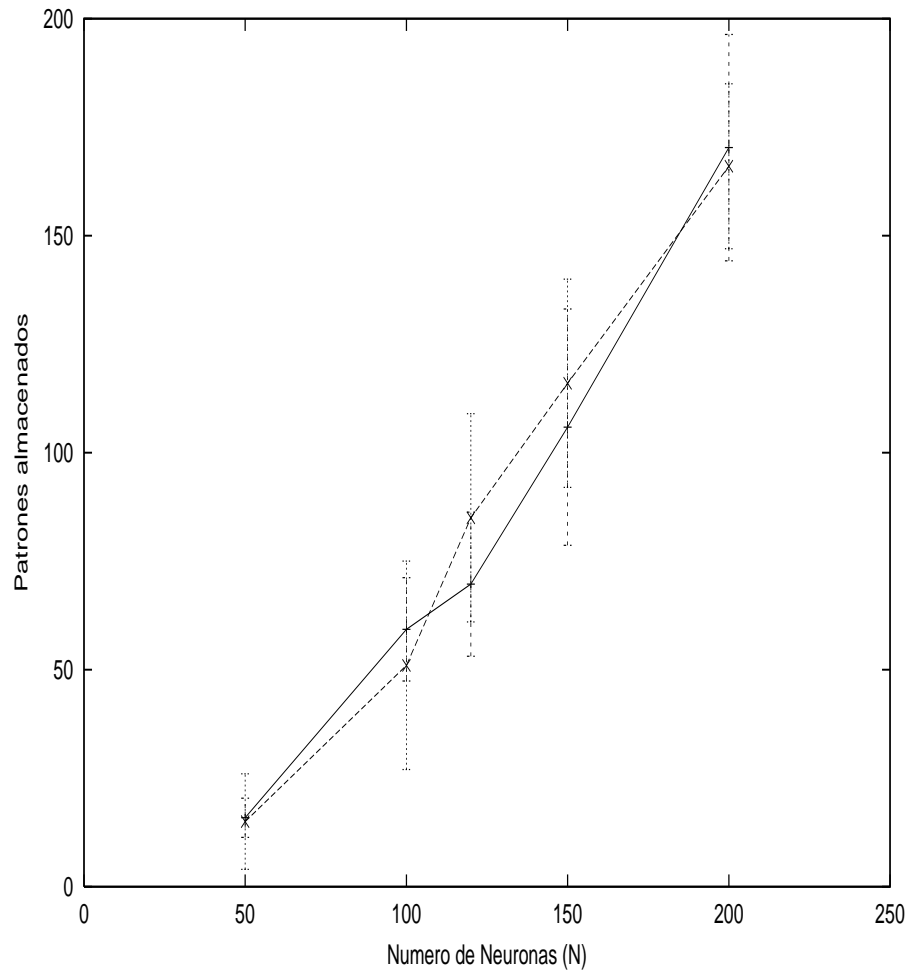


Figura 7.29: Máximo número de patrones P^* con 10 neuronas activas que se pueden almacenar para una red de N neuronas. La línea punteada es utilizando la regla de aprendizaje de Willshaw dada por la ecuación 7.13 y la línea continua con la ecuación 7.14.

Capítulo 8

Conclusión

El comportamiento no lineal de una neurona HH se aprecia en su respuesta a las diferentes entradas de corriente introducidas en la neurona, en muchos casos se observa comportamiento caótico, es decir, los tiempos entre disparos varían de una forma no lineal.

El modelo de Hodgkin y Huxley es un modelo no lineal, lo que implica que no es fácil de estudiar. La respuesta de una neurona HH presenta en muchos casos comportamiento caótico que no es sencillo de reproducir. El modelo IF puede ser una primera aproximación para entender la teoría de la codificación de la información en los tiempos de disparos. Las neuronas IF son neuronas muy sencillas que reproducen la esencia de una neurona HH.

Las neuronas HH pueden excitarse con entradas excitadoras e inhibitoras mientras que una neurona IF sólo se puede excitar con una entrada de corriente excitatoria.

En una neurona HH existe un valor umbral de corriente para que ocurra excitación de la membrana, con corrientes $I_s > 6.3 \mu A/cm^2$ se observa excitaciones periódicas, cuyo periodo aumenta en cuanto aumenta la magnitud de la corriente.

Una entrada periódica de corriente en una neurona HH tiene como resultado excitaciones con intervalos no constantes, para ciertos valores de corriente estos intervalos tienen cierta periodicidad. Un ejemplo es el caso de una entrada de periodo $T_i = 10 \text{ mseg}$, se observan 3 excitaciones en la membrana cada 40 mseg, hay tres valores diferentes de intervalos, que se repiten cada tres oscilaciones. Por cuatro pulsos de entrada se observan 3 pulsos de salida. Esta razón de pulsos en la entrada contra pulsos en la salida cambia dependiendo del periodo

en la entrada. Para el caso de $T_i = 5 \text{ msec}$ la razón es de 5:2, en muchos casos no existe esta razón, es sólo para casos particulares. Puede ser posible reproducir una salida de este estilo con una neurona IF.

Cuando se adiciona a la entrada periódica una corriente constante, la salida muestra un comportamiento caótico, los intervalos entre disparos de salida se distribuyen en un espacio acotado, cuyos límites dependen del periodo en la entrada. Adicionar un nivel DC en la entrada periódica de corriente hace que las oscilaciones en la membrana ocurran en intervalos que varían de forma caótica, esto se debe a que cada una de las contribuciones puede causar oscilaciones por sí sola. Cuando se superponen las dos entradas el comportamiento es caótico.

Es interesante ver que al variar los tiempos entre disparos en la entrada de corriente, los intervalos entre disparos en la salida se mueven alrededor de atractores, no converge ni diverge. Sería interesante aplicar teoría de caos y estudiar ese comportamiento más a fondo. Existen estados interesantes que vale la pena estudiar.

Los tiempos entre disparos de las salidas pertenecen a un espacio acotado, esto se debe tanto al tiempo refractario de la neurona que no permite tener dos disparos muy pegados, como al flujo de corriente entrando en la membrana que causa excitaciones y no deja que estén demasiado espaciadas.

Se pudo observar que una neurona de disparo al igual que la neurona HH responde a un tren de pulsos con otro tren de pulsos, que se complica a medida que el tren de pulsos de entrada se vuelve más complicado. Es difícil reproducir el comportamiento no lineal de una neurona HH con una neurona tan sencilla como es la de disparo, personalmente creo que las neuronas IF pueden llegar a ser útiles para reproducir salidas sencillas que provienen de entradas sencillas, pero nunca van a poder reproducir el comportamiento caótico de una neurona HH.

Una neurona IF no es muy útil para reproducir la salida de una neurona HH en la medida que tiene que ser ajustada a la entrada de corriente que se tenga. El ajuste no es difícil para una entrada con modulación constante (salida periódica), pero se puede complicar bastante para entradas con modulaciones más elaboradas.

La oscilación de una neurona IF es periódica para una entrada de corriente constante, este mismo caso se presenta para una neurona de tipo Hodgkin Huxley, por esta razón es sencillo ajustar la neurona IF para reproducir la salida de una neurona HH. El periodo de oscilación en una neurona IF con entrada de corriente constante depende de tres parámetros fundamentalmente, depende

de la capacitancia C de la neurona, de la constante de tiempo τ asociada a la neurona y de la magnitud de la entrada de corriente I_s . En las figuras 6.1 y 6.2 se puede apreciar como es la variación del periodo de oscilación cambiando estos parámetros. Gráficas de este estilo pueden llegar a ser muy útiles para encontrar los parámetros que reproducen la salida periódica de una neurona HH con entrada constante. Puede ser posible encontrar una relación analítica entre el periodo de oscilación de una neurona IF y los tres parámetros que describen la neurona para no tener que recurrir a tablas o a simulaciones computacionales.

Cuando una neurona IF se estimula con una entrada periódica, ajustar la neurona se vuelve complicado y tedioso, pero se puede intentar implementar con más tiempo. El comportamiento de la neurona ya no es tan sencillo, la salida presenta un comportamiento caótico.

Una entrada modelada senoidalmente también muestra comportamiento caótico en una neurona Hodgkin Huxley, este comportamiento es imposible de reproducir con una neurona tan sencilla como lo es una IF, es posible que una neurona FighzHugh-Nagumo puede tener un comportamiento similar por su naturaleza caótica. Sería interesante saber si las neuronas tipo FighzHugh-Nagumo también se pueden ajustar, estas neuronas son un poco más parecidas a las neuronas Hodgkin Huxley y tal vez presenten más propiedades que las de una neurona IF.

Dos neuronas acopladas forman un oscilador. Cuando una de las dos neuronas recibe un estímulo lo suficientemente grande para generar un potencial de acción, se genera una excitación, que por medio de una sinapsis entra a otra neurona un tiempo después. Esta otra neurona se excita y pasa el impulso nervioso nuevamente a la primera neurona, el ciclo se repite indefinidamente. Una oscilación infinita nos lleva a pensar en la idea de una memoria.

Una memoria asociativa almacena ciertos patrones en sus pesos sinápticos y asocia la salida de la red a una entrada específica. Los patrones almacenados son estados estables, además de atractores. Una vez la red llega a un estado estable tiende a quedarse ahí. Para recuperar los patrones no se necesita conocer a la perfección el patrón, sólo tener una idea de como es el patrón. Es posible encontrar algún patrón almacenado sin conocerlo, utilizando entradas aleatorias y buscando los estados estables de la red.

Los patrones se recuperan cada $\tau_d + \tau_r = 12.45 \text{ msec}$, el tiempo que demora en llegar un impulso nervioso de una neurona a otra y el tiempo que demora en reaccionar la segunda neurona a este estímulo. Una neurona excitada va a transmitir un impulso nervioso a todas las neuronas con las que se tenga

una conexión, después de un tiempo las neuronas correspondientes a un patrón estarán oscilando.

El tiempo de retraso para la transmisión del impulso nervioso de una neurona a otra es fundamental para determinar la oscilación de las neuronas, además es fundamental para poder almacenar patrones. Esto se puede apreciar en el caso de dos neuronas conectadas entre sí, cuando el tiempo de retraso determina la convergencia del intervalo entre disparos. A medida que aumentamos el tiempo de retraso de la neurona va disminuyendo la velocidad de convergencia. Sería prudente estudiar más a fondo el efecto del tiempo de retraso sobre la capacidad de memoria y sobre la convergencia que tenga la red, se escogió el valor de $\tau_d = 10 \text{ msec}$ por que muestra una convergencia rápida, además se acerca al valor medido experimentalmente para la transmisión del impulso a través del axón de la neurona.

Los patrones están almacenados en los pesos sinápticos, no se observó mucha diferencia al cambiar las reglas de aprendizaje de Willshaw, aunque en la literatura se documenta que si [18, 20], no me atrevo a decir que una es mejor que la otra, se tendría que hacer un análisis mucho más extenso para poder determinar cual es el efecto de la función escalón en la regla de aprendizaje (ecuación 7.14). Sería interesante estudiar reglas no simétricas, como la que trabaja Yoshioka [15] para una memoria asociativa de neuronas FighzHugh Nagumo, una regla no simétrica le da mas peso a la neurona pre sináptica que a la post sináptica o viceversa.

Aumentar el número de neuronas N en una red al igual que reducir el número de neuronas activas M por patrón, aumenta la capacidad de memoria. Con un mayor número de neuronas o menos neuronas activas la densidad de unos en los patrones disminuye, lo que hace que la matriz de pesos sinápticos tenga conexiones más débiles, o simplemente menos conexiones. Cuando hay más conexiones entre las neuronas, el número de unos en la matriz de pesos W_{jk} aumenta, lo que ocasiona que la recuperación del patrón no sea del todo exitosa.

Al observar la figura 7.29 se encuentra que para poder almacenar un patrón de 10 neuronas activas se necesitan alrededor de 43 neuronas como mínimo, lamentablemente no tuve tiempo para probar esto experimentalmente, ya que requiere modificar sustancialmente los programas.

Un complemento a este trabajo sería comparar la capacidad de memoria en una red de neuronas FighzHugh [15] o una red de IF [23] como la implementada por Mueller y Herz [22], con la red de neuronas tipo Hodgkin Huxley como la implementada en este trabajo.

El modelo de Hodgkin Huxley (1952) se sigue utilizando en la actualidad sin cambios significativos, muchos modelos han aparecido y seguirán apareciendo, pero el modelo de Hodgkin Huxley es la mejor aproximación que se tiene del proceso biológico.

Las neurociencias computacionales son un área que hasta ahora está comenzando, falta mucho por hacer. A medida que transcurre el tiempo y progresan las investigaciones aparecerán problemas mucho más específicos. Este trabajo deja muchas incógnitas abiertas que pueden dar paso a otros proyectos interesantes. Los programas hechos en el lenguaje C quedan abiertos al público. Alguien que desee continuar con este trabajo ahorraría mucho tiempo de programación, del cual se puede disponer para hacer un análisis más extenso.

Apéndice A

Programas C.

A.1 Neurona Hodgkin Huxley

A.1.1 Entradas sinápticas

Los programas a continuación sirven para generar una corriente $I_{ext} = I_s + I_p$ con su respectiva modulación temporal en el intervalo entre disparos. Genera una entrada sináptica con estas condiciones y escribe los datos en un archivo “in” y otro archivo “in.in”. El archivo “in” también guarda el tiempo máximo de integración y el intervalo utilizado para la integración. El archivo “in.in” tiene en una columna el tiempo y en otra la corriente. Los tiempos en que se presentan los disparos son guardados en un archivo “maximos” y un archivo “const” guarda la corriente I_s . Todos los programas hacen referencia a la función alfa que se describe también en este apéndice.

Entrada sináptica de M picos espaciados T mseg.

```
// Genera M picos con un periodo de T para un tiempo Tmax
// También genera archivos con los tiempos en que ocurren los máximos.

#include <stdio.h>
#include <math.h>

#define Va 30
#define Vsyn -50
#define tao 2
#define gsyn 0.5
// #define T 20
// #define M 5

#define AMP 30
#define MIN 0
```

```

double alfa( double );

main()
{
    double t,dt=0.01,tmax,tant=0,tant2=0,
           Is=0,Ip;
    int m=0,M,T,n=0,nm=0;

    FILE *ent,*ent1,*Ui,*fase,*ent2;

    ent=fopen("in.in","w");
    ent1=fopen("in","w");
    Ui=fopen("ui","w");
    fase=fopen("fase_ent","w");
    ent2=fopen("const","w");

    printf("Tmax=");
    scanf("%lf",&tmax);

    printf("M=");
    scanf("%d",&M);

    printf("T=");
    scanf("%d",&T);

    fprintf(ent1,"%lf\n%lf\n",dt,tmax);
    fprintf(ent2,"%lf\n%lf\n",dt,tmax);

    for(t=0;t<tmax;t+=dt)
    {
        Ip=0;
        for(m=0;m<M;m++)
            Ip+=gsyn*alfa(t-m*T)*(Va-Vsyn);
        if(n%(100*T)==0)
        {
            fprintf(fase,"%lf\t%lf\n",tant-tant2,t-tant);

            if(nm<M)fprintf(Ui,"%lf\t%lf\n",t,Va);
            tant2=tant;
            tant=t;
            nm++;
        }

        fprintf(Ui,"%lf\t%lf\n",t,M*Ip);
        fprintf(ent2,"%lf\n",Is);
        fprintf(ent,"%lf\t%lf\n",t,Is+Ip);
        fprintf(ent1,"%lf\n",Is+Ip);
        Is++;
    }
}

```

Entrada con modulación senoidal

Crea un archivo “sin” y “sin.ent” que remplazan los archivos “in” y “in.in” descritos arriba. En el archivo está la corriente generada por un tren de pulsos de la forma descrita en la sección 5.2 para una modulación senoidal.

//entrada con modulación temporal senoidal, “ui” tiene los tiempos en que ocurren los máximos.

```

#include <stdio.h>
#include <math.h>

#define Va 30
#define Vsyn -50
#define tao 2
#define gsyn 0.5

```

```

#define Tp 100
#define d0 20
#define d1 10
#define AMP 5
#define MIN 0

float alfa( float );

main()
{
    float t,dt=0.01,tmax=5000,tin=0,tina=0,tinaa=0,
        Is=0,Ip;
    int n=0;

    FILE *ent,*ent1,*Ui,*fase;

    ent=fopen("sin.ent","w");
    ent1=fopen("sin","w");
    Ui=fopen("ui","w");
    fase=fopen("fase_ent","w");

    fprintf(ent1,"%lf\n%lf\n",dt,tmax);

    for(t=0;t<tmax;t+=dt)
    {
        if(t>tin)
        {
            tinaa=tina;
            tina=tin;
            tin=d0+d1*sin(2*M_PI*t/Tp);

            if(n>10)
                fprintf(fase,"%lf\t%lf\n",tin-tina,tina-tinaa);
            fprintf(Ui,"%lf\t%d\n",t-dt,AMP);
            n++;
        }

        fprintf(Ui,"%lf\t%d\n",t,MIN);

        Ip=gsyn*alfa(t-tina)*(Va-Vsyn);

        fprintf(ent,"%f\t%f\n",t,Is+Ip);
        fprintf(ent1,"%f\n",Is+Ip);
    }
}

```

Modulación Caótica

Utiliza el método de integración numérica de Runge-Kutta de cuarto orden, devuelve archivos similares a los de las otras entradas con la corriente de entrada modulada de forma caótica. Los modelos caóticos y los parámetros utilizados están descritos en la sección 5.2.

Modelo Rossler

```

// Genera una entrada modulada en tiempo caóticamente por un atractor de Rossler

#include <stdio.h>
#include <math.h>

#define Va 30
#define Vsyn -50

```

```

#define tao 2
#define gsyn 0.5
#define d0 20
#define d1 10
#define a 0.36
#define b 0.4
#define c 4.5
#define p 0.1

float alfa( float );
float derx(float ,float );
float dery(float ,float );
float derz(float ,float );

main()
{
    float t,dt=0.01,dt2,tmax=2000,tin=0,tina=0,
        Is=25,Ip,
        x=0,y=0,z=0.1,
        kx[4],ky[4],kz[4];
    int n=0;

    FILE *ent,*ent1;

    ent=fopen("ross.ent","w");
    ent1=fopen("ross","w");

    dt2=p*dt;

    fprintf(ent1,"%lf\n%lf\n",dt,tmax);

    for(t=0;t<tmax;t+=dt)
    {
        if(t>tin)
        {
            tina=tin;
            tin+=d0+(d1/10)*x;
        }

        kx[0] = dt2 * derx(y,z);
        ky[0] = dt2 * dery(x,y);
        kz[0] = dt2 * derz(x,z);

        kx[1] = dt2 * derx(y+ky[0]/2,z+kz[0]/2);
        ky[1] = dt2 * dery(x+kx[0]/2,y+ky[0]/2);
        kz[1] = dt2 * derz(x+kx[0]/2,z+kz[0]/2);

        kx[2] = dt2 * derx(y+ky[1]/2,z+kz[1]/2);
        ky[2] = dt2 * dery(x+kx[1]/2,y+ky[1]/2);
        kz[2] = dt2 * derz(x+kx[1]/2,z+kz[1]/2);

        kx[3] = dt2 * derx(y+ky[2],z+kz[2]);
        ky[3] = dt2 * dery(x+kx[2],y+ky[2]);
        kz[3] = dt2 * derz(x+kx[2],z+kz[2]);

        x+=kx[0]/6 + kx[1]/3 + kx[2]/3 + kx[3]/6;
        y+=ky[0]/6 + ky[1]/3 + ky[2]/3 + ky[3]/6;
        z+=kz[0]/6 + kz[1]/3 + kz[2]/3 + kz[3]/6;

        Ip=gsyn*alfa(t-tina)*(Va-Vsyn);

        fprintf(ent,"%f\t%f\n",t,Is+Ip);
        fprintf(ent1,"%f\n",Is+Ip);

        n++;
    }
}

/*****

```

derivada utilizadas para la integración numérica con Runge-Kutta para las variables x,y,z del modelo de Rossler


```
float derx(float y,float z)
{
    return -y-z;
}
float dery(float x,float y)
{
    return x*a*y;
}
float derz(float x,float z)
{
    return b*x-c*z+x*z;
}
```

Modelo Lorenz //Genera entradas con modulación caótica en tiempo utilizando el atractor de lorenz

```
#include <stdio.h>
#include <math.h>

#define Va 30
#define Vsyn -50
#define tao 2
#define gsyn 0.5
#define d0 20
#define d1 20
#define d 10
#define e 28
#define f 8/3
#define p 0.01
#define AMP 5
#define MIN 0

double alfa( double );
double derx(double ,double );
double dery(double ,double ,double);
double derz(double ,double ,double);

main()
{
    double t,dt=0.01,dt2,tmax=300,tin=0,tina=0,tinaa=0,
           Is=0,Ip,
           x=0,y=0,z=0.1,
           kx[4],ky[4],kz[4];
    int n=0;

    FILE *ent,*ent1,*Ui,*fase;

    ent=fopen("lor.ent","w");
    ent1=fopen("lor","w");
    Ui=fopen("ui","w");
    fase=fopen("fase_ent","w");

    dt2=p*dt;

    fprintf(ent1,"%lf\n%lf\n",dt,tmax);

    for(t=0;t<tmax;t+=dt)
    {
        if(t>tin)
        {
            tinaa=tina;
            tina=tin;
            tin+=d0+(d1/25)*(z-25);
            if(n>10)
                fprintf(fase,"%lf\t%lf\n",tin-tina,tina-tinaa);
            fprintf(Ui,"%lf\t%d\n",t-dt,AMP);
            n++;
        }
    }
}
```

```

    }

    fprintf(Ui, "%lf\t%d\n", t, MIN);

    kx[0] = dt2 * derx(x, y);
    ky[0] = dt2 * dery(x, y, z);
    kz[0] = dt2 * derz(x, y, z);

    kx[1] = dt2 * derx(x+kx[0]/2, y+ky[0]/2);
    ky[1] = dt2 * dery(x+kx[0]/2, y+ky[0]/2, z+kz[0]/2);
    kz[1] = dt2 * derz(x+kx[0]/2, y+ky[0]/2, z+kz[0]/2);

    kx[2] = dt2 * derx(x+kx[1]/2, y+ky[1]/2);
    ky[2] = dt2 * dery(x+kx[1]/2, y+ky[1]/2, z+kz[1]/2);
    kz[2] = dt2 * derz(x+kx[1]/2, y+ky[1]/2, z+kz[1]/2);

    kx[3] = dt2 * derx(x+kx[2], y+ky[2]);
    ky[3] = dt2 * dery(x+kx[2], y+ky[2], z+kz[2]);
    kz[3] = dt2 * derz(x+kx[2], y+ky[2], z+kz[2]);

    x=(kx[0]/6 + kx[1]/3 + kx[2]/3 + kx[3]/6);
    y=(ky[0]/6 + ky[1]/3 + ky[2]/3 + ky[3]/6);
    z=(kz[0]/6 + kz[1]/3 + kz[2]/3 + kz[3]/6);

    Ip=gsyn*alfa(t-tina)*(Va-Vsyn);

    fprintf(ent, "%lf\t%lf\n", t, Is+Ip);
    fprintf(ent1, "%lf\n", Is+Ip);

}
}
/*****
Derivada de las variables x,y,z utilizadas en el método de Runge Kutta
*****/

double derx(double x, double y)
{
    return d*(y-x);
}
double dery(double x, double y, double z)
{
    return e*x-y-x*z;
}
double derz(double x, double y, double z)
{
    return -f*z+x*y;
}

```

A.1.2 Función sináptica.

Esta es la función sináptica que describe como la corriente pasa de una neurona a otra. Todas las corrientes I_p tienen esta forma. Solo recibe el tiempo donde se presentan los disparos y devuelve la corriente I_p .

```

double alfa(double t)
{
    if(t>0)
        return (t/tao)*exp(-t/tao);
    else
        return 0;
}

```

A.1.3 Runge-Kutta de cuarto orden para una neurona HH.

Este programa recibe el nombre del archivo con una entrada de corriente, el archivo debe incluir el tiempo que dura la entrada y el paso de integración que se utilizó. También recibe el nombre que se desea tener en el archivo de salida, el cual es generado por integración numérica de cuarto orden con Runge - Kutta del modelo de Hodgkin Huxley.

```
//Runge Kutta de cuarto orden para el modelo de Hodgkin y Huxley recibe los nombre de los archivos de entrada y de salida.

#include <stdio.h>
#include <math.h>

#define C 1 //e-6
#define gna 120 //e-3
#define gk 36 //e-3
#define gl 0.3 //e-3
#define Vna 50 //e-3
#define Vk -77 //e-3
#define Vl -54.5 //e-3

double der( double , double , double );
double derv( double , double , double , double , double );

main()
{
    double V,
        kv[4],km[4],kn[4],kh[4],
        am,an,ah,
        bm,bn,bh,
        m,n,h,
        t,dt,tmax,
        i;
    char archivo[10],archivo2[10];
    FILE *volt,*entrada;

//abrir archivos

    printf("\n\n\t\tArchivo de entradas?\n");
    scanf("%s",&archivo);
    printf("\n\n\t\tarchivo de salida?\n");
    scanf("%s",&archivo2);

    volt=fopen(archivo2,"w");
    entrada=fopen(archivo,"r");

//inicializar

    V=-65;
    t=0;
    m=0.0526;
    n=0.313;
    h=0.6;

    fscanf(entrada,"%lf",&dt);
    fscanf(entrada,"%lf",&tmax);
//rutina

    for(t=0;t<tmax;t+=dt) //durante el tiempo de simulación
    {

//calculo a terminos del modelo HH
        am= (0.1) * (V + 40) / ( 1 - exp( - ( V + 40) / 10 ) );
        an= (0.01) * (V + 55) / ( 1 - exp( - ( V + 55) / 10 ) );
        ah= (0.07) * exp( - ( V + 65 ) / 20);
```

```
//calculo b terminos del modelo HH
bm= 4 * exp( - ( V + 65 ) / 18 );
bn= 0.125 * exp( - ( V + 65 ) / 80 );
bh= 1 / ( 1 + exp( - ( V + 35 ) / 10));

//calculo mhn con runge kutta
km[0] = dt * der(am,bm,m);
kh[0] = dt * der(ah,bh,h);
kn[0] = dt * der(an,bn,n);

km[1] = dt * der(am,bm,m + km[0]/2);
kh[1] = dt * der(ah,bh,h + kh[0]/2);
kn[1] = dt * der(an,bn,n + kn[0]/2);

km[2] = dt * der(am,bm,m + km[1]/2);
kh[2] = dt * der(ah,bh,h + kh[1]/2);
kn[2] = dt * der(an,bn,n + kn[1]/2);

km[3] = dt * der(am,bm,m + km[2]);
kh[3] = dt * der(ah,bh,h + kh[2]);
kn[3] = dt * der(an,bn,n + kn[2]);

m+=km[0]/6 + km[1]/3 + km[2]/3 + km[3]/6;
h+=kh[0]/6 + kh[1]/3 + kh[2]/3 + kh[3]/6;
n+=kn[0]/6 + kn[1]/3 + kn[2]/3 + kn[3]/6;

//entrada
fscanf(entrada,"%lf",&I);

//calculo V
kv[0]=dt*derv(m,h,n,V,I);
kv[1]=dt*derv(m,h,n,V+kv[0]/2,I);
kv[2]=dt*derv(m,h,n,V+kv[1]/2,I);
kv[3]=dt*derv(m,h,n,V+kv[2],I);

V+=kv[0]/6 + kv[1]/3 + kv[2]/3 + kv[3]/6;

//almacenó
fprintf(volt,"%f\t%f\n",t,V);
}
}
```

double der(double a, double b, double n)

Se utiliza en el método de Runge- Kutta es la derivada de las funciones n,m, y h del modelo de Hodgkin -Huxley.

```
.....
der es la derivada de las funciones m,n y h,todas tienen derivadas muy parecidas. se utiliza en el método de Runge Kutta
...../

double der(double a, double b, double m)
{
    return ( -( a + b ) * m + a );
}
}
```

double derv(double n, double h, double n, double V, double I)

Se utiliza para la integración numérica. es la derivada del voltaje en el modelo de Hodgkin y Huxley.

```
.....
derv es la derivada del voltaje a través de la membrana se utiliza en el método de Runge - Kutta
...../
```

```

double deriv(double m, double h, double n, double V,double I)
{
    return (1/C) * (- gna * pow(m,3) * h * (V-Vna) -
    gk * pow(n,4) * (V - Vk) - gl * (V - Vl) + I);
}

```

Calcular máximos.

Este programa lee el archivo donde se encuentran los voltajes en la neurona, y escribe un archivo con los máximos, y otro con las diferencias entre los máximos.

```

/*****
devuelve archivos con la diferencia entre los tiempos en que se encuentran los máximos.para acople de neuronas.
*****/

/*****
librerías
*****/

#include <stdio.h>

/*****
parámetros
*****/

#define AMP 30
#define MIN 0

/*****
programa max.c
*****/

main()
{

    FILE *max,*volt,*dif,*Uo;
    double t1=0,t2,v1,v2,v21,v22,d=0;
    int n=0;
    char archivo[10];

    max=fopen("maximos","w");
    dif=fopen("dif","w");
    volt=fopen("out","r");
    Uo=fopen("uo","w");

    fprintf(Uo,"%lf\t%lf\n",t1,MIN);

    fscanf(volt,"%lf",&t1);
    fscanf(volt,"%lf",&v1);
    fscanf(volt,"%lf",&v21);
    fscanf(volt,"%lf",&t2);
    fscanf(volt,"%lf",&v2);
    fscanf(volt,"%lf",&v22);

    fprintf(Uo,"%lf\t%lf\n",t2,MIN);

    while(!feof(volt))
    {
        v1=v2;

        fscanf(volt,"%lf",&t2);
        fscanf(volt,"%lf",&v2);
        fscanf(volt,"%lf",&v22);

        if(v2>v1) /***** si el voltaje es creciente*****/
        {
            if(v2>0&&v1<0) /***** si acaba de pasar por v=0 *****/
            {

```

```

        fprintf(max, "%f\t%f\n", t1, t2); /*****almaceno valor *****/

        //if (n>2)
            fprintf(dif, "%f\t%f\n", t2-t1, d);
            d=t2-t1;
            t1=t2;
            fi++;
            fprintf(Uo, "%lf\t%d\n", t2, AMP);
        }
        else
            fprintf(Uo, "%lf\t%d\n", t2, MIN);
    }
    else
        fprintf(Uo, "%lf\t%d\n", t2, MIN);

}

fclose(max);
fclose(dif);
fclose(volt);
fclose(Uo);
}

```

A.2 Neurona IF

Para las simulaciones de una neurona IF se utilizó programación por objetos.

Entrada Constante

La función constante(double tmax, double Is) recibe el tiempo máximo de simulación y el valor de la corriente, a partir de estos valores genera un archivo con los valores de la corriente constante, durante el tiempo de simulación, se utiliza un paso de $dt=0.01$.

```

// Programa que genera una entrada constante, recibe Is

/*****
LIBRERIAS
*****/

#include <stdio.h>
#include <math.h>

constante(double tmax, double Is)
{
    double t, dt=0.01, tin=0;

    FILE *ent, *enti;

    ent=fopen("in.in", "w");
    enti=fopen("in", "w");

    fprintf(enti, "%f\n%f\n", dt, tmax);

    // printf("\tIs=?\t");
    // scanf("%f", &Is);

    for(t=0; t<tmax; t+=dt)
        fprintf(enti, "%f\n", Is);

    fclose(ent);
}

```

```

        fclose(ent1);
    }

```

Entrada periódica

```

/*****
    LIBRERIAS
    *****/

#include <stdio.h>
#include <math.h>

/*****
    //PARAMETROS
    *****/

#define Va 30
#define Vsyn -50
#define tao 2
#define gsyn 0.5

double alfa(double);
void entrada(double,double);

/*****
    entrada genera una entrada periodica con periodo T, tmax es el tiempo de simulación
    *****/

void entrada(double tmax,double T)
{
    double t,dt=0.01,
           Is=25,Ip;
    int m;
    FILE *ent,*ent1;

    //ABRIR ARCHIVOS

    ent=fopen("in.in","w");
    ent1=fopen("in","w");

    //    printf(" número de eventos(TMAX) = ");    //recibe el tiempo maximo de simulacion

    //    scanf("%lf",&tmax);
    //    printf("tmax = %f\n",tmax);

    fprintf(ent1,"%lf\n%lf\n",dt,tmax);    //escribe en el archivo de entradas dt y tiempo maximo de simulacion

    for(t=0;t<tmax;t+=dt)
    {
        Ip=0;
        for(m=0;m<T<tmax;m++)
            Ip+=gsyn*alfa(t-m*T)*(Va-Vsyn);

        fprintf(ent,"%lf\t%lf\n",t,Is+Ip);
        fprintf(ent1,"%lf\n",Is+Ip);
    }
    fclose(ent);
    fclose(ent1);

    return;
}

```

IF(double C,double tao)

Está función integra numéricamente el modelo para una neurona IF descrita en el capítulo 6. Recibe el valor de la capacitancia y el valor del tiempo de carga asociado a la neurona.

```

/*****
LIBRERIAS
*****/

#include <stdio.h>

/*****
PARAMETROS
*****/

#define C 15 //e-6
#define tao 20
#define gna 120 //e-3
#define gk 36 //e-3
#define gl 0.3 //e-3

#define Vth -55
#define Vr -75 //e-3
#define Vd -10
#define Vl 75
#define Vk -77 //e-3
#define V1 -54.5 //e-3

/*****

double derv( double , double , double , double );
double derv2(double ,double ,double );

IF(double C,double tao)
{
    double V,k[4],t,dt,tmax,I,delta=0.01;
    char archivo[10]="in",archivo2[10]="outif";
    FILE *volt,*ent,*uo;

//abrir archivos

    //scanf("%s",&archivo);
    //scanf("%s",&archivo2);
    volt=fopen(archivo2,"w");
    ent=fopen(archivo,"r");
    uo=fopen("uofif","w");

//pide parametros
/*****
    printf("Escriba C=?\t"); //pide la capacitancia

    scanf("%lf",&C);

    printf("Escriba tao=?\t"); //pide el tiempo de carga
    scanf("%lf",&tao);
*****/

//inicializar

    V=Vr;
    t=0;

    fscanf(ent,"%lf",&dt); //lee dt y tiempo maximo de simulacion
    fscanf(ent,"%lf",&tmax);

//    printf("%lf\n",dt);
//    printf("%d",g/C);

//rutina

```

```

        for(t=0;t<tmax;t+=dt)
        {
//entrada
        fscanf(ent,"%lf",&I);

        if(V>Vth)
        {
            V+=Vi;
            //if(t>100)
            fprintf(volt,"%f\t%f\n",t,V);
            fprintf(uo,"%f\t%d\n",t,5);

            while(V-Vr-Vd>delta)
            {

                k[0]=dt*deriv2(V,I,tao);
                k[1]=dt*deriv2(V+k[0]/2,I,tao);
                k[2]=dt*deriv2(V+k[1]/2,I,tao);
                k[3]=dt*deriv2(V+k[2],I,tao);

                V+=k[0]/6 + k[1]/3 + k[2]/3 + k[3]/6;
                //if(t>100)
                fprintf(volt,"%f\t%f\n",t,V);
                fprintf(uo,"%f\t%d\n",t,0);
                //printf("%f\n",V);
            }
        }
//calculo V

        k[0]=dt*deriv(V,I,C,tao);
        k[1]=dt*deriv(V+k[0]/2,I,C,tao);
        k[2]=dt*deriv(V+k[1]/2,I,C,tao);
        k[3]=dt*deriv(V+k[2],I,C,tao);

        V+=k[0]/6 + k[1]/3 + k[2]/3 + k[3]/6;

//almaceno
        //if(t>100)
        fprintf(volt,"%f\t%f\n",t,V);
        fprintf(uo,"%f\t%d\n",t,0);

    }

}

/*****
Derivadas asociadas al modelo de IF utilizadas por el método de Runge Kutta.
*****/

double deriv(double V,double I,double C,double tao)
{
    return (I/C)-(1/tao)*(V-Vr);
}

double deriv2(double V,double I,double tao)
{
    double tr;
    tr=tao/100;
    return -(1/tr)*(V-Vr-Vd);
}

```

double maxif()

Lee el archivo "outif" que genera la función IF(double,double) y genera un archivo con los tiempos donde se obtienen los disparos.

```

#include <stdio.h>

double maxif()

```

```

{
    FILE *max,*volt,*dif;
    double t,t2,v,v2,d=0;
    int n=0;
    char arch[10];

    max=fopen("maxif.txt","w");
    dif=fopen("difif.txt","w");

//    scanf("%s",&archivo);

//    volt=fopen(archivo2,"w");

    volt=fopen("outif","x");

    fscanf(volt,"%lf\t%lf\n",&t2,&v);
//fscanf(volt,"%lf\t%lf\n",&t2,&v2);

    while(!feof(volt))
    {
        if(v>19)
        {
            if(n>0)
            {
                fprintf(max,"%f\t%f\n",t,t2);
                fprintf(dif,"%f\t%f\n",t2-t,d);
//                printf("\n\t%f",d);
                d=t2-t;
            }

            t=t2;
            n++;
        }

        fscanf(volt,"%lf\t%lf\t",&t2,&v);
//printf("%f\t%f\n",t2,v);
    }
    fclose(volt);
    return d;
}

```

Simulación de una neurona IF

Este programa simula una neurona IF con una entrada constante de $I_s = 25$. Los resultados de la simulación están en los archivos “outif” y “maxif”. Imprime en pantalla el periodo de la salida.

```

#include <stdio.h>
#include <math.h>

extern constante(double,double);
extern IF(double,double);
extern double maxif();

#define TMAX 300

main()
{
    double C,tao,Is,T,d=1,arriba,abajo,dif;
    FILE *periodo;

    printf("Cuanto quiere que sea C????\n");
    scanf("%lf",&C);

    printf("Cuanto quiere que sea TA00000????\n");

```

```

scanf("%lf",&tao);

Is=25;

constante(TMAX,Is);
IF(C,tao);
d=maxif();
printf("%lf",d);
}

```

void hh(void)

Simula una neurona HH con el método de Runge Kutta, lee la corriente de entrada de un archivo "in" y escribe la salida en un archivo "outhh".

```

//Runge Kutta de cuarto orden para el modelo de hodkin y Huxley
//todavia sin apuntadores reproduce todo lo del primer artículo
//recibe los nombre de los archivos de entrada y de salida.

/*****
LIBRERIAS
*****/

#include <stdio.h>
#include <math.h>

/*****
PARAMETROS
*****/

#define C 1 //e-6
#define gna 120 //e-3
#define gk 36 //e-3
#define gl 0.3 //e-3
#define Vna 50 //e-3
#define Vk -77 //e-3
#define V1 -54.5 //e-3

/*****
DEFINICION DE FUNCIONES
*****/

double der( double , double , double );
double derv( double , double , double , double , double );
void hh(void)
{
    double V,
    kv[4],km[4],kn[4],kh[4],
    am,an,ah,
    bm,bn,bh,
    m,R,h,
    t,dt,tmax,
    I;
    char archivo[10]="in",archivo2[10]="outhh";
    FILE *volt,*entrada;
//abrir archivos
/*****
printf("\n\n\t\tArchivo de entradas?\n");
scanf("%s",&archivo);
printf("\n\n\t\tarchivo de salida?\n");
scanf("%s",&archivo2);
*****/
volt=fopen(archivo2,"w");
entrada=fopen(archivo,"r");
//inicializar
V=-65;
t=0;
m=0.0526;
ñ=0.313;
h=0.6;

```

```

        fscanf(entrada,"%lf",&dt);
        fscanf(entrada,"%lf",&tmax);
//rutina
for(t=0;t<tmax;t+=dt)
{
//calcula ais
am= (0.1) * (V + 40) / ( 1 - exp( - ( V + 40) / 10 ) );
an= (0.01) * (V + 55) / ( 1 - exp( - ( V + 55) / 10 ) );
ah= (0.07) * exp( - ( V + 65 ) / 20);
//calcula bis
bm= 4 * exp( - ( V + 65 ) / 18 );
bn= 0.125 * exp( - ( V + 65 ) / 80 );
bh= 1 / ( 1 + exp( - ( V + 35 ) / 10));
//calcula mhn
km[0] = dt * der(am,bm,m);
kh[0] = dt * der(ah,bh,h);
kn[0] = dt * der(an,bn,n);
km[1] = dt * der(am,bm,m + km[0]/2);
kh[1] = dt * der(ah,bh,h + kh[0]/2);
kn[1] = dt * der(an,bn,n + kn[0]/2);
km[2] = dt * der(am,bm,m + km[1]/2);
kh[2] = dt * der(ah,bh,h + kh[1]/2);
kn[2] = dt * der(an,bn,n + kn[1]/2);
km[3] = dt * der(am,bm,m + km[2]);
kh[3] = dt * der(ah,bh,h + kh[2]);
kn[3] = dt * der(an,bn,n + kn[2]);
m+=km[0]/6 + km[1]/3 + km[2]/3 + km[3]/6;
h+=kh[0]/6 + kh[1]/3 + kh[2]/3 + kh[3]/6;
n+=kn[0]/6 + kn[1]/3 + kn[2]/3 + kn[3]/6;
//entrada
        fscanf(entrada,"%lf",&I);
//calcula V
kv[0]=dt*der(m,h,n,V,I);
kv[1]=dt*der(m,h,n,V+kv[0]/2,I);
kv[2]=dt*der(m,h,n,V+kv[1]/2,I);
kv[3]=dt*der(m,h,n,V+kv[2],I);
V+=kv[0]/6 + kv[1]/3 + kv[2]/3 + kv[3]/6;
//almaceno
        fprintf(volt,"%f\t%f\n",t,V);
}
        fclose(volt);
}

```

void max()

Lee el archivo "outhh" resultado de la simulación de una neurona HH y genera un archivo con los tiempos de disparo.

```

//devuelve archivos con la diferencia entre los tiempos en que se encuentran los máximos.
#include <stdio.h>
#define AMP 30
#define MIN 0
double max()
{
        FILE *max,*volt,*dif,*Uo;
        double t1=0,t2,v1,v2,d=0;
        int n=0;
        char archivo[10];
        max=fopen("maxhh","w");
        dif=fopen("difhh","w");
        volt=fopen("outhh","r");
        Uo=fopen("uohh","w");
        fprintf(Uo,"%lf\t%lf\n",t1,MIN);
        fscanf(volt,"%lf",&t1);
        fscanf(volt,"%lf",&v1);
        fscanf(volt,"%lf",&t2);
        fscanf(volt,"%lf",&v2);
        fprintf(Uo,"%lf\t%lf\n",t2,MIN);

        while(!feof(volt))

```

```

{
    v1=v2;

    fscanf(volt,"%lf",&t2);
    fscanf(volt,"%lf",&v2);
    if(v2>v1)
    {
        if(v2>=0&&v1<0)
        {

            fprintf(max,"%f\t%f\n",t1,t2);

            //if(n>2)
                fprintf(dif,"%f\t%f\n",t2-t1,d);
            d=t2-t1;
            t1=t2;
            ñ++;
            fprintf(Uo,"%f\t%d\n",t2,AMP);
        }
        else
            fprintf(Uo,"%f\t%d\n",t2,MIN);
    }
    else
        fprintf(Uo,"%f\t%d\n",t2,MIN);
}
fclose(max);
fclose(dif);
fclose(volt);
fclose(Uo);
return(d);
}

```

Programa para simular una neurona HH.

Este programa sirve para hacer una simulación sencilla de una neurona HH ya sea con entrada constante o con una entrada periodica.

```

#include <stdio.h>
extern entrada(double, double);
extern constante(double,double);
//extern if(double,double);
extern max();
//extern maxif();
extern hh();
#define TMAX 300
main()
{
    double T;
    // printf("Por favor inserte el periodo(mseg)?");
    printf("Por favor inserte Is(microamp)?");
    scanf("%lf",&T);
    constante(TMAX,T);
    // entrada(TMAX,T);
    hh();
    max();
}

```

Calculo de periodos

Esta función calcula los periodos de la salida de una neurona HH a diferentes valores de una entrada de corriente constante. Escribe los periodos en un archivo "periodos"

```

#include <stdio.h>
extern entrada(double, double);
extern constante(double,double);

```

```

extern double max();
extern hh();
#define TMAX 300
main()
{
    double Is,dif;
    FILE *periodo;
    periodo=fopen("periodos","w");
    for(Is=10;Is<35;Is+=5)
    {
        constante(TMAX,Is);
        hh();
        dif=max();
        fprintf(periodo,"%f\t%f\n",Is,dif);
    }
    fclose(periodo);
}

```

Calculo del periodo de oscilación variando τ

Este programa escribe en un archivo "taovsT" como varia el período de oscilación de una neurona IF cuando se varia τ dejando C constante. Recibe el valor de corriente para una entrada constante.

```

#include <stdio.h>
#include <math.h>

extern constante(double,double);
extern IF(double,double);
extern double maxif();

#define TMAX 300

main()
{
    double C=4,tao,Is,T,d,arriba,abajo,dif;
    FILE *periodo;

    periodo=fopen("taovsT","w");

    /*****
    printf("Cuanto quiere que sea C????\n");
    scanf("%lf",&C);
    *****/

    printf("Cuanto quiere que sea Is????\n");
    scanf("%lf",&Is);

    constante(TMAX,Is);

    for(tao=8;tao<25;tao+=0.1)
    {
        IF(C,tao);
        d=maxif();
        fprintf(periodo,"%lf\t%lf\n",tao,d);
        printf("tao %lf\tT= %lf\n",tao,d);
    }
    fclose(periodo);
}

```

Ajuste de τ para reproducir la salida de una neurona HH.

Este programa lee el archivo “periodos” que tiene el valor de una entrada de corriente constante y el período de oscilación respectivo de una neurona HH. Dejando constante la capacitancia de la neurona, utiliza una búsqueda binaria para encontrar el tiempo τ que reproduce esta salida. Con una sencilla modificación sirve para ajustar la capacitancia de la neurona para un τ y una entrada específica.

```

#include <stdio.h>
#include <math.h>

extern constante(double,double);
extern IF(double,double);
extern double maxif();

#define TMAX 300

main()
{
    double C,tao,Is,T,d=1,arriba=30,abajo=0,dif;
    FILE *periodo,*tabla;

    periodo=fopen("periodos","r");
    tabla=fopen("tabla","w");

    printf("Cuanto quiere que sea C????\n");
    scanf("%lf",&C);
    /******
    printf("arriba?");
    scanf("%lf",&arriba);
    printf("abajo?");
    scanf("%lf",&abajo);
    *****/
    /******
    printf("Cuanto quiere que sea TA00000????\n");
    scanf("%lf",&tao);
    *****/

    while(!feof(periodo))
    {
        arriba=30;
        abajo=0;

        fscanf(periodo,"%lf",&Is);
        constante(TMAX,Is);
        fscanf(periodo,"%lf",&T);

        tao=(arriba+abajo)/2;

        printf("\nIs=%lf\tT=%lf",Is,T);

        dif=d-T;
        while(fabs(dif)>(0.02))
        {
            IF(C,tao);
            d=maxif();
            if(d>T||d==0)
                abajo=tao;
            else
                arriba=tao;

            tao=(arriba+abajo)/2;
            printf("\n\ttao=%lf\t%d=%lf",tao,d);
            dif=d-T;
        }
    }
}

```

```

        fprintf(tabla,"%lf\t%lf\t%lf\n",Is,T,tao);
    }
    fclose(periodo);
}

```

Calculo de períodos cambiando la capacitancia

Este programa calcula el período de oscilación de una neurona IF para una corriente específica y un τ específico, para diferentes valores de capacitancias.

```

#include <stdio.h>
#include <math.h>

extern constante(double,double);
extern IF(double,double);
extern double maxif();

#define TMAX 300

main()
{
    double C,tao=10,Is,T,d,arriba,abajo,dif;
    FILE *periodo;

    periodo=fopen("CvsTtao10","w");

    /*****
    printf("Cuanto quiere que sea C????\n");
    scanf("%lf",&C);
    *****/

    printf("Cuanto quiere que sea Is????\n");
    scanf("%lf",&Is);

    constante(TMAX,Is);

    for(C=0.1;C<12;C+=0.1)
    {
        IF(C,tao);
        d=maxif();
        fprintf(periodo,"%lf\t%lf\n",C,d);
        printf("C %lf\tT= %lf\n",C,d);
    }
    fclose(periodo);
}

```

A.3 Acople de Neuronas

A.3.1 Acople dos Neuronas

Este programa es una integración numérica de cuarto orden con el método de Runge-Kutta. Utiliza la función `buscar_maximos` que se describe a continuación.

```

/*****
ES una simulación para 2 neuronas HH acopladas sinápticamente, necesita un archivo "in" con las corrientes de entrada, y genera
un archivo con el voltaje en la membrana y otro con los tiempos de disparo.
*****/

/*****

```

```

        librerías
        *****/

#include <stdio.h>
#include <math.h>

/*****
        parámetros
        *****/

#define C 1 //e-6
#define gna 120 //e-3
#define gk 36 //e-3
#define gl 0.3 //e-3
#define Vna 50 //e-3
#define Vk -77 //e-3
#define Vl -54.5 //e-3
#define td 10
#define tao 2
#define Cjk 40

/*****
        FUNCIONES
        *****/

double alfa( double );
double der( double , double , double );
double deriv( double , double , double , double , double );
int buscar_maximos(double *,double *,double , double *, double *, double*,double);
void hh(double *,double ,double ,double *,double *, double *,double);
double corriente(double ,double [500],int );

/*****
        PROGRAMA PRINCIPAL
        *****/

main()
{
    double V[2],Vant[2],
           t,dt,tmax,to[2][500],
           I,Iint[2],m[2],n[2],h[2];

    int i=0,j=0;
    char archivo[10]="in",archivo2[10]="out";
    FILE *volt,*entrada,*realimentacion,*tiempo,*tiempo2;

/*****   archivos   *****/

    volt=fopen(archivo2,"w");
    tiempo=fopen("tiempos","w");
    tiempo2=fopen("tiempos2","w");
    entrada=fopen(archivo,"r");
    realimentacion=fopen("maxout","w");

/*****   inicializacion   *****/

    to[0][0]=0;
    to[1][0]=0;

    Iint[0]=0;
    Iint[1]=0;

    V[0]=-65.025499;
    Vant[0]=-65.025499;
    fi[0]=0.317286;
    m[0]=0.052774;
    h[0]=0.597012;

    V[1]=-65.025499;
    Vant[1]=-65.025499;
    fi[1]=0.317286;

```

```

n[1]=0.052774;
h[1]=0.597012;

fscanf(entrada,"%lf",&dt);
fscanf(entrada,"%lf",&tmax);

/***** simulacion *****/

for(t=0;t<tmax;t+=dt)
{

    fscanf(entrada,"%lf",&I);

    if(j>0)
        Iint[0]=corriente(t,to[1],j);

    if(i>0)
        Iint[1]=corriente(t,to[0],i);
/*****
i y j llevan la cuenta del número de máximos en las respectivas neuronas. Buscar máximos de vuelve un 1 cuando se presenta un
disparo, to[n][i] es el
i-esimo tiempo de disparo de la neurona n.
*****/

    if(buscar_maximos(&Vant[0],&V[0],I+Iint[0],&n[0],&m[0],&h[0],dt))
    {
        to[0][i]=t;
        i++;
        printf("\t\t%lf\n",to[0][i-1]-to[0][i-2]);
//
        fprintf(tiempo,"%lf\t%lf\n",t,to[0][i-1]-to[0][i-2]);
    }

    if(buscar_maximos(&Vant[1],&V[1],Iint[1],&n[1],&m[1],&h[1],dt))
    {
        to[1][j]=t;
        j++;
        printf("\t\t\t%lf\n",to[1][j-1]-to[1][j-2]);
//
        fprintf(tiempo2,"%lf\t%lf\n",t,to[1][j-1]-to[1][j-2]);
    }
    fprintf(volt,"%lf\t%lf\t%lf\n",t,V[0],V[1]);
    fprintf(realimentacion,"%lf\t%lf\t%lf\n",t,I+Iint[0],Iint[1]);
}
}

```

buscar_maximos(double *V1, double *V2, double I, double *n, double *m, double *h, double dt)

Recibe apuntadores a las variables V, n,m, y h del modelo de Hodgkin y Huxley, también recibe el voltaje de la iteración anterior para calcular cuando ocurre un disparo. Dentro de esta función se ejecuta la función hh(double *V,double I, double *n, double *m, double *h, double dt) que calcula la siguiente iteración de la integración numérica. Buscar_maximos devuelve un 1 si en ese instante ocurre un máximo y devuelve cero si no ocurren máximos.

```

/*****
buscar_maximos Retorna un uno se el voltaje a través de la membrana pasa sobre cero voltios desde abajo, calcula la siguiente iteración
del voltaje a través de la membrana con el método de Runge Kutta, recibe las variables utilizadas en el modelo de Hodgkin y Huxley
*****/

```

```

...../

int buscar_maximos(double *v1,double *v2,double I, double *n, double *m,double *h,double dt)
{
    int b;
    /*v1 es el voltaje una iteracion antes que v2*/

    if(*v2>*v1) //si el voltaje es mayor que en la iteracion anterior, esta creciendo
    {
        *v1=*v2;
        hh(v2,I,n,m,h,dt);

        if(*v2>=0&&*v1<0)
            b=1;
        else b=0;
    }
    else
    {
        *v1=*v2;
        hh(v2,I,n,m,h,dt);
        b=0;
    }
    return b;
}

```

hh(double *V,double I, double *n, double *m, double *h, double dt)

Recibe un apuntador al voltaje y a las variables n,m y h. Con esto calcula la siguiente iteración por el método de integración numérica con Runge-Kutta.

```

void hh(double *V,double I,double *n,double *m, double *h,double dt)
{
    double kv[4],km[4],kn[4],kh[4],
           am,an,ah,
           bm,bn,bh;

    //calculo ais
    am= (0.1) * (*V + 40) / ( 1 - exp( - ( *V + 40) / 10 ) );
    an= (0.01) * (*V + 55) / ( 1 - exp( - ( *V + 55) / 10 ) );
    ah= (0.07) * exp( - ( *V + 65 ) / 20);

    //calculo bis
    bm= 4 * exp( - ( *V + 65 ) / 18 );
    bn= 0.125 * exp( - ( *V + 65 ) / 80 );
    bh= 1 / ( 1 + exp( - ( *V + 35 ) / 10));

    //calculo mhn
    km[0] = dt * der(am,bn,*m);
    kh[0] = dt * der(ah,bh,*h);
    kn[0] = dt * der(an,bn,*n);

    km[1] = dt * der(am,bn,*m + km[0]/2);
    kh[1] = dt * der(ah,bh,*h + kh[0]/2);
    kn[1] = dt * der(an,bn,*n + kn[0]/2);

    km[2] = dt * der(am,bn,*m + km[1]/2);
    kh[2] = dt * der(ah,bh,*h + kh[1]/2);
    kn[2] = dt * der(an,bn,*n + kn[1]/2);

    km[3] = dt * der(am,bn,*m + km[2]/2);
    kh[3] = dt * der(ah,bh,*h + kh[2]/2);
    kn[3] = dt * der(an,bn,*n + kn[2]/2);

    *m==km[0]/6 + km[1]/3 + km[2]/3 + km[3]/6;
    *h==kh[0]/6 + kh[1]/3 + kh[2]/3 + kh[3]/6;
    *n==kn[0]/6 + kn[1]/3 + kn[2]/3 + kn[3]/6;
}

```

```

//calculo *V
    kv[0]=dt*derv(*m,*h,*n,*V,I);
    kv[1]=dt*derv(*m,*h,*n,*V+kv[0]/2,I);
    kv[2]=dt*derv(*m,*h,*n,*V+kv[1]/2,I);
    kv[3]=dt*derv(*m,*h,*n,*V+kv[2],I);

    *V+=kv[0]/6 + kv[1]/3 + kv[2]/3 + kv[3]/6;

}

```

corriente(double t, double to[],int i)

Calcula la corriente de interacción de una de las neuronas debido a la otra. Se calcula según se describe en la sección 7.2

```

double corriente(double t,double to[500],int i)
{
    int j;
    double I=0;

    for(j=0;j<i;j++)
        I+=Cjk*alfa(t-to[j]);

    return I;
}

```

A.3.2 Acople 100 Neuronas

Generar patrones.

La función `patron()` recibe una semilla, el número de neuronas activas, y la cantidad de patrones que se desea almacenar y genera patrones aleatorios. Almacena los patrones en un archivo “patrones”

```

// librerías

#include <stdio.h>
#include <math.h>
#include <stdlib.h>
#include <time.h>

#define N 100 //número de neuronas activas

extern float ran2(long *); //funciones de Numerical Recipes
extern float ran1(long*);

/*****
patron(double p, double m, long semilla)
genera p patrones aleatorios donde el porcentaje de unos es m, los patrones consisten en 100 Neuronas de las cuales hay m prendidas,
toca introducir la semilla.
*****/

void patron(double p, double m,long semilla)
{
    int j=0,u=0,n=0,cantidad=0;
    double numero=0,f=0;
    FILE* patrones;

    patrones=fopen("patrones","w"); //En este archivo se almacena el número de patrones y los patrones.*/

    printf("\nSe almacenaron %.0f patrones con M= %.0f\n",p,m);
}

```

```

    fprintf(patrones,"%f\n",p);

    f=m/N; //porcentaje de neuronas activas

    for(n=0;n<N;n++) // El primer patrón
    {

        if(n<M) //consiste en las primeras M neuronas prendidas y el resto apagadas
            fprintf(patrones,"%d\t",1);
        else
            fprintf(patrones,"%d\t",0);

    }

    fprintf(patrones,"\n"); //cambia de patron

    for(u=1;u<p;u++) //El resto de patrones
    {
        cantidad=0; //cantidad guarda el número de neuronas prendidas por patrón
        for(n=0;n<N;n++) //Recorre las neuronas
        {
            numero=ran2(&semilla); //número aleatorio
            if(numero<f) //Si el número aleatorio es menor que el porcentaje de neuronas prendidas se almacena
            un 1
                {
                    fprintf(patrones,"%d\t",1);
                    cantidad++;
                }
            else
                fprintf(patrones,"%d\t",0);
        }
        fprintf(patrones,"\n"); //nuevo patron
        printf("%d\n",cantidad); //Como control
    }
    fclose(patrones);
}

```

añadir patrones

Esta función sirve para añadir patrones al archivo “patrones”, recibe una semilla, recibe el número de patrones que se van a almacenar y el número de neuronas activas.

```

/*****
Añade patrones a patrones previamente almacenados, no cambia el número de patrones escrito al principio del archivo patrones.
recibe el número de patrones que se van a almacenar, el número de neuronas prendidas, y la semilla.
*****/

void añadir_patrones(int numero,double m,long semilla)
{
    FILE *patrones;
    int i=0,n=0,p=0,cantidad=0;
    double f;
    float aleatorio;

    patrones=fopen("patrones","a"); //abre archivo para adjuntar

    f=m/N; //porcentaje de neuronas prendidas

    for(p=0;p<numero;p++) //recorre patrones
    {
        cantidad=0;
        for(n=0;n<N;n++) //recorre un patrón
        {
            aleatorio=ran2(&semilla); //número aleatorio

```

```

                                if(aleatorio<f)           //Si el número aleatorio es menor que el porcentaje de neuronas prendidas
se almacena un 1
                                {
                                    fprintf(patrones,"%d\t",1);
                                    cantidad++;
                                }
                                else
                                    fprintf(patrones,"%d\t",0);
                                }
                                fprintf(patrones,"\n");           //cambia de patron
//                                printf("%d\n",cantidad);
                                }

                                fclose(patrones);
}

```

Generar semilla

Esta función genera una semilla utilizando el reloj interno de la computadora.

```

/*****
generar_semilla genera semillas aleatorias para usar con los programas ran1.c y ran2.c de "Numerical Recipes", cada vez se genera
una semilla diferente.
*****/

void generar_semilla(long *s1)
{
    int stime;
    long ltime;

    ltime=time(NULL);
    stime=(unsigned int)ltime/2;
    srand(stime);
    *s1 = (long)-1*srand();
}

```

Generar patrones

Combina las funciones descritas anteriormente.

```

/*****
Genera Patrones aleatorios, la semilla se genera con el reloj del computador.
*****/

#include <stdio.h>
#include <math.h>
#include <stdlib.h>

extern float ran2(long*);
extern float ran1(long*);
extern patron(double,double,long);
extern e100n(int,double,double );
extern hh100n(double );
extern double mdt(int ,double,double);
extern anadir_patrones(int,double,long);
extern generar_semilla(long*);

main()
{
    int p=0,M,F;
    double eme=0,TMAX;
    long semilla;

    printf("\n\nEscriba número de patrones?\n");
    scanf("%d",&p);
}

```

```

printf("\nEscriba m? (neuronas activas) \n");
scanf("%d",&m);

generar_semilla(&semilla);
patron(p,m,semilla);

}

```

e100n(int patron,double tmax, double p)

Genera un archivo con la entrada para N neuronas, la entrada son pulsos para reproducir un patrón determinado, recibe el número del patrón que se desea reproducir (patrón), el tiempo que se desea efectuar la simulación y el número de patrones (p) que se tienen almacenados. Los patrones están almacenados en un archivo “patrones” generados por la función void patron(). Los patrones consisten en unos y ceros como los que son descritos en el documento.

```

/*****
librerías
*****/

#include <stdio.h>
#include <math.h>
#include <stdlib.h>

/*****
parámetros
*****/

#define Va 30
#define Vsyn -50
#define tao 2
#define gsyn 0.5
#define N 100

/*****
DEFINICIONES
*****/

double alpha( double );
//void e100n(int ,double);

/*****
FUNCIONES
*****/

/*****
Genera un archivo con la entrada para N neuronas, la entrada son pulsos para reproducir un patrón determinado, recibe el número
del patrón que se desea reproducir (patron), el tiempo que se desea efectuar la simulación y el número de patrones (p) que se tienen
almacenados. Los patrones están almacenados en un archivo patrones generados por g_patrones.c Los patrones consisten en unos y ceros.
*****/

void e100n(int patron,double tmax,double p)
{
    double t,dt=0.01,
           Is=0,Ip[N],p2;
    int n=0,m=1,M,T,i,prueba,vale_list[100];

    FILE *ent,*ent1,*vale;

    ent=fopen("in.in","w"); //aquí se escribe la entrada donde la primera columna es el tiempo y las otras N columnas son las
entradas.
    ent1=fopen("in","w"); //Este archivo es igual que el de arriba pero no incluye el tiempo, incluye tmax y dt.

```

```

vale=fopen("patrones","r"); //aquí están almacenados los patrones

fscanf(vale,"%lf\n",&p2); //para mover el apuntador en el primer espacio está el número de patrones almacenados

for(i=0;i<patron;i++) //recorre patrones, filas en el archivo patrones, para cuando se lee el patrón deseado.
  for(n=0;n<N;n++) //recorre neuronas, columnas en el archivo patrones
  {
    fscanf(vale,"%d\t",&prueba); //lee el patron
    vale_list[n]=prueba; //almacena el patron
    Ip[n]=0; //inicializar ip
  }

// vale_list[7]=0; //Se utilizaron para las simulaciones con ruido.
// vale_list[4]=0;
// vale_list[2]=0;
// vale_list[5]=0;
// vale_list[10]=0;

// vale_list[6]=0;
// vale_list[56]=1;
// vale_list[60]=1;
// vale_list[81]=1;

// printf("\n");

fprintf(ent1,"%lf\n%lf\n",dt,tmax); //Se almacena el tiempo de simulación y en intervalo de tiempo.

for(t=0;t<tmax;t+=dt) //Recorre el tiempo
{
    fprintf(ent,"%lf\t",t); //escribe el tiempo

    for(i=0;i<N;i++) //recorre las neuronas, columnas
    {
        if(vale_list[i]==1) //si la neurona es activa
            Ip[i]=gsyn*alpha(t)*(Va-Vsyn); //se tiene corriente sinaptica

        else Ip[i]=0; //si está apagada no hay corriente sinaptica

        fprintf(ent,"%lf\t",Is+Ip[i]); //se almacena la corriente, is=0
        fprintf(ent1,"%lf\t",Is+Ip[i]);
    }

    fprintf(ent1,"\n");
    fprintf(ent,"\n");
}

fclose(ent);
fclose(ent1);
fclose(vale);
}

```

void hh100n()

Esta función simula la recuperación del patrón. Recibe el número de patrones almacenados en la red. El voltaje en la membrana de las 100 neuronas se almacena en un archivo "out".

```

/*****
    librerías
*****/

#include <stdio.h>
#include <math.h>

```

```

#include <stdlib.h>

/*****
    CONSTANTES
*****/

#define C 1 //e-6
#define gna 120 //e-3
#define gk 36 //e-3
#define gl 0.3 //e-3
#define Vna 50 //e-3
#define Vk -77 //e-3
#define Vl -54.5 //e-3
#define td 10
#define tao 2
#define Cjk 40
#define N 100
#define GEXC 0.3
#define GINH 0.24
#define PMAX 100
#define Va 30
#define Vc -50
#define MAXMAX 50

/*****
    FUNCIONES
*****/

double alfa( double );
double der( double , double , double );
double deriv( double , double , double , double , double );
int buscar_maximos(double *,double *,double , double *, double *, double*,double);
void hh(double *,double ,double *,double *, double *,double);
double corriente(double ,double [N] [MAXMAX],short int [N],int [N],int );
void calcular_gexc(short int [N] [N],double p);
void escribir_maximos(int [N],double [N] [MAXMAX]);

/*****
HH100N con ayuda de otras funciones, corre la simulación de la red, escribe en el archivo "out" los voltajes de las N neuronas.
recibe el número de patrones almacenados en la red.
*****/

void hh100n(double p)
{
    double V[N],Vant[N],
           t,dt,tmax,to[N] [MAXMAX],
           I,Iint[N],m[N],n[N],h[N];
    int i=0,j=0,maximos[N];
    short int Gexc[N] [N],zita[N];
    char archivo[10]="in",archivo2[10]="out";
    FILE *volt,*entrada,*corr;
    volt=fopen(archivo2,"w"); //se almacena el voltaje en las N neuronas
    entrada=fopen(archivo,"r"); //aquí están las entradas de las N neuronas(ver e100n.c)
    corr=fopen("corriente","w"); //se almacenan las corrientes de acople

    calcular_gexc(Gexc,p); //Se calcula la matriz de pesos
    //inicialización de variables
    for(i=0;i<N;i++)
    {
        to[i] [0]=0;
        Iint[i]=0;
        V[i]=-65.025499;
        Vant[i]=-65.025499;
        n[i]=0.317286;
        m[i]=0.052774;
        h[i]=0.597012;
        maximos[i]=0; //cuenta el número de máximos de la neurona i
    }
    //inicializar máximos
    for(i=0;i<N;i++)
        for(j=0;j<MAXMAX;j++)to[i] [j]=0; //to[i] [j] es el tiempo del j-avo máximo de la neurona i
    fscanf(entrada,"%lf",&dt);
    fscanf(entrada,"%lf",&tmax);
}

```

```

//simulacion
for(t=0;t<tmax;t+=dt)
{
    fprintf(volt,"\n%lf",t); //se lee el tiempo
    fprintf(corr,"%lf",t);
    for(j=0;j<N;j++) //para cada neurona
    {
        fscanf(entrada,"%lf",&I); //lee la corriente de entrada externa
        Iint[j]=corriente(t,to,Gexc[j],maximos,j); //se calcula la corriente de acople
        fprintf(corr,"\t%lf",Iint[j]); //se almacena la corriente de acople
        /* dentro de buscar_maximos se corre el metodo de Runge- Kutta y retorna un uno si se lleugo a un máximo, */
        if(buscar_maximos(&Vant[j],&V[j],I+Iint[j],&n[j],&h[j],dt)) //si existe un máximo
        {
            to[j][maximos[j]]=t; //escribo el tiempo en que se obtuvo el máximo para la neurona j
        }
        maximos[j]++; //incremento el número de máximos de la neurona j
        fprintf(volt,"\t%lf",V[j]); //Se escribe en volt el voltaje de la membrana para la neurona j
        fscanf(entrada,"%n",&I); //Se lee la entrada para la siguiente neurona
    }
    fprintf(corr,"\n");
}
escribir_maximos(maximos,to); //se escriben los tiempos en que se obtuvieron máximos
fclose(volt);
fclose(entrada);
fclose(corr);
}

```

Almacenación de patrones utilizando la regla de aprendizaje dada por la ec. 7.13.

Está función calcula los pesos sinápticos de la red utilizando una de las reglas de Willshaw, la matriz

```

/*****
Calcula la matriz de pesos sinapticos, a partir de los patrones que se van a almacenar.
*****/
void calcular_gexc(short int Gexc[N][N], /*matriz de pesos sinapticos*/
double p /* número de patrones aprehendidos*/)
{
    int zita[N][PMAX];
    int j,k,u;
    double suma,p2;
    FILE *pat;

    pat=fopen("patrones","r"); //aquí están los patrones
    fscanf(pat,"%lf",&p2); //cuantos patrones hay en la matriz
    /* Se leen solo el número de patrones que se desean almacenar y se escriben en zita*/
    for(u=0;u<p;u++)
        for(j=0;j<N;j++)
            fscanf(pat,"%d",&zita[j][u]);
    /* Calculo de pesos sinapticos, con la regla escrita en el documento*/
    for(j=0;j<N;j++) //es una matriz de N x N
    {
        for(k=0;k<N;k++)
        {
            suma=0;
            for(u=0;u<p;u++)
                suma+=zita[j][u]*zita[k][u];
            if(suma>0)
                Gexc[j][k]=1;
            else
                Gexc[j][k]=0;
        }
    }
    fclose(pat);
}

```

Almacena patrones utilizando la regla de aprendizaje (ec.7.14)

```

/*****
Calcula la matriz de pesos sinapticos, a partir de los patrones que se van a almacenar.
*****/

void calcular_gexc2(short int Gexc[N][N], /*matris de pesos sinapticos*/
double p /* número de patrones aprehendidos*/)
{
    int zita[N][PMAX];
    int j,k,u;
    double suma,p2;
    FILE *pat;

    pat=fopen("patrones","r"); //aquí están los patrones

    fscanf(pat,"%lf\n",&p2); //cuantos patrones hay en la matris

    /* Se leen solo el número de patrones que se desean almacenar y se escriben en zita*/
    for(u=0;u<p;u++)
        for(j=0;j<N;j++)
            fscanf(pat,"%d",&zita[j][u]);

    /* Calculo de pesos sinapticos, con la regla escrita en el documento*/

    for(j=0;j<N;j++) //es una matris de N x N
    {
        for(k=0;k<N;k++)
        {
            suma=0;
            for(u=0;u<p;u++)
                suma+=zita[j][u]*zita[k][u];
            /* if(suma>0)
                Gexc[j][k]=1;
            else
                Gexc[j][k]=0;*/
            Gexc[j][k]=suma;
        }
    }
    fclose(pat);
}

```

Calculo de corriente de interacción.

Calcula la corriente de acople que entra en una neurona j, debido al aporte de todas las otras neuronas.

```

/*****
Corriente calcula la corriente de acople que le entra a la neurona j, es una superposicion del aporte sinaptico de todas las otras
neuronas.
*****/

double corriente(double t,double to[N][MAXMAX],short int Gexc[N],int maximos[N],int j)
{
    int k,m;
    double I=0,I2,G;

    /*se recorren los picos de voltajes (máximos) de todas las neuronas, para determinar la corriente de acople*/
    for(k=0;k<N;k++)
    {
        for(m=0;m<maximos[k];m++)
            if(k!=j)
            {
                if(Gexc[k]) //si el proceso es excitatorio.( si hay conexion con la neurona k)
                    G=0.06;
                else G=-0.24; //Si es inhibitorio (no hay conexion entre j y k)

                //contribucion de la neurona k, cada pico (maximo m) genera un pico de corriente en to[k][m] con un tiempo de retraso td,
                I2=G*(Va-Vc)*alfa(t-td-to[k][m]);
                I+=I2;
            }
    }
}

```

```

        if(i>0)
            return i;
        else
            return 0;
    }

```

Escribir máximos

Esta función escribe los máximos del vector $to[i][n]$ en un archivo "maximos".

En el vector $to[i][n]$ se tienen el n-esimo tiempo de disparo de la neurona i.

En escribir_maximos se escriben los máximos de las N neuronas en un archivo "maximos", cada columna es una neurona y cada fila es un tiempo en que ocurrió un máximo, la primera fila escribe el número de máximos para cada neurona, Recibe el vector con el número de máximos de cada neurona, y la matriz con los tiempos en que se obtuvieron esos máximos.

```

...../
void escribir_maximos(int maximos[N],double to[N][MAXMAX])
{
    FILE *max;
    int j,m,maximo=0;
    max=fopen("maximos","w");
    for(j=0;j<N;j++)
    {
        fprintf(max,"%d\t",maximos[j]);
        if(maximos[j]>maximo)
            maximo=maximos[j];
    }
    fprintf(max,"\n");
    for(m=0;m<maximo;m++)
    {
        for(j=0;j<N;j++)
            fprintf(max,"%f\t",to[j][m]);
        fprintf(max,"\n");
    }
    fclose(max);
}

```

calcula la variable $m(t)$.

Esta función evalua la recuperación del patrón p. y devuelve el valor de la variable $m(t)$.

```

...../
    librerías
...../

#include <stdio.h>
#include <math.h>

...../
    parámetros
...../

#define N 100
#define MAXMAX 150
#define hp 2.45
#define td 10
#define tr 2.54

...../
    DEFINICION DE FUNCIONES
...../

double emedete(int [N],int [N],double [N][MAXMAX],double);
void leer_patrones(int [N],int);
void leer_maximos(int [N],double [N][MAXMAX]);

```

```

double mdt(int, double,double);

/*****
mdt recibe el patrón que se desea analizar el tiempo de simulación y el número de patrones almacenados en la red. calcula la
variable eme, y neta, para saber si fue una recuperación exitosa, neta muestra la evolución de la recuperación del patrón en el tiempo
*****/

double mdt(int p,double tsax,double patrones)
{
    int zita[N];
    int maximos[N];
    double to[N] [MAXMAX];
    leer_maximos(maximos,to);
    leer_patrones(zita,p);
    return emedete(zita,maximos,to,tsax);
}

```

leer patrones

Esta función recibe el número de patrón que se desea recuperar, lo lee del archivo “patrones” y lo escribe en el apuntador `vale_list[]`.

```

/*****
leer_patrones recibe el número patrón que se desea probar y lo lee del archivo "patrones", lo escribe en vale_list[]
*****/
void leer_patrones(int vale_list [N],
int patron /*patrón a probar*/)
{
    FILE* pattern;
    int i,n;
    double p2;

    pattern=fopen("patrones","r");
    fscanf(pattern,"%lf",&p2);
    for(i=0;i<patron;i++)
        for(n=0;n<N;n++)
            fscanf(pattern,"%d\t",&vale_list[n]);
    fclose(pattern);
}

```

leer máximos

Esta función lee los máximos que están escritos en el archivo “maximos” y los almacena en el vector `to[][]`.

```

/*****
leer_maximos lee los tiempos en que ocurrieron los picos de las N neuronas, estos están escritos en el archivo "maximos", máximos
almacena el número de máximos y to[j] [n] tiene el n-avo máximo de la neurona j.
*****/
void leer_maximos(int maximos[N],double to[N] [MAXMAX])
{
    FILE *max;
    int j,m,maximo=i;
    max=fopen("maximos","r");
    for(j=0;j<N;j++)
    {
        fscanf(max,"%d\t",&maximos[j]);
        if(maximos[j]>maximo)
            maximo=maximos[j];
    }
    fscanf(max,"\n");
    for(n=0;n<=maximo;n++)
    {
        for(j=0;j<N;j++)
            fscanf(max,"%lf\t",&to[j] [n]);

        fscanf(max,"\n");
    }
}

```

```

    }
    fclose(max);
}

```

Evalua la recuperación de los patrones

Calcula la variable $m(t)$ para evaluar que tan buena fue la recuperación del patrón.

```

/*****
emedete calcula m(t), mira si la recuperación fue exitosa. cada tr + td =12.54 mseg +- 2.45 mseg mira cuales neuronas tuvieron
un máximo, estas son las exitosas.
*****/
double emedete(int zita[N],int maximos[N],double to[N][MAXMAX],double tmax)
{
    double T[MAXMAX],eme,dif,m2,m3;
    int t,idiota,n,num;
    FILE *archivo,*neta;
    archivo=fopen("eme","w");
    neta=fopen("neta","w");
    //calculo los periodos
    T[0]=2;
    // printf("\n Cual es tmax?\n");
    // scanf("%lf",&tmax);
    for(t=1;T[t]<tmax;t++)
        T[t]=T[t-1]+td+tr; //T[t] tiene los tiempos en que debe haber máximos
    for(t=0;T[t]<tmax;t++)
    {
        eme=0;
        for(n=0;n<N;n++)
        {
            idiota=0; //contador para eme
            for(num=0;num<=maximos[n];num++) //recorre los máximos
            {
                dif=fabs(to[n][num]-T[t]); // es la diferencia entre el máximo ideal y el obtenido
                if(dif<=hp&&to[n][num]!=0) // si la diferencia es menor que 2.45 y el máximo no es cero.
                {
                    idiota+=1;
                    fprintf(neta,"%i.%lf\t%d\n",T[t],n+1); //se escribe neta
                }
            }
            //calculo de eme (ver documento)
            m2=(2*zita[n]-1);
            m3=(2*idiota-1);
            eme+=m2*m3;
        }
        fprintf(archivo,"%lf\t%lf\n",T[t],eme); //se escribe eme
    }
    fclose(archivo);
    fclose(neta);
    return eme;
}

```

Programa Principal

Este programa es la integración de todos los programas, genera patrones, almacena patrones, e intenta recuperar un patrón, además mira que tan exitosa estuvo la recuperación.

```

/*****
librerías
*****/
#include <stdio.h>
#include <math.h>
#include <stdlib.h>

```



```

#define P 1
#define TMAX 500
#define N 100
/*****
    FUNCIONES REQUERIDAS
    *****/
extern float ran2(long*);
extern float ran1(long*);
extern patron(double,double,long);
extern e100n(int,double,double );
extern hh100n(double );
extern double mdt(int ,double,double);
extern anadir_patrones(int,double,long);
extern generar_semilla(long*);
/*****
    PROGRAMA PRINCIPAL
    *****/
main()
{
    int p=0,pnuevo,abajo,arriba,mitad,dif,m,i;
    FILE *popeye,*hkp;
    double eme=0;
    long semilla;
    // popeye=fopen("estable","a"); //escribe los resultados de la recuperación de los patrones.
    // hkp=fopen("archivo","a"); //Se escribe el número de neuronas activas contra el máximo número de patrones que se pueden
almacenar
    arriba=N; //limites de la busqueda
    abajo=0;
    dif=arriba-abajo; //se utiliza para la busqueda binaria
    generar_semilla(&semilla);
    // scanf("%d",&m);
    for(m=6;m<=25;m+=1) //para cada uno de estos emes se busca el optimo de patrones
    {
        popeye=fopen("estable","a"); //escribe los resultados de la recuperación de los patrones.
        hkp=fopen("archivo","a"); //Se escribe el número de neuronas activas contra el máximo número de patrones que
se pueden almacenar
        generar_semilla(&semilla);
        arriba=N;
        abajo=0;

        dif=arriba-abajo;
        mitad=dif/2;
        p=mitad;
        pnuevo=mitad;

        patron(N,m,semilla); //crea el archivo con N patrones (son suficientes) con m neuronas prendidas
        while(dif>=5) //mientras la busqueda se acerca
        {
            fprintf(hkp,"m=%d p=%d\t",m,pnuevo);
            printf("\tM= %d\t p= %d\t",m,pnuevo);
            e100n(P,TMAX+50,pnuevo); //se crea un archivo con las entradas
            hh100n(pnuevo); //se corre la simulacion
            eme=mdt(P,TMAX,pnuevo); //se analiza el resultado de la simulacion

            fprintf(hkp,"\t\t\t%.0lf\n",eme); //eme dice que tan exacto se recupero el patron
            printf("%lf\n",eme);
            if(eme!=N)arriba=mitad; //busqueda binaria
            else abajo=mitad;
            dif=arriba-abajo;

            mitad=(arriba+abajo)/2;
            pnuevo=mitad;
        }
        /**** Cuando la busqueda ya está lo suficientemente cerca se comienza a aumentar uno a uno el número de patrones memorizados
        *****/
        eme=N;
        pnuevo=abajo;
        while(eme==N) //mientras se recuperen perfectamente
        {
            fprintf(hkp,"m=%d p=%d\t",m,pnuevo);
            printf("\tM= %d\t p= %d\t",m,pnuevo);
            e100n(P,TMAX+50,pnuevo); //escribe entradas
            hh100n(pnuevo); //corre simulacion
        }
    }
}

```

```
    eme=mdt(P,TMAX,pnuevo);          //analiza la recuperación
    fprintf(hkp, "\\t\\t\\t%.0lf\\n", eme);
    printf("eme= %lf\\n", eme);
    pnuevo++;                          //aumenta el número de patrones
}

fprintf(popeye, "%d\\t%d\\n", m, pnuevo-2);

fclose(hkp);
fclose(popeye);
}
}
```

Bibliografía

- [1] GRIFFITH J. S. *Mathematical Neurobiology*, Academic Press, London 1971
- [2] A. L. HODGKIN, A. F. HUXLEY, *J. Physiol.* **117**, 500 (1952).
- [3] A. L. HODGKIN, A. F. HUXLEY, *Nature* (london), Action potentials recorded from inside a nerve fiber, **144**;710-711.
- [4] M. J. ZIGMOND, F. E. BLOOM, S. C. LANDIS, *Fundamental Neuroscience*, Academic Press, p. 129-154.
- [5] KOCH Christof. *Biophysics of Computation*, Oxford University Press, 1999.
- [6] PURVES *Life, The Science of Biology*, Freeman.
- [7] H. HASEGAWA (E-preprint: cond-mat/9907387).
- [8] R. FITZHUGH, *Biophys. J.* **1**, 445(1961).
- [9] J. S. NAGUMO, YOSHIZAWA S. ARIMOTO, *Proc. IRE* **50**, 2061(1962).
- [10] M. J. ZIGMOND, F. E. BLOOM, S. C. LANDIS, *Fundamental Neuroscience*, Academic Press, p. 1470-1485.
- [11] J.J. HOPFIELD, *Proc. Natl. Acad. Sci. USA* **79** (1982) 2554.
- [12] H. HASEGAWA (E-preprint: cond-mat/9906020).
- [13] H. HASEGAWA (E-preprint: cond-mat/0012033).
- [14] H. HASEGAWA (E-preprint: cond-mat/0007198).
- [15] M. YOSHIOKA, M. SHIINO, *Phys. Rev. E*, **58**, 3628 (1998).

- [16] J.J.B. JACK, D. NOBLE, R. W. JSIEN, *Electric Current Flow in Excitable Cells* (Clarendon Press, Oxford, 1975).
- [17] W. GERSTNER, J. L. VON HEMMEN, *Network* **3**, 139 (1992).
- [18] W.W. LYTTON, *J.Comput. Sci.* **5**, 353 (1998).
- [19] Numerical Recipies, www.nr.com.
- [20] D. J. WILSHAW, O. P. BUNEMAN, H. C. LONGUE-HIGGINS, *Nature* **222**, 960 (1969).
- [21] VAN DER POL, VAN DER MARK, "The heartbeat considered as a relaxation oscillation and an electrical model of the heart" *Phil. Mag. Suppl.* **6**: 763-775 (1928).
- [22] R. MUELLER, A. V. M. HERZ, *Phys. Rev. E* **59**, 3330 (1999).
- [23] W. MAASS, *Neural Comput.* **9** (1997) 279.
- [24] S. THORPE, D. FIZE, C. MARLOT, *Nature*, **381**, 520 (1996).
- [25] E.T.ROLLS, M. J. TOVEE, *Proc. Roy. Soc B*, **257**, 9 (1994).