Universidade de São Paulo

Ciências Moleculares

Relatório EP - Modelo de Hodgkin-Huxley

Alunos
Eduardo Falbel - 11262900
Gabriel Schwartz - 5486096
Lucca Pagnan - 11259329
Heloísa Lazari - 11093093

Professor
Saulo R. M. Barros

Contents

1 Introdução

Nesse EP, buscamos implementar um modelo descrito por equações diferenciais ordinárias (EDOs) e resolvê-lo de maneira numérica, utilizando as técnicas vistas em aula. Para isso, escolhemos o modelo de Hodgkin-Huxley, que descreve o potencial de ação de neurônios, e utilizamos a linguagem Python 3 e as seguintes bibliotecas:

- **numpy:** Utilizada para realizar cálculos eficientes com matrizes e vetores.
- **pathlib:** Utilizada para padronizar as *filepaths* e evitar conflito entre sistemas operacionais (uso de \no Windows e / no Linux/Unix, por exemplo).
- **matplotlib:** Utilizada para visualizar o modelo em execução.
- **multiprocessing:** Otimização de execução. Desenvolvemos nosso programa de modo a permitir que várias simulações sejam calculadas simultaneamente, acelerando muito a execução.
- **argparse:** Essa biblioteca foi utilizada para configurar as opções na interface de linha de comando (CLI) para o programa.

2 Contextualização

2.1 História da Neurociência

Os primeiros registros de procedimentos no cérebro datam do período neolítico; entre 5 e 10% de todos os crânios encontrados desse período possuem buracos que evidenciam a realização de Trepanação, uma intervenção cirúrgica que busca aliviar a pressão craniana removendo parte do crânio e expondo a dura máter (uma camada de proteção do cérebro parecida com couro). No Egito antigo, o cérebro era visto como um tecido conjuntivo do crânio, sem grande relevância para a fisiologia corporal, e o processo de mumificação envolvia a remoção da massa cerebral. Nessa época, acreditava-se que o coração era responsável pelos pensamentos e emoções.

A ideia de que o coração era a fonte da consciência não foi contestada até a época do filósofo e médico grego Hipócrates. Ele acreditava que o cérebro não estava apenas envolvido com as sensações - já que a maioria dos órgãos especializados (por exemplo, olhos, ouvidos, língua) estão localizados na cabeça perto do cérebro - mas também era a sede da inteligência. Outros filósofos gregos e romanos também contribuíram para essa hipótese. Platão defendia que o cérebro abrigava a parte racional da alma, e o médico romano Cláudio Galeno descreveu diversos casos de pacientes que perderam suas faculdades mentais frente a grandes traumas cranianos e outros danos no cérebro.

Entretanto, durante a Europa da Idade Média, foi disseminada a ideia do coração como fonte do pensamento e emoções, defendido pelo filósofo grego Aristóteles e apoiado pela Igreja Católica. Nesse período, as principais contribuições para o estudo do sistema nervoso vieram do Mundo Islâmico, com diversos médicos muçulmanos descrevendo diversas doenças e tratamentos associados ao cérebro. Com o início do renascimento e o resgate dos valores culturais e intelectuais greco-romanos, preservados e cultivados pelos filósofos e intelectuais muçulmanos, o cérebro volta a ser visto como responsável pelos pensamentos e sensações, e diversas contribuições para a neurociência, principalmente para a neuroanatomia, foram feitas nessa época.

2.2 Descoberta e função do neurônio

Em meados do século XVIII, o trabalho de Luigi Galvani sobre a excitabilidade elétrica de músculos e neurônios deu início a uma série de descobertas sobre o funcionamento do sistema nervoso. Entre o início do século XVIII e o final do século XIX, fenômenos elétricos envolvendo nervos e músculos foram descritos em diversas situações fisiológicas. Em 1843, Emil du Bois-Reymond demonstrou a natureza elétrica do sinal nervoso, e em 1875, Richard Caton encontrou fenômenos elétricos nos hemisférios cerebrais de coelhos e macacos. Mais tarde, em 1890, Adolf Beck publicou observações semelhantes da atividade elétrica espontânea do cérebro de coelhos e cães.

Contudo, foi apenas com o advento das técnicas de microscopia que os primeiros estudos do sistema nervoso a níveis celulares puderam ser realizados. No final da década de 1890, o biólogo e patologista Camilo Golgi desenvolveu a técnica de coloração por prata, que permite a visualização de tecido nervoso em microscópios ópticos. Essa técnica foi amplamente utilizada pelo neurocientista e patologista Santiago Ramón y Cajal, que estudou a anatomia e fisiologia de diversas partes do sistema nervoso e levou ao desenvolvimento da chamada doutrina neuronal, que define o neurônio como unidade funcional do cérebro (e de todo o sistema nervoso). Ramón y Cajal também postulou a Lei da Polarização Dinâmica, que afirma que um neurônio recebe sinais em seus dendritos e no corpo celular e os transmite, como potenciais de ação, ao longo do axônio em uma direção:

para longe do corpo celular. Em 1906, ambos dividiram o Prêmio Nobel de Medicina e Fisiologia por suas descobertas.

2.3 Neurociência no século XX e o trabalho de Hodgkin-Huxley

No início do século 20 a neurociência inicia seu processo de independência de outras disciplinas, passando a focar no seu próprio objeto de estudo: o neurônio. Com a teoria celular já bem estabelecida e os avanços nas técnicas bioquímicas, iniciam-se diversos estudos que buscam caracterizar a fisiologia do neurônio e como se propagam os potenciais de ação definidos por Ramón y Cajal. Em 1937, o influente biólogo britânico John Zachary Young propôs que o axônio gigante encontrado em lulas poderia ser utilizado para estudar as propriedades elétricas do neurônio.

Em 1952, Alan Lloyd Hodgkin e Andrew Huxley apresentaram um modelo matemático para transmissão de sinais elétricos em neurônios do axônio gigante de uma lula, os chamados potenciais de ação, e como eles são iniciados e propagados. Esse modelo ficou conhecido como modelo de Hodgkin-Huxley e é utilizado até hoje. Em seus experimentos, Hodgkin e Huxley estudaram as propriedades dos canais de K+ e Na+ na lula gigante axônio. Em particular, eles queriam estudar a dependência da tensão e a dependência temporal dos canais de K+ e Na+. Para tanto, eles utilizaram a pinça de tensão, um dispositivo que mantém o potencial de membrana de uma célula a qualquer tensão de "comando" desejada (V_c) e mede a corrente necessária para manter essa tensão. Em 1963, eles dividiram o Premio Nobel de Medicina e Fisiologia por seu trabalho.

3 O modelo

O modelo de Hodgkin–Huxley em si é um circuito elétrico (figura 1) equivalente a como ocorre o *spike* de um neurônio, descrito por um conjunto de equações diferenciais. Nesse modelo, temos que a corrente que percorre o neurônio é dada por:

$$I = C_m \frac{\mathrm{d}V_m}{\mathrm{d}t} + g_K(V_m - V_K) + g_{Na}(V_m - V_{Na}) + g_l(V_m - V_l), \tag{1}$$

sendo C_m a capacitância da membrana do axônio, V_m o potencial elétrico da membrana, g_K a condutância do canal de potássio, V_K o potencial de Nernst[1] do canal de potássio, $g_N a$ a condutância do canal de sódio, $V_N a$ o potencial de Nernst do canal de sódio, g_l a condutância do canal de vazamento e $V_N a$ o potencial de Nernst do canal de vazamento. A complexidade do modelo advém da dependência da condutância dos canais de sódio e potássio tanto com voltagem quanto com tempo, além da dependência da voltagem de canal de potássio com o tempo.

Para completamente descrever o comportamento da corrente sináptica, portanto, necessitam-se de outras equações diferenciais e a introdução de novas variáveis. Chegamos, então, no modelo final:

$$I = C_m \frac{\mathrm{d}V_m}{\mathrm{d}t} + \bar{g}_{\mathrm{K}} n^4 (V_m - V_K) + \bar{g}_{\mathrm{Na}} m^3 h (V_m - V_{Na}) + \bar{g}_l (V_m - V_l) \tag{2}$$

$$\frac{dn}{dt} = \alpha_n(V_m)(1-n) - \beta_n(V_m)n \tag{3}$$

$$\frac{dm}{dt} = \alpha_m(V_m)(1-m) - \beta_m(V_m)m \tag{4}$$

$$\frac{dh}{dt} = \alpha_h(V_m)(1-h) - \beta_h(V_m)h \tag{5}$$

A equação (2), em relação à (1), adiciona três parâmetros e modifica outros três: $n, m, h, \bar{g}_K, \bar{g}_{Na} e \bar{g}_l$. Os três primeiros são valores adimensionais que variam entre 0 e 1 e indicam a abertura de cada um dos três canais iônicos (potássio, sódio e vazamento, respectivamente). Com eles, podemos substituir as variáveis de condutância na equação original por essas novas, que indicam apenas a condutância total dos canais, eliminado sua dependência temporal e transferindo-a para as novas. Disso surgem as equações (3),(4) e (5). Elas explicitam essa dependência e permitem o cálculo iterativo das variáveis e, consequentemente, da corrente de *spike*.

Os parâmetros α_i e $\beta_i, i = n, m, h$ são constantes de velocidade para cada um dos canais iônicos e podem ser calculadas de vários jeitos. Nós usamos as seguintes equações:

$$\begin{array}{lll} \alpha_n(V_m) = \frac{0.01(55+V_m)}{-\exp\left(\frac{55+V_m}{10}\right)+1} & \alpha_m(V_m) = \frac{0.1(40+V_m)}{-\exp\left(\frac{65+V_m}{10}\right)+1} & \alpha_h(V_m) = 0.07\exp\left(-\frac{V_m+65}{20}\right) \\ \beta_n(V_m) = 0.125\exp\left(-\frac{V_m+65}{80}\right) & \beta_m(V_m) = 4\exp\left(-\frac{V_m+65}{18}\right) & \beta_h(V_m) = \frac{1}{\exp\left(-\frac{35+V_m}{10}\right)+1} \end{array} \tag{6}$$

[1] O potencial de Nernst é o potencial elétrico no qual não há fluxo de íons pelo canal.

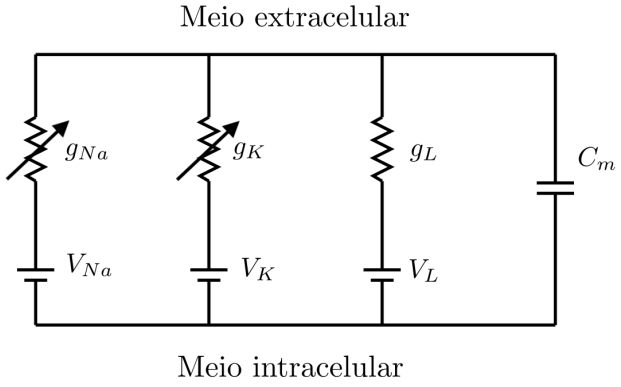

Figure 1: Circuito elétrico representativo do processo sináptico de um neurônio

4 Resolução das EDOs

A parte matemática desse EP está presente no programa ode—solutions.py, onde há o algorítimo de Runge-Kutta de quarta ordem e Euler para a solução de EDO. Os dois métodos foram escolhidos pois o modelo é apenas uma ilustração do que ocorre nos neurônios, a voltagem no resultado é apenas uma representação do que é possível em situações reais, e minimizar o erro não é uma grande preocupação, pois a partir da forma do gráfico já é possível saber se ouve grandes destoações do resultado ideal.

```
def solving_method(f_prime, x, n, interval):
    t0 = interval[0]
    tf = interval[1]
    t = t0
    h = (tf-t0)/n
```

Ambos os métodos de solução recebem os mesmos parâmetros: A equação diferencial, um valor inicial x, o número de de iteração a serem feitos pelo algorítimo, e o intervalo total de análise. A partir disso, podemos achar h, o tamanho de cada iteração do algorítimo.

Como o modelo de Hodgkin-Huxley é um sistema de 4 equações diferenciais, a melhor maneira de resolver é vetorizar o algorítimo para que todas as equações sejam resolvidas ao mesmo tempo para cada passo dado. Assim as constantes são atualizadas.

Para isso, utilizamos as arrays do numpy, as equações e parâmetros são essas arrays como argumentos das funções de resolução. O resultado é dado como uma array normal de arrays de numpy, com o tempo e os valores de voltagem, inicializado da seguinte maneira

```
points = [np.array([t,*x])]
```

4.1 Euler

4.1.1 Teoria

O método de Euler é o mais simples e o mais rápido, mas por consequência possui o maior erro. Ele se baseia em estender a o valor de $f(t_n)$ na direção da derivada de f nesse ponto, da maneira que $f(t_{n+1})$ é o valor da derivada no ponto $t + h$, onde h é tamanho do intervalo.

4.1.2 Implementação

```
def euler(f_prime, x, n, interval):
    t0 = interval[0]
    tf = interval[1]
    t = t0
    h = (tf-t0)/n
    points = [np.array([t,*x])]
    for i in range(0,n):
        x = x + h*np.array(f_prime(t,x))
        t = (i + 1)*h
```

```
        points.append(np.concatenate(([t],x)))
    return np.vstack(points)
```

4.2 Runge-Kutta

4.2.1 Teoria

O método de Runge-Kutta foi desenvolvido no início do século XX por matemáticos alemães e está na família dos métodos explícitos-implícitos de resolução de equações diferenciais ordinárias. Onde o valor de $f(t_{n+1})$ é dado pela seguinte expressão em função de $f(t_n)$:

$$f(t_{n+1}) = f(t_n) + \frac{1}{6}h(k_1 + 2k_2 + 2k_3 + k_4) \tag{7}$$

As constantes k_n são derivadas no intervalo, onde k_1 é derivada no início do intervalo utilizando $f(t_n)$, que em si é o método de Euler, k_2 é o valor no meio do intervalo partindo de $f(t_n)$ seguindo a linha de k_1, k_3 possui a mesma lógica que o anterior mas utilizando a linha de k_2. E, para finalizar, k_4 é a derivada no final do intervalo na reta k_3.

4.2.2 Implementação

```
def runge_kutta(f_prime, x, n, interval):
    t0 = interval[0]
    tf = interval[1]
    t = t0
    h = (tf-t0)/n
    points = [np.array([t,*x])]
    for i in range(0,n):
        k1 = np.array(f_prime(t,x))
        k2 = np.array(f_prime(t+0.5*h, x+0.5*k1*h))
        k3 = np.array(f_prime(t+0.5*h, x+0.5*k2*h))
        k4 = np.array(f_prime(t+h, x+h*k3))
        x = x + h/6*(k1+2*k2+2*k3+k4)
        t = t0 + (i + 1)*h
        points.append(np.concatenate(([t],x)))
 return np.vstack(points)
```

5 O programa

5.1 Arquitetura Geral

A implementação do modelo foi feita em Python. Para calcular o sistema utilizando as quatro equações simultaneamente, nós definimos uma nova função $R^4 \to R^4$, que tinha como componente cada uma das variáveis que estávamos calculando. Com os métodos de cálculo vetorizados, foi possível executar essa simulação apenas aplicando esses métodos à essa nova função.

Além de várias constantes, o modelo utiliza alguns valores iniciais e a corrente a ser simulada, retornando um conjunto de pontos no tempo que aproximam o comportamento do modelo. Desse modo, para produzir um gráfico animado nós calculamos o comportamento do modelo para um intervalo de correntes.

5.1.1 Multiprocessing

O programa funciona calculando, para cada corrente em uma faixa, as aproximações resultantes da aplicação do método escolhido ao sistema de EDOs. Como esses cálculos são independentes entre si, ou seja, uma corrente pode ser calculada sem nenhuma informação de outras, nós aceleramos a execução do código paralelizando a aplicação da aproximação escolhida com a biblioteca *multiprocessing* e utilizando metade das cores disponíveis.

5.1.2 Cache

Para facilitar diferentes visualizações em situações com os mesmos parâmetros, o programa salva em uma pasta cache os dados brutos resultantes de sua execução, com variáveis no nome do arquivo que representam os parâmetros dados. Ao executar o programa com um conjunto de parâmetros, ele procura um arquivo cache equivalente e, se um existe, apenas carrega os dados ao invés de calcular do zero.

5.1.3 Argumentos da Linha de Comando

Para rodar este programa, foi escolhido utilizar uma interface de linha de comando, dando algumas liberdades ao usuário, utilizando a biblioteca *argparser*.

```
$ equations.py [-h] [-tf FINAL_TIME] (-E | -R) [-V | -N | -M | -H]
```

A opção $-h$ é apenas um menu de ajuda, caso o usuário deseja. É obrigatório especificar o método utilizado através das opções $-R$ para Runge-Kutta de quarta ordem ou $-E$ para Euler.

Por padrão, o output do programa é a animação de um gráfico da voltagem por tempo, variando a corrente medida no neurônio. Esse comportamento pode ser especificado com a $-V$.

Também é possível mostrar a animação do gráfico das constantes de abertura dos canais n, m, h com as opções $-N, -M, -H$, respectivamente. Uma ultima opção é o tempo total de análise, o padrão é 300ms, mas ele pode ser alterado com a opção $-tf$ seguida de um número inteiro, que será o tempo em ms.

5.2 Constantes e parâmetros iniciais

As constantes e parâmetros iniciais que precisam ser estabelecidos para a execução do programa são aquelas referentes às propriedades físicas do axônio sendo simulado supramencionadas (potenciais e condutâncias). Optamos por utilizar os mesmos valores obtidos no artigo original, portanto o axônio sendo simulado é o de um neurônio de uma lula-gigante.

Constantes:

- $\bar{g}_K$: $36 mS/cm^2$
- $\bar{g}_{Na}$: $120 mS/cm^2$
- $\bar{g}_l$: $0.3 mS/cm^2$
- V_m: $-65 mV$
- V_K: $-77 mV$
- V_{Na}: $50 mV$
- V_l: $-54.387 mV$
- C_m: $1 mF/cm^2$

Parâmetros iniciais:

- n: 0.05
- m: 0.32
- h: 0.6

5.3 Output

O programa produz um gráfico do potencial de ação para cada corrente de *input* no intervalo de -1 até 10 miliAmpère em 100 passos (portanto em incrementos de 0,11mA). Durante a execução, são produzidos gráficos como os das figuras 2,3 e 4.

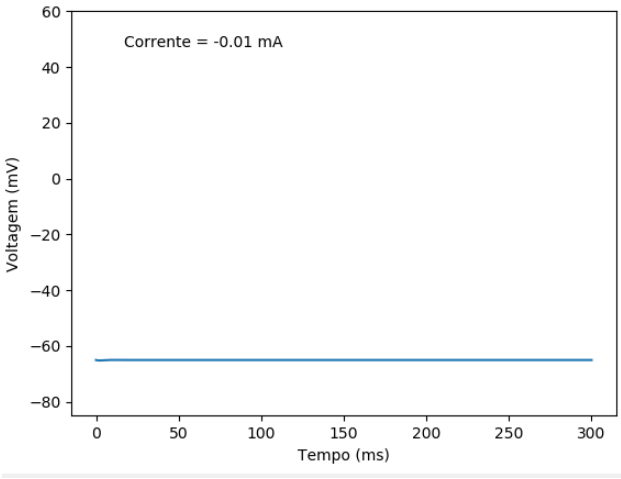

Figure 2: Potencial de ação para corrente 0,01mA. EDO resolvida por Runge-Kutta.

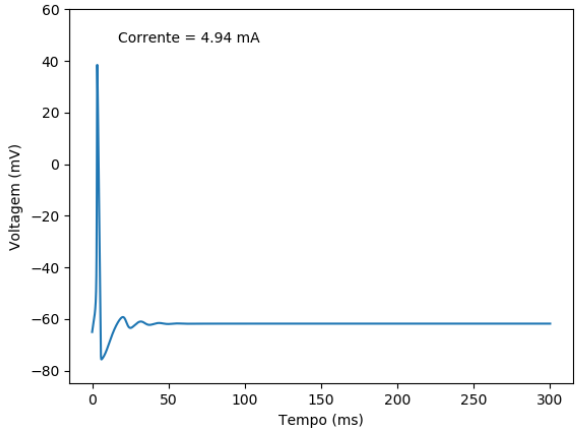

Figure 3: Potencial de ação para corrente 4,94mA. EDO resolvida por Runge-Kutta.

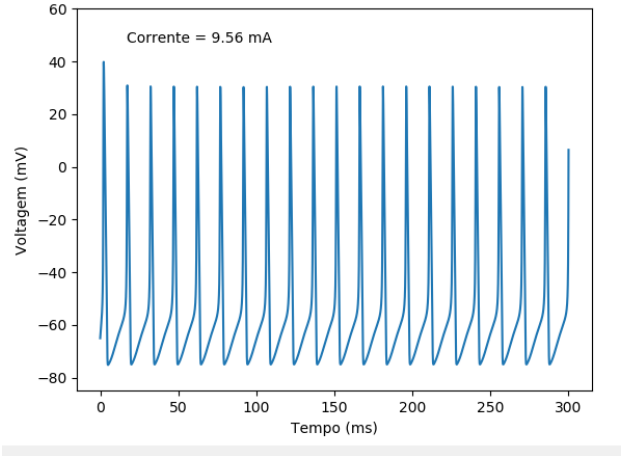

Figure 4: Potencial de ação para corrente 9,56mA. EDO resolvida por Runge-Kutta.

6 Considerações finais

Aprendemos muito com essa atividade e gostamos principalmente da liberdade oferecida pelo projeto, que nos deu a oportunidade de entrar em contato com uma aplicação inesperadamente tangível e multifacetada do

conhecimento teórico e abstrato adquirido em aula. A inserção dos métodos numéricos em um modelo incrivelmente relevante para pesquisas interdisciplinares em neurociência computacional nos mostrou a importância de se obter uma fundação sólida nesses conceitos base.