# Conductance-based single-compartment models of cortical neurons

12/06/2023
WiM DRP
Waterloo, Ontario

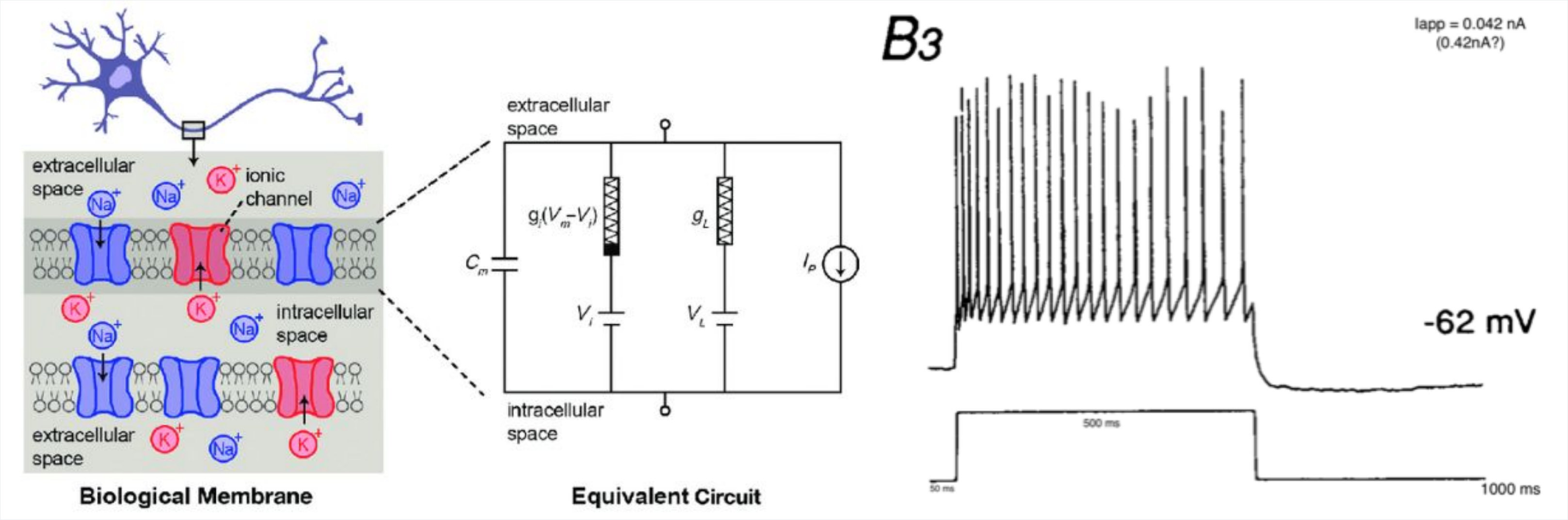
Erica Han - Applied Mathematics, 3rd year
Vinyga Kumararajah - Mathematics/Financial Analysis and Risk Management, 3rd year

# Outline

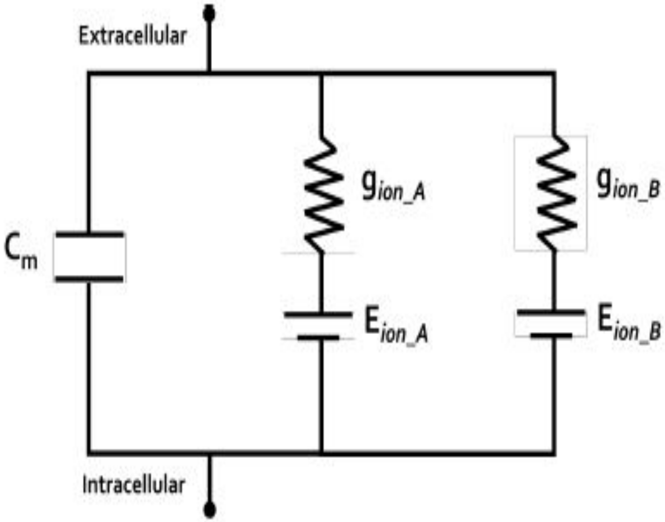
1. Project Objective
2. Introduction to Conductance-based Models
3. LTS Cell
4. NetPyne
5. Evolutionary algorithm
6. Next Steps

# Goal

Objective: Use NetPyne's parameter optimization tool to fit a conductance-based single neuron model to experimental data.



# Basics of Conductance-based models




$$I_C = C_m \frac{dV_m}{dt} \tag{1}$$

$$I_{ion} = g_{ion} \cdot (V_m - E_{ion}) \tag{2}$$

$$C_m \frac{dV_{m_i}}{dt} = -g_L(V_{m_i} - E_L) - \sum_j I_{ion}^{ji} - \sum_k I_{syn}^{ki} \tag{3}$$

$$g_{ion} = \bar{g}_{ion} m^N h^M \tag{4}$$

$$\frac{dm}{dt} = \frac{m_\infty(V) - m}{\tau_m(V)} \tag{5}$$

# Sample Model

$$C_m \frac{dV}{dt} = -I_{Leak} - I_{Na} - I_K - I_M \tag{6}$$

$$I_{Leak} = g_{Leak} \cdot (V - V_{Leak}) \tag{7}$$

$$I_{Na} = g_{Na} \cdot m_{Na}^3 \cdot h_{Na} \cdot (V - V_{Na}) \tag{8}$$

$$\alpha_{m_{Na}} = \frac{0.32 \cdot (-V - 50)}{\exp(\frac{-V-50}{4}) - 1} \tag{9}$$

$$\beta_{m_{Na}} = \frac{0.28 \cdot (V + 23)}{\exp(\frac{V+23}{5}) - 1} \tag{10}$$

$$\alpha_{h_{Na}} = 0.128 \cdot \exp\left(\frac{-V - 46}{18}\right) \tag{11}$$

$$\beta_{h_{Na}} = \frac{4}{1 + \exp\left(\frac{-V-23}{5}\right)} \tag{12}$$

$$I_K = g_K \cdot n_K^4 \cdot (V - V_K) \tag{13}$$

$$\alpha_{n_K} = \frac{0.032 \cdot (-V - 48)}{\exp(\frac{-V-48}{5}) - 1} \tag{14}$$

$$\beta_{n_K} = 0.5 \cdot \exp\left(\frac{-V - 53}{40}\right) \tag{15}$$

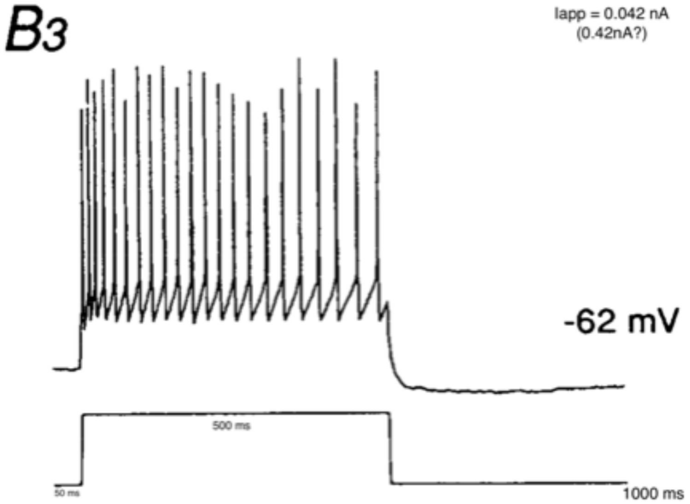
$$I_M = g_M \cdot m_M \cdot (V - V_K) \tag{16}$$

$$m_{M_\infty} = \frac{1}{1 + \exp\left(\frac{-V-35}{10}\right)} \tag{17}$$

$$\tau_{m_M} = \frac{1000}{3.3 \cdot \left(\exp\left(\frac{V+35}{20}\right) + \exp\left(\frac{-V-35}{20}\right)\right)} \tag{18}$$

# LTS Cell – Parameters & Target Model

$$C_m \frac{dV}{dt} = I_{hold} + I_{app} - I_h - \boxed{I_{Naf}} - I_{Nap} - I_{Kdr} - I_{Ka} - \boxed{I_{K2}} - \boxed{I_{Km}} - \boxed{I_{Kc}} - I_{Kahp} - \boxed{I_{CaT}} - I_{CaL} - I_{Leak} - I_{syn}$$



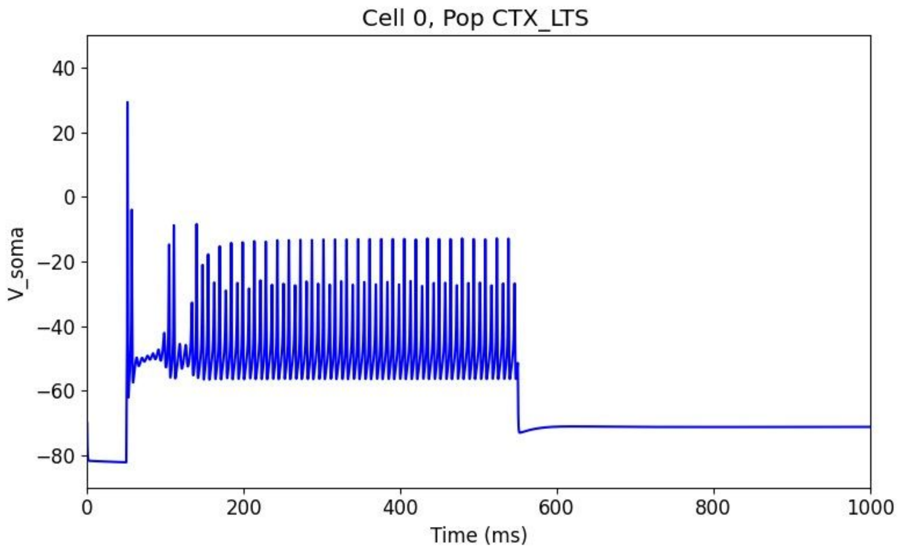

$I_{Naf}$ : $Na^{+}$ current

$I_{K2}$, $I_{Km}$, $I_{Kc}$ : $K^{+}$ currents

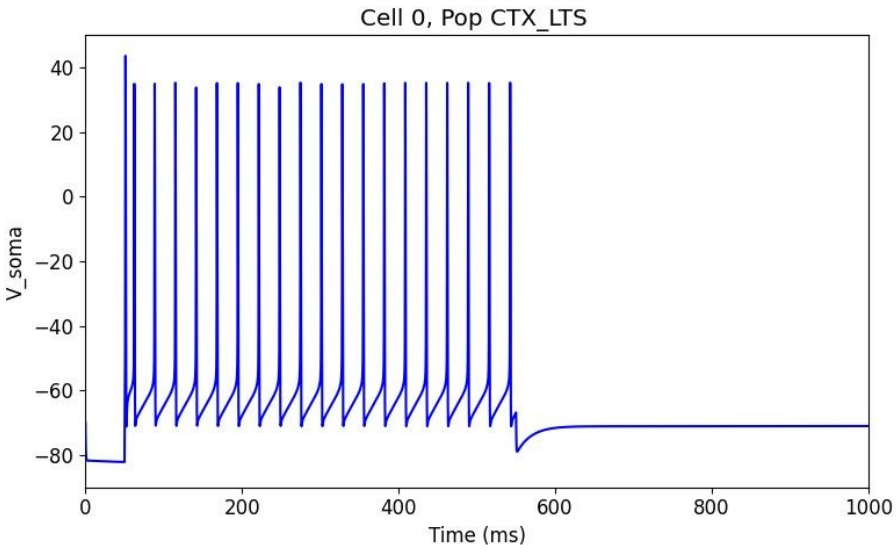
$I_{CaT}$ : $Ca^{2+}$ current

# Firing behaviour

Voltage against time: Default firing



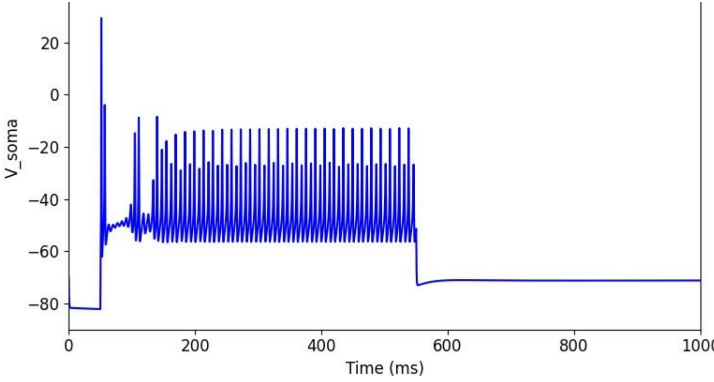

Voltage against time: Target firing



# Using NetPyne

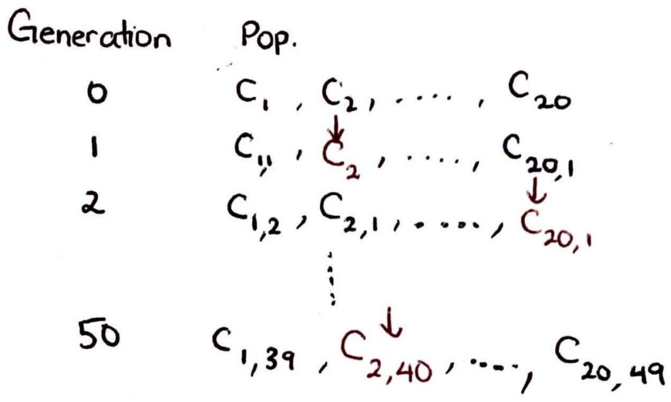
Single cell compartment:

```
netParams.cellParams['LTS'] = {
        'secs': {'soma':
                {'vinit':-70,
                'geom': {'diam': 16, 'L': 20, 'Ra': 250, 'cm':0.9},
                'mechs': {
                    'nafL':{'gnabar_nafL':0.1e-3}
                    ,'nap':{'gnabar_nap':0.15e-3}
                    ,'kdrL':{'gkbar_kdrL':100e-3}
                    ,'kcL':{'gkbar_kcL':150e-3}
                    ,'ka':{'gkbar_ka':1e-3}
                    ,'km':{'gkbar_km':75e-3}
                    ,'k2':{'gkbar_k2':45e-3}
                    ,'kahp':{'gkbar_kahp':0.1e-3}
                    ,'cal':{'gcabar_cal':0.1e-3}
                    ,'catL':{'gcabar_catL':25e-3}
                    ,'h':{'ghbar_h':0.1e-3,'ehcn':-43}
                    ,'pas':{'g':2.4e-3,'e':-70}
                    ,'cad':{'depth':4e-3,'taur':100}
                }}}}
```



# Evolutionary Algorithm (survival of the fittest)

- Purpose: Efficiently automating the process of fitting any model to a target
- Parameters of Algorithm:
  - Set bounds on model parameters
  - Fitness function: Evaluate whether it throws away the candidate or uses it




```
# min and max allowed value for each param optimized:
#    [gnabar_nafL, gkbar_kcL, gkbar_km, gkbar_k2, gcabar_catL]
minParamValues = [0.01e-3, 0.01e-3, 0.01e-3, 0.01e-3, 0.01e-3]
maxParamValues = [400e-3, 400e-3, 400e-3, 400e-3, 400e-3]
```

```
#call evolution iterator
final_pop = my_ec.evolve(generator=generate_netparams,  # assign design parameter generator to iterator parameter generator
                     evaluator=evaluate_netparams,     # assign fitness function to iterator evaluator
                     pop_size=20,                      # each generation of parameter sets will consist of 10 individuals
                     maximize=False,                   # best fitness corresponds to minimum value
                     bounder=ec.Bounder(minParamValues, maxParamValues), # boundaries for parameter set ([probability, weight, delay])
                     max_evaluations=100,               # evolutionary algorithm termination at 50 evaluations
                     num_selected=20,                  # number of generated parameter sets to be selected for next generation
                     mutation_rate=0.2,                # rate of mutation
                     num_inputs=5,                     # len([gnabar_nafL, gkbar_kcL, gkbar_km, gkbar_k2, gcabar_catL])
                     num_elites=1)                     # 1 existing individual will survive to next generation if it has better fitness
than an individual selected by the tournament selection
```

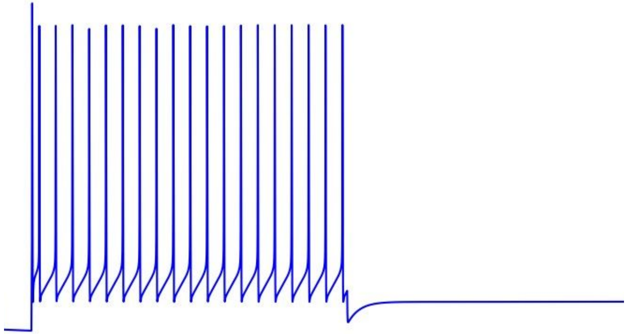
# Our Attempts - Fitness function based on target firing:

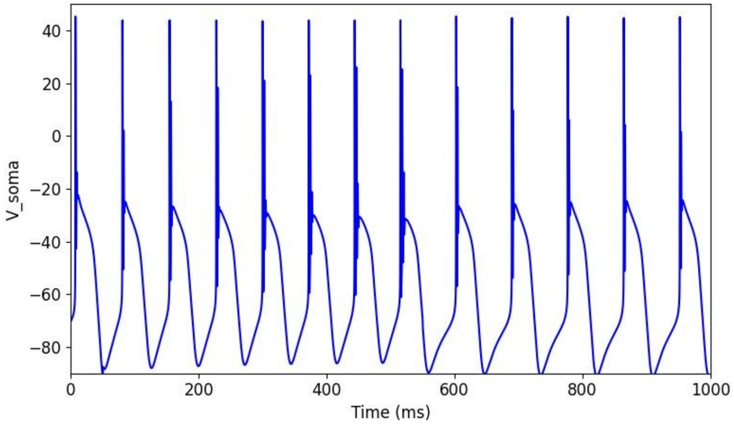
Fitness: Firing rate (number of spikes)

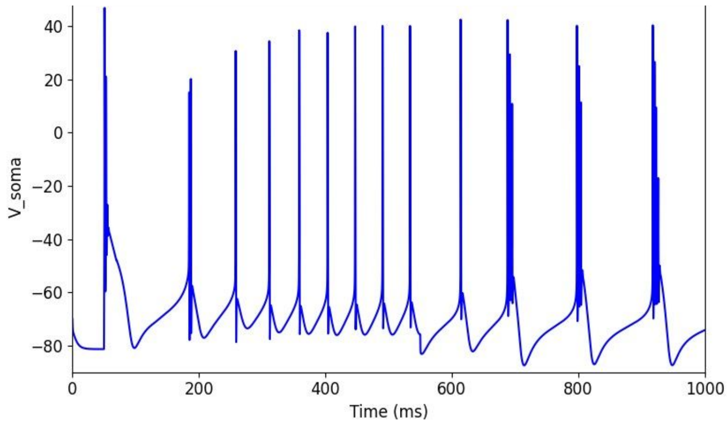
```
numSpikes = float(len(sim.simData['spkt']))
numCells = float(len(sim.net.cells))
duration = LTScell.simConfig.duration/1000.0
netFiring = numSpikes/numCells/duration

# calculate fitness for this candidate
fitness = abs(targetFiring - netFiring)  # min
```

Target firing = 20





# Our Attempts - Efel feature extraction:

- Efel: Python library - you give it a voltage trace and it **efficiently** extracts features
- Fitness: Set number of bursts and spikes per bursts = 0
- New fitness function and resulting voltage trace

```
# calculate fitness for this candidate
fitness = abs(targetFiring - netFiring) + num_bursts + spikes_per_burst ** 2
```

# Next Steps - Fitting the exact voltage traces:

- Target: Vector defining target voltage at each time
- Proposed fitness function:

$$\left|\frac{\overrightarrow{V_T} - \overrightarrow{V_O}}{\vec{\sigma}}\right|^2$$

- Hypothesis: This approach gives us the result that is closest in target behaviour. More importantly, it may help us find appropriate boundary values, which would narrow down our search for better fits. It may make the simulation inefficient or slower.

```
minParamValues = [0.01e-3, 0.01e-3, 0.01e-3, 0.01e-3, 0.01e-3]
maxParamValues = [400e-3, 400e-3, 400e-3, 400e-3, 400e-3]
```

# References

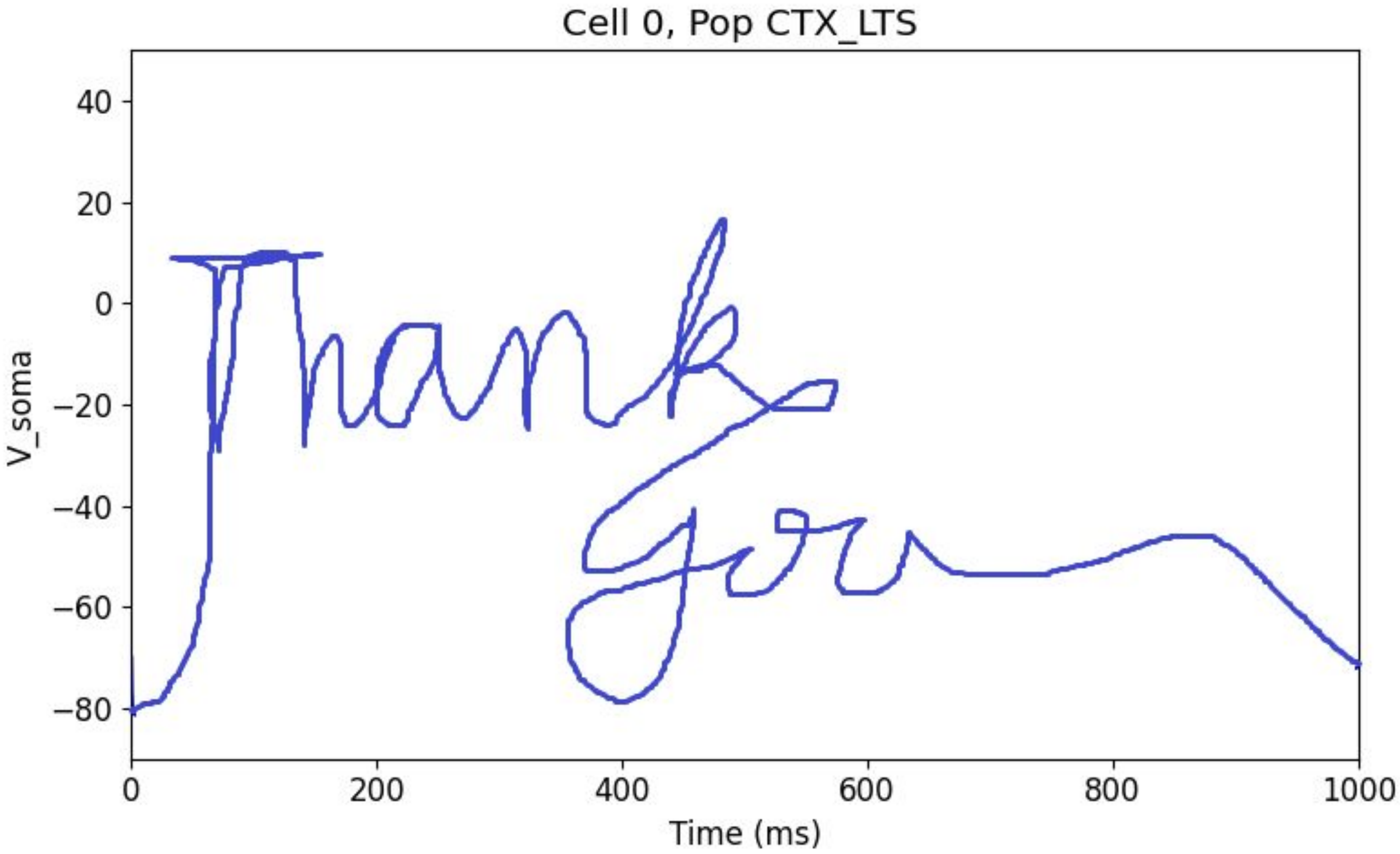

B. Ermentrout and D. H. Terman. *Mathematical foundations of neuroscience*, volume 35, pages 331–367. Springer, New York.

Henry Markram, Maria Toledo-Rodriguez, Yun Wang, Anirudh Gupta, Gilad Silberberg, and Caizhi Wu. Interneurons of the neocortical inhibitory system. *Nature reviews neuroscience*, 5(10):793–807, 2004.

Nicholas T Carnevale and Michael L Hines. *The NEURON book*. Cambridge University Press, 2006.

Salvador Dura-Bernal, Benjamin A Suter, Padraig Gleeson, Matteo Cantarelli, Adrian Quintana, Facundo Rodriguez, David J Kedziora, George L Chadderdon, Cliff C Kerr, Samuel A Neymotin, et al. NetPyne, a tool for data-driven multiscale modeling of brain circuits. *Elife*, 8:e44494, 2019.

Van Geit W, Gevaert M, Chindemi G, Rössert C, Courcol J-D, Muller EB, Schürmann F, Segev I and Markram H (2016) BluePyOpt: Leveraging Open Source Software and Cloud Infrastructure to Optimise Model Parameters in Neuroscience. Front. *Neuroinform*. 10:17. doi: 10.3389/fninf.2016.00017


Cell 0, Pop CTX_LTS
V_soma
Time (ms)
40
20
0
−20
−40
−60
−80
0
200
400
600
800
1000
Thank you