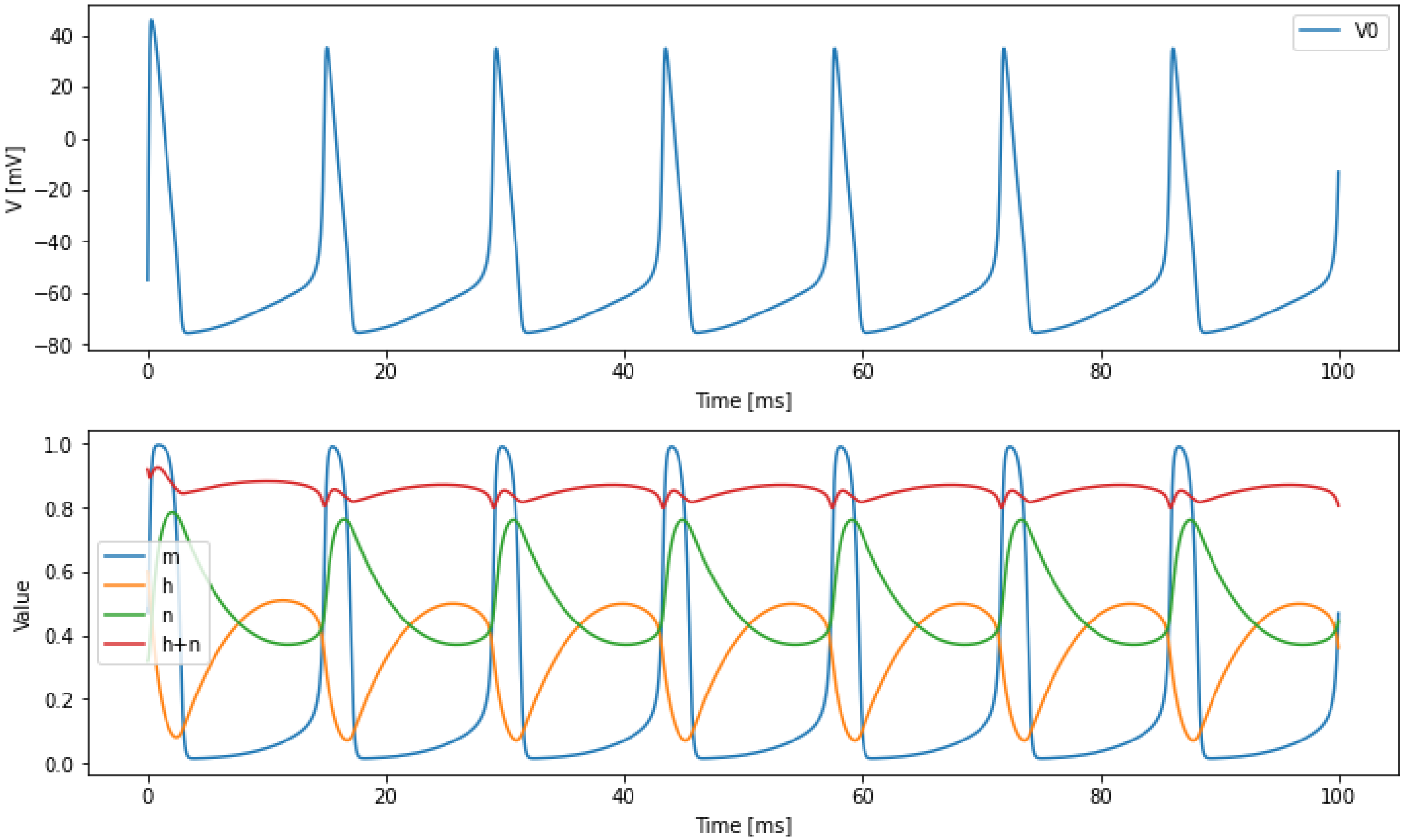
Single Neuron Model (3):

Reduced neuron model

Neuron models	HH neuron model
	Dynamics Analysis
	LIF neuron model
	Exponential IF model
Synapse models	AMPA/GABA/NMDA synapse
	Exponential synapse
Network Models	E/I balance network
	Continuous attractor network
	Working memory model
	Decision making model

Four-variable HH model reduced to two-variable model



- m behaves instantaneously.

$$m = m_\infty(V) = \frac{\alpha_m(V)}{\alpha_m(V) + \beta_m(V)}$$

- $h + n = 0.855$

A two-variable neuron model

$$C\frac{dV}{dt} = -(\bar{g}_{Na} m_\infty^3 h(V - E_{Na}) + \bar{g}_K (0.855 - h)^4 (V - E_K) + g_{leak}(V - E_{leak})) + I(t)$$

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

$$m_\infty = \frac{\alpha_m(V)}{\alpha_m(V) + \beta_m(V)}$$

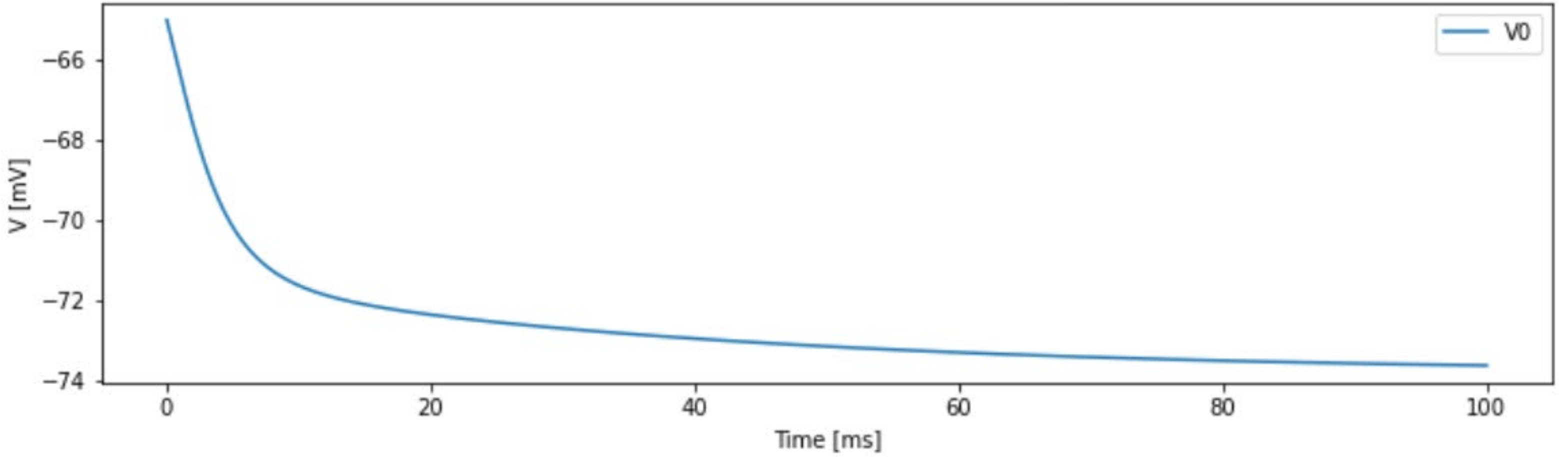
$$\alpha_m(V) = \frac{0.1(V + 40)}{1 - \exp(\frac{-(V+40)}{10})}$$

$$\beta_m(V) = 4.0 \exp(\frac{-(V + 65)}{18})$$

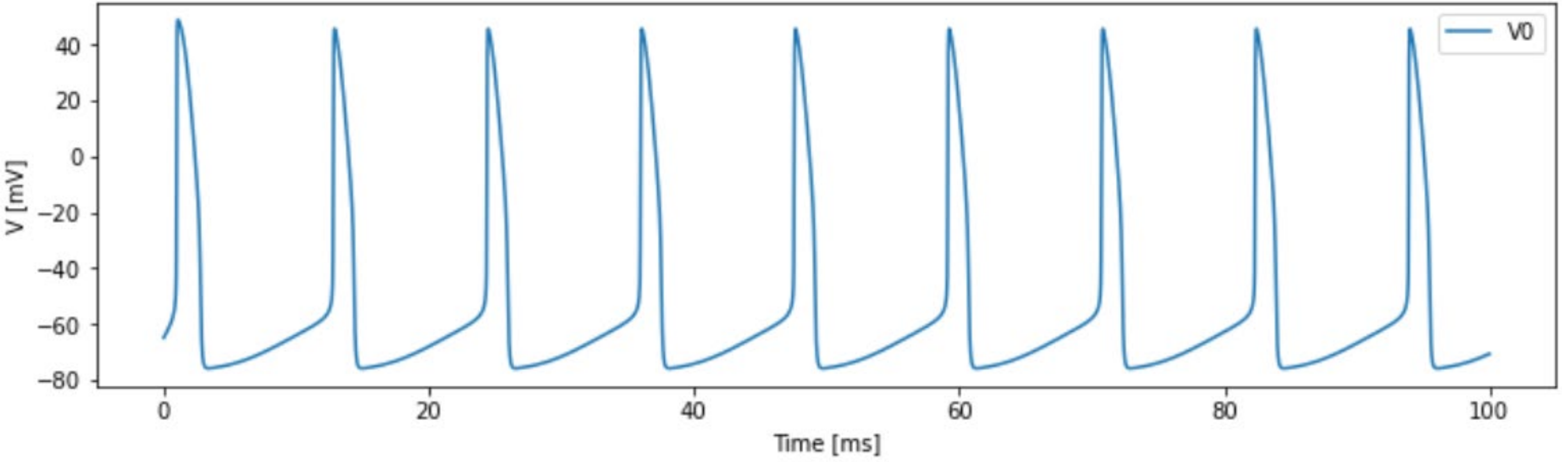
$$\alpha_h(V) = 0.07 \exp(\frac{-(V + 65)}{20})$$

$$\beta_h(V) = \frac{1}{1 + \exp(\frac{-(V+35)}{10})}$$

I = 0.



I = 5.

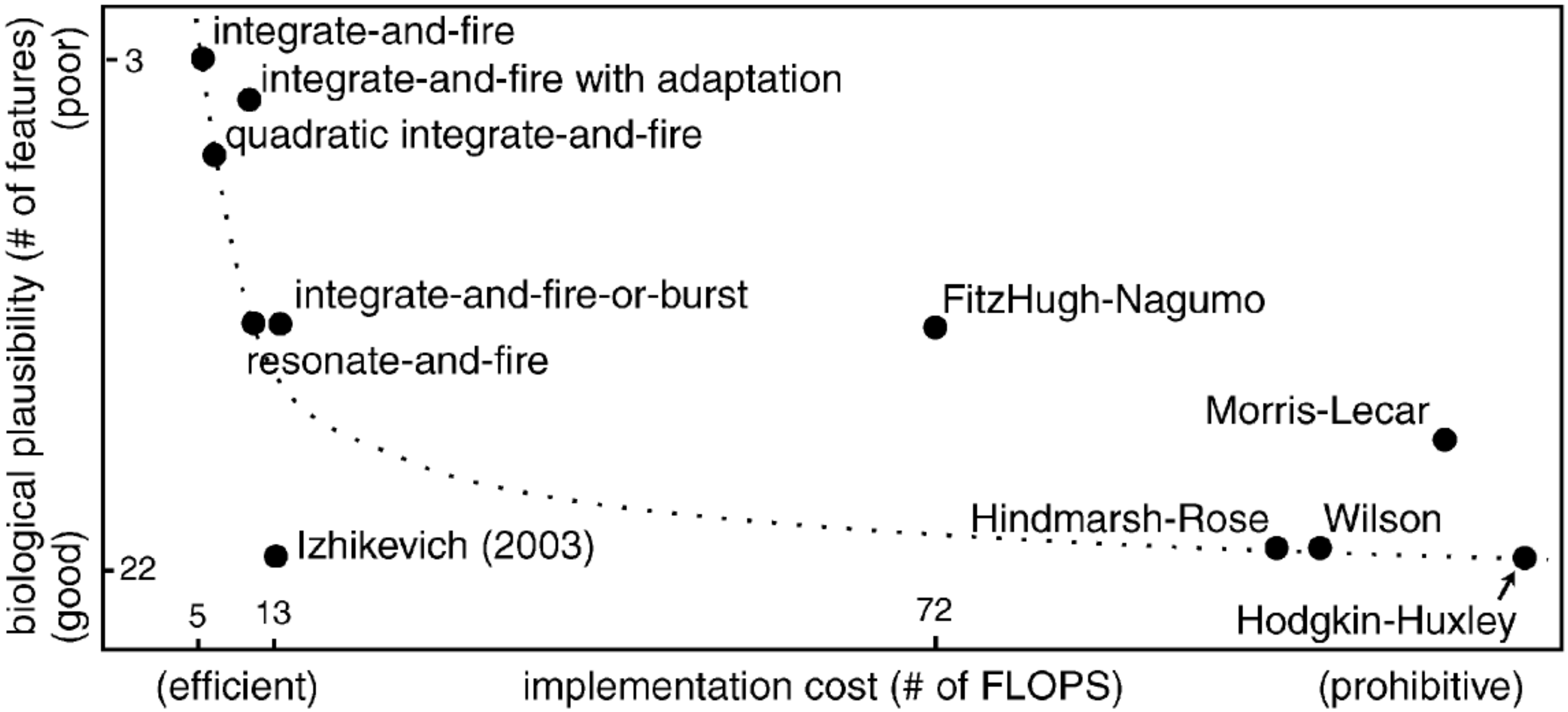


What determines the steady state of the model?

Methods for Dynamics Analysis

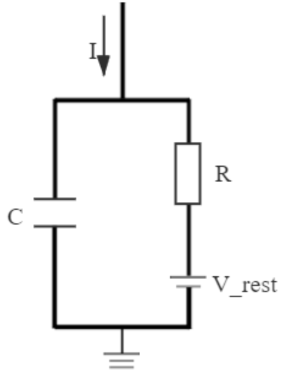
……

Trade-off between biological plausibility and implementation cost



Izhikevich, Eugene M. "Simple model of spiking neurons." IEEE Transactions on neural networks 14.6 (2003): 1569-1572.

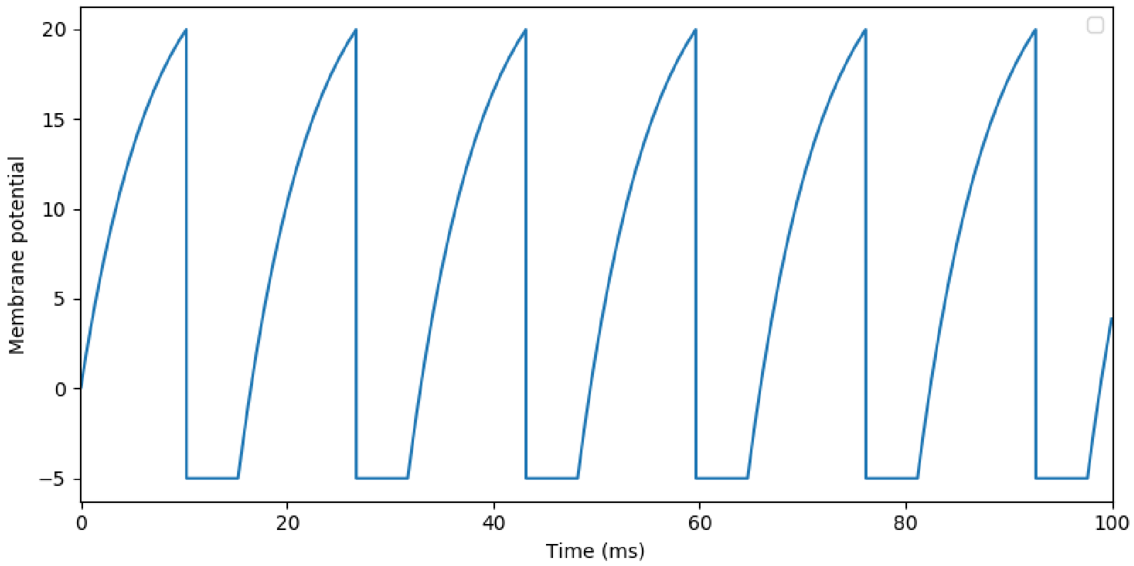
Leaky integrate-and-fire model (LIF)



$$\tau \frac{dV}{dt} = -\left(V - V_{rest}\right) + R * I(t)$$

If $V > V_{th}, V \rightarrow V_{reset}$, last for τ_{ref} ms.

Integrate + Reset



Implementation of the LIF model with BrainPy

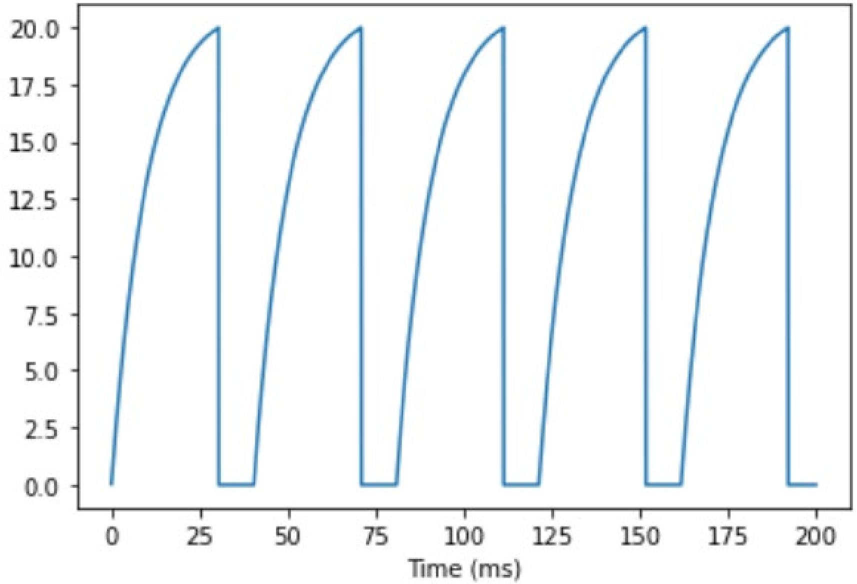
```python
class LIF(bp.NeuGroup):
  target_backend = ['numpy', 'numba']

  @staticmethod
  @bp.odeint
  def integral(V, t, Iext, V_rest, R, tau):
    dvdt = (-V + V_rest + R * Iext) / tau
    return dvdt

  def __init__(self, size, t_ref=1., V_rest=0., V_reset=0.,
               V_th=20., R=1., tau=10., **kwargs):
    super(LIF, self).__init__(size=size, **kwargs)

    # parameters
    self.V_rest = V_rest
    self.V_reset = V_reset
    self.V_th = V_th
    self.R = R
    self.tau = tau
    self.t_ref = t_ref

    # variables
    self.t_last_spike = bp.ops.ones(self.num) * -1e7
    self.refractory = bp.ops.zeros(self.num, dtype=bool)
    self.spike = bp.ops.zeros(self.num, dtype=bool)
    self.V = bp.ops.ones(self.num) * V_rest
    self.input = bp.ops.zeros(self.num)
```

```python
  def update(self, _t):
    for i in range(self.num):
      spike = False
      refractory = (_t - self.t_last_spike[i]) <= self.t_ref
      if not refractory:
        V = self.integral(self.V[i], _t, self.input[i],
                          self.V_rest, self.R, self.tau)
        spike = (V >= self.V_th)
        if spike:
          V = self.V_reset
          self.t_last_spike[i] = _t
          refractory = True
        self.V[i] = V
      self.spike[i] = spike
      self.refractory[i] = refractory
      self.input[i] = 0.
```

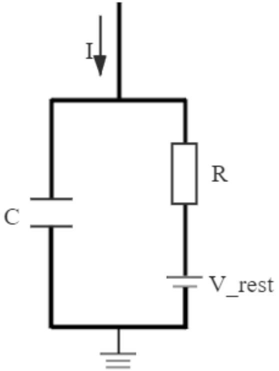
```
lif = LIF(1, t_ref=10., monitors=['V'])
lif.run(200, inputs=('input', 21), report=True)
```

```
Compilation used 0.1918 s.
Start running ...
Run 10.0% used 0.008 s.
Run 20.0% used 0.016 s.
Run 30.0% used 0.024 s.
Run 40.0% used 0.033 s.
Run 50.0% used 0.041 s.
Run 60.0% used 0.049 s.
Run 70.0% used 0.057 s.
Run 80.0% used 0.065 s.
Run 90.0% used 0.073 s.
Run 100.0% used 0.081 s.
Simulation is done in 0.081 s.
```

```
bp.visualize.line_plot(lif.mon.ts, lif.mon.V)
```



Analytical solution of the LIF model



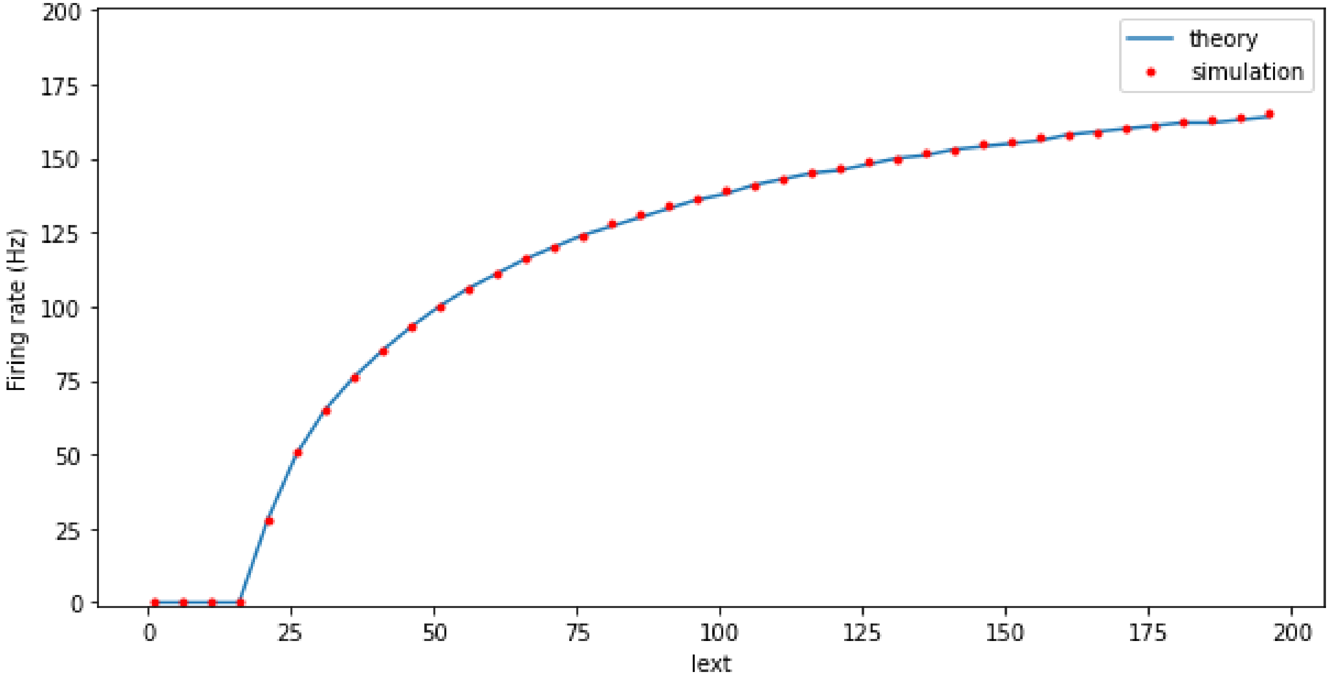
$$\tau \frac{dV}{dt} = -(V - V_{rest}) + R * I(t)$$

If $V > V_{th}, V \rightarrow V_{reset}$, last for τ_{ref} ms.

Given constant input I_c, and the initial state $V(t_0) = V_{rest}$,

- The solution: $V(t) = V_{rest} + RI_c(1 - e^{-\frac{t-t_0}{\tau}})$
- The time to fire: $T = \tau \ln\left[1 - \frac{V_{th} - V_{rest}}{RI_c}\right]$
- The firing rate: $f = \frac{1}{T + \tau_{ref}}$

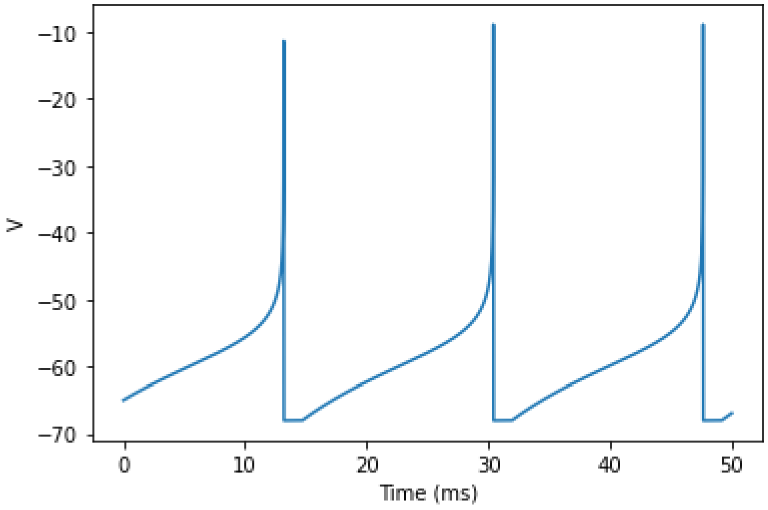
F-I curve of the LIF model



Exponential integrate-and-fire model (EIF)

Explicitly modeling the action potential of a neuron.

$$\tau \frac{dv}{dt} = -(V - V_{rest}) + \Delta T e^{\frac{V - V_T}{\Delta T}} + RI(t)$$



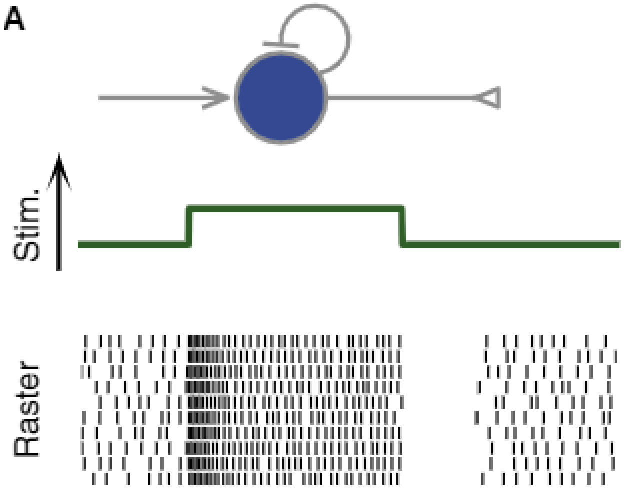
- 在指数项中V_T是动作电位初始化的临界值，在其下V缓慢增长，其上V迅速增长。
- ΔT是ExpIF模型中动作电位的斜率。当$\Delta T \to 0$时，ExpIF模型中动作电位的形状将趋近于$V_{th} = V_T$的LIF模型。

Adaptive Exponential Integrate-and-Fire Model (AdExIF)

Explicitly modeling the neuron adaptation.

$$\tau_v \frac{dv}{dt} = -(V - V_{rest}) + \Delta T e^{\frac{V - V_T}{\Delta T}} - Rw + RI(t)$$

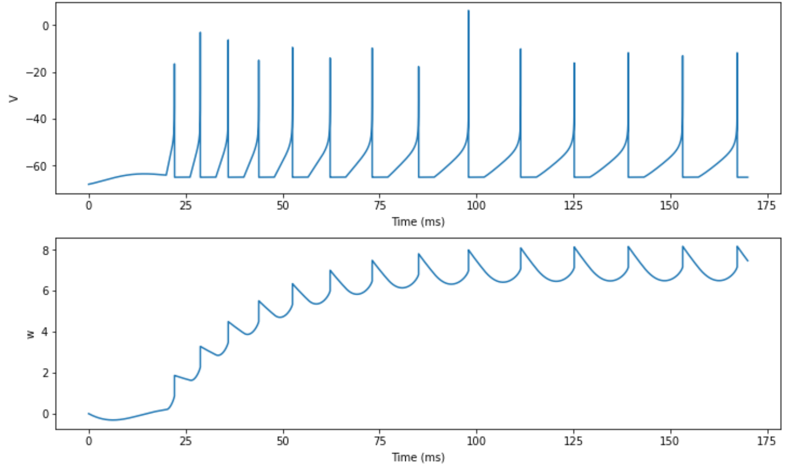
$$\tau_w \frac{dw}{dt} = a(V - V_{rest}) - w + b\tau_w \sum \delta(t - t^f)$$



Neuron Adaptation

(Jan Benda, Current Biology, 2020)

- a描述了权重变量w对V的下阈值波动的敏感性，b表示w在一次发放后的增长值。
- 给神经元一个恒定输入，在连续数次发放后，w的值将会上升到一个高点，减慢V的增长速度，从而降低神经元的发放率。



Exercise

1. Implement Reduced HH model
2. Make phase plane and bifurcation analysis for the reduced HH model.
3. Implement LIF model
4. Implement EIF model
5. Implement AdExIF model