Point models of single neurons: the E-GLIF

Alice Geminiani

Department of Brain and Behavioral Sciences
University of Pavia

RisingNet internal workshop
10th December 2020

alice.geminiani@unipv.it

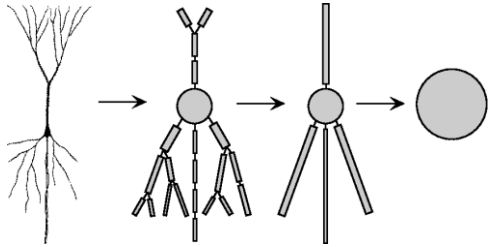
UNIVERSITÀ DI PAVIA

Outline

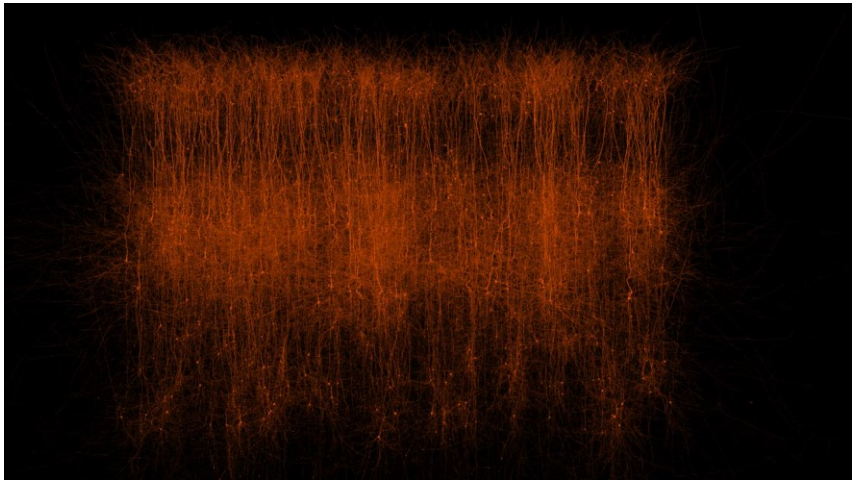
- Modelling single neurons
- E-GLIF
- Cerebellar use case
- A deep dive into optimization
- A deep dive into simulation
- RisingNet use cases

Modelling single neurons

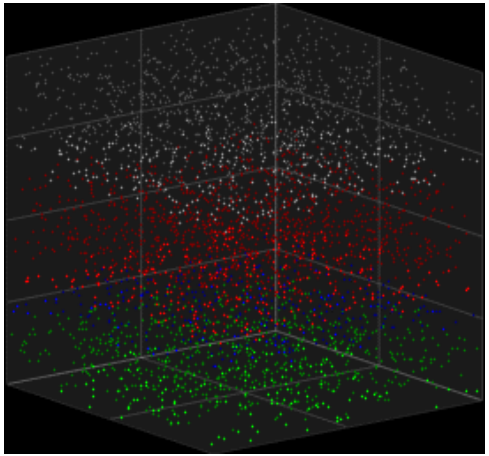
- Different levels of ***morphological*** detail: from multi-compartment to point neuron models



Multi-compartment neuron models describe the activity of each neuron element (dendrites, axons, …) taking into account morphological features. Example from the neo-cortex microcircuit [Markram et al., *Cell Reports*, 2015]:

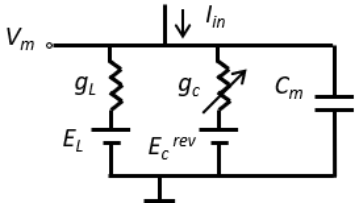


Point neuron models describe the activity of neurons as collapsed in a single point, neglecting compartment differences and morphological features. They represent more the computational properties of neurons, than the electrical activity and its spatial distribution

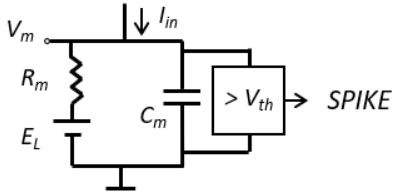


Modelling single neurons

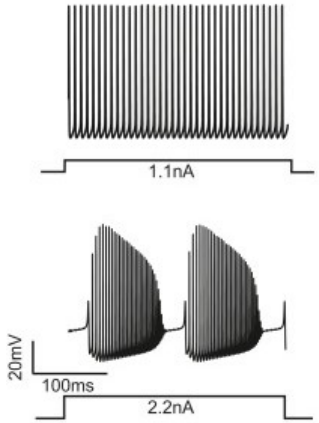
- Different levels of *electrical* detail: Hodgkin-Huxley (HH) and Leaky Integrate-and-Fire (LIF)

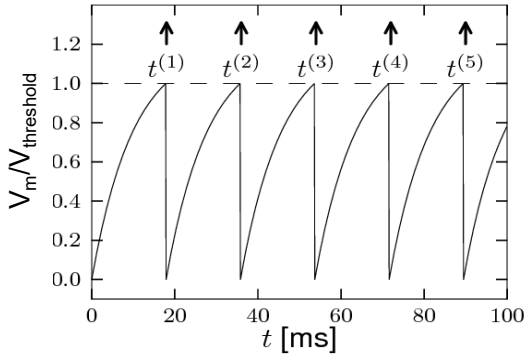


HH: membrane potential V_m computed considering the resting potential (E_L) and the contribution of each membrane **ion channel** (represented by the conductance g_c and reversal potential E_c^{rev}).



Only **passive** membrane properties are considered (capacitance C_m and resistance R_m). The output is a spike train, corresponding to time instants of threshold overcoming



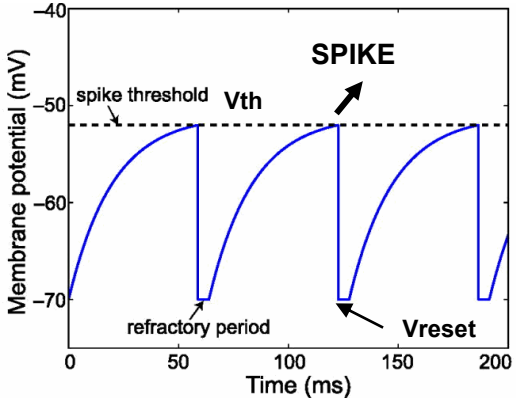


Modelling single neurons: the Leaky Integrate and Fire neuron

In the Leaky Integrate-and-Fire (LIF) neuron, the subthreshold dynamics of the membrane potential is modelled through a single passive term:

Membrane potential dynamics → $\tau_m \frac{dV_m(t)}{dt} = -(V_m(t) - E_L) + R_m \cdot I_{in}(t)$

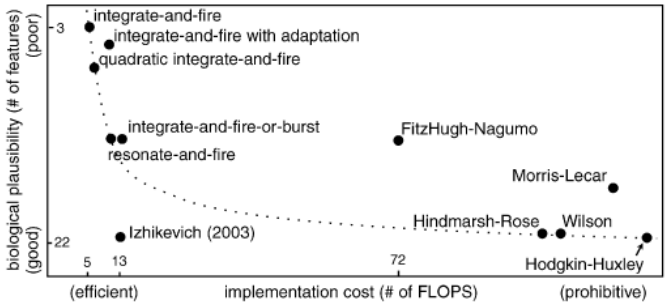
Spike condition → $If\ V_m > V_{th}, then\ V_m = V_{reset}$



- τ_m is the membrane time constant ($\tau_m = R_m \cdot C_m$, where R_m and C_m are the membrane resistance and capacitance, respectively)
- E_L is the resting potential
- I_{in} is the input current.

- Action potentials are approximated as **single spike instants**: whenever V_m reaches a firing threshold V_{th}, the membrane potential is reset to a fixed value V_{reset} and goes back to firing after t_{ref}

Modelling single neurons: biological plausibility vs computational load



[Izhikevich, *IEEE Trans Neural Networks*, 2003]

Compromise between biological plausibility and implementation cost:

- HH multi-compartment models with morphology representation ✔ biological plausibility ✗ computational load
- LIF point neuron models ✗ biological plausibility ✔ computational load
- Multi-dimensional LIF models:
  - Izhikevich (non linear)
  - Adaptive Exponential Leaky Integrate and Fire (LIF) model (non linear)
  - Generalized LIF (linear)
- Fitting with experimental traces for OPTIMIZATION – also optimization has a cost!

Multi-dimensional LIF models

Izhikevich model:

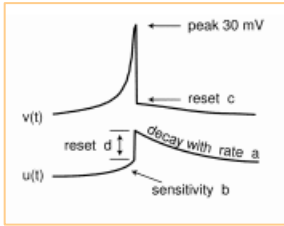
$$\begin{cases} V'(t) = 0{,}04 \cdot V^2(t) + 5 \cdot V + 150 - u(t) + I \\ u'(t) = a \cdot (b \cdot V(t) - u(t)) \end{cases}$$

→ ***Membrane Potential***

→ ***Membrane Recovery variable*** *(K+ activation and Na+ inactivation)* → for adaptation

If $V(t) \geq 30mV \rightarrow$ *SPIKE:*

$$\begin{cases} V(t+1) = c \\ u(t+1) = u(t) + d \end{cases}$$



[Izhikevich, *IEEE Trans Neural Networks*, 2003]

AdEx model:

$$\begin{cases} C_m \cdot V'(t) = -g_L \cdot (V(t) - E_L) + g_L \cdot \Delta_T \cdot e^{\frac{V(t) - V_{th}}{\Delta_T}} - w(t) + I \\ \tau_w \cdot w'(t) = a \cdot (V(t) - E_L) - w(t) \end{cases}$$

→ ***Membrane Potential***

→ ***Adaptive current***

If $V(t) \geq V_{th} \rightarrow$ *SPIKE:*

$$\begin{cases} V(t+1) = V_r \\ w(t+1) = w(t) + b \end{cases}$$

[Brette and Gerstner, *J Neurophysiol*, 2005]

Properties:

- multiple electroresponsive properties based on parameter values
- replacement of the strict voltage threshold by a more realistic smooth spike initiation zone.
- subthreshold resonances or adaptation.

Multi-dimensional LIF models: Generalized LIF neuron

The model:

$$\begin{cases} C_m \cdot V'(t) = -g_L \cdot (V(t) - E_L) + \sum_j I_j(t) + I_e \\ I'(t) = -k_j \cdot I_j(t) \\ V'_{th} = a \cdot (V(t) - E_L) - b \cdot (V_{th}(t) - V_\infty) \end{cases}$$

→ ***Membrane Potential***

→ ***Spike-triggered current***

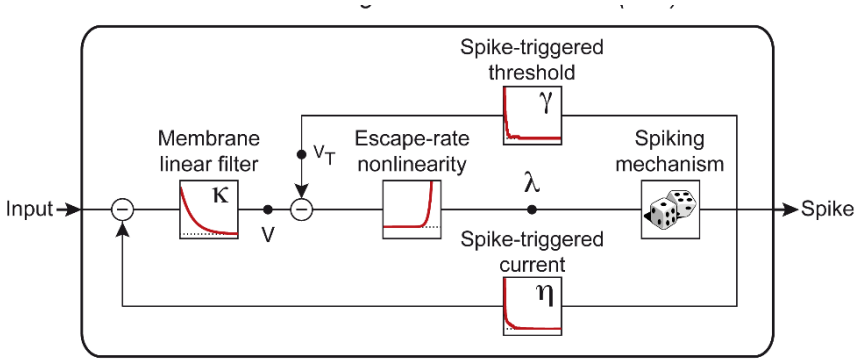
→ ***Spike-triggered threshold***

[Pozzorini et al., *Plos Comp Biol*, 2015; Mihalas and Niebur, *Neural Comput.*, 2009]

If $V(t) \geq V_{th}$ → *SPIKE:*

$$\begin{cases} V(t+1) = V_r \\ I_j(t+1) = R_j \cdot I_j(t) + A_j \\ V_{th}(t+1) = \max(V_\infty, V_{th}(t)) \end{cases}$$

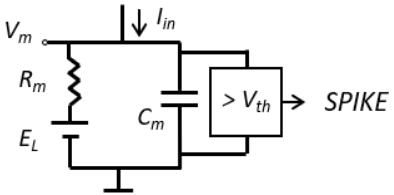
Block representation: The **membrane** acts as a low-pass filter κ(t) on the input current I(t) to produce the modeled potential V(t). The **exponential nonlinearity** (escape-rate) transforms this voltage into an instantaneous firing intensity λ(t), according to which spikes are generated. Each time a spike is emitted, both a **current η(t) and a movement of the firing threshold γ(t) are triggered**.

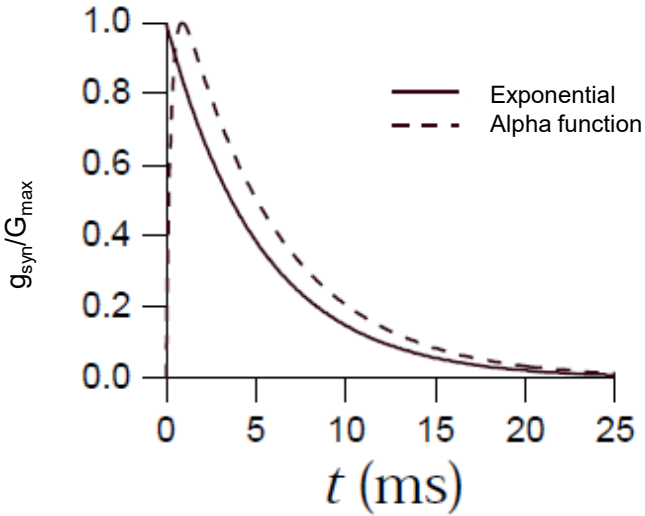


Synaptic inputs

An additional current, I_{syn}, is provided as an input to the model membrane potential to model the contribution of input spikes (conductance-based model):

$$I_{syn}(t) = g_{syn}(t) \cdot (V_m(t) - E_L)$$

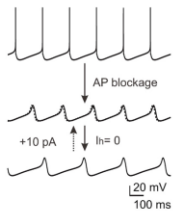


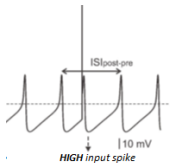


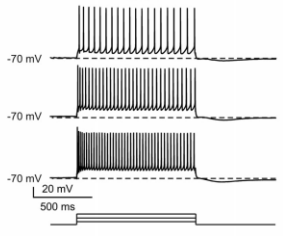
Following a spike, the synaptic conductance g_{syn} can change as an exponential function (direct decay) or as an alpha function (rise and decay)

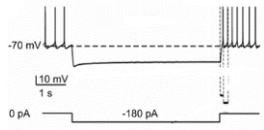
Olivocerebellar single neuron dynamics

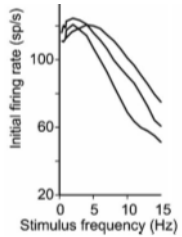
CEREBELLAR CELLS	Auto-rhythm	Sub-Threshold Oscillations	Phase reset	Depolarization-induced burst	Adaptation (SFA)	Post-inhibitory rebound burst	Resonance
Golgi Cell	✓ 12 Hz	✓	✓	✓	✓	✓	✓ (ϑ band)
Granule Cell	-	✓	-	✓	-	-	✓ (ϑ band)
Purkinje Cell	✓ 40-80 Hz	-	-	✓	✓	✓	-
Deep Cerebellar Nuclei	✓ 10-30 Hz	-	-	✓	✓	✓	-
Inferior Olive	-	✓ 1-4 Hz	✓	n.a.		(✓) spike	✓







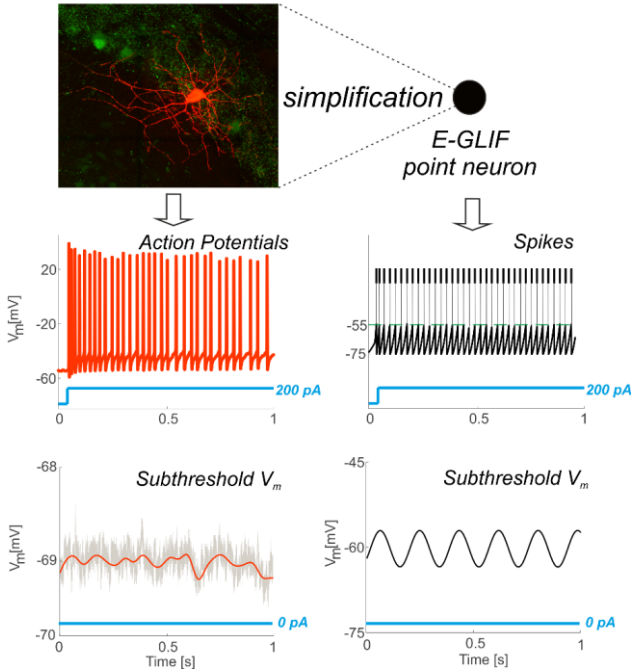




Adapted from Solinas et al, Front Cell Neurosci, 2007a,b

Towards a unified point neuron model for cerebellar neurons

- Aim: a model able to reproduce all the cerebellar electroresponsive mechanisms, while keeping:
  - **Neurophysiological realism** (elements in the model ↔ biophysical mechanisms)
  - **Low computational load** (→ linear and analytically solvable, to increase simulation step without loosing precision within large-scale Spiking Neural Networks - SNNs)
  - **Generalized features** (not fitting on single traces)
  - Different **sets of parameters** for different cells, reproducing **all** the electrophysiological properties of each population, i.e. **spike patterns** more than sub/supra-threshold mechanisms (since within SNN)



Extended-Generalized LIF neuron model (E-GLIF)

- State variables:

Membrane potential

$$\begin{cases} V_m'(t) = \frac{1}{C_m}\left(\frac{C_m}{\tau_m}(V_m(t) - E_L) + I_{stim} + I_e + I_{dep}(t) - I_{adap}(t)\right) \\ I_{adap}'(t) = k_{adap}(V_m(t) - E_L) - k_2 I_{adap}(t) \\ I_{dep}'(t) = -k_1 I_{dep}(t) \end{cases}$$

Adaptive current

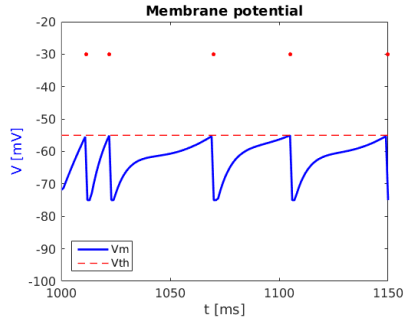
Spike-triggered depolarizing current

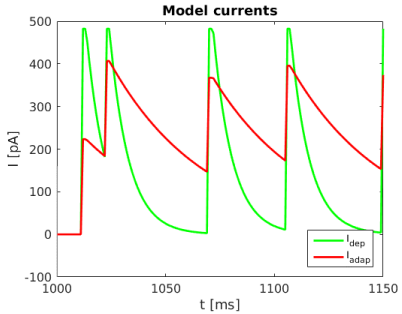
- Spike generation at t_{spk}:

$$\begin{cases} t_{spk} \notin \Delta t_{ref} \\ rng < (1 - e^{-\lambda(t_{spk})t_{spk}}) \end{cases}$$

→ Refractory period

→ Stochasticity





- Update rules:

$$\begin{cases} V_m(t_{spk}) \leftarrow V_r \\ I_{dep}(t_{spk}) \leftarrow A_1 \\ I_{adap}(t_{spk}) \leftarrow I_{adap}(t_{spk} - 1) + A_2 \end{cases}$$

[Geminiani et al, *Front Neuroinform*, 2018]

Extended-Generalized LIF neuron model (E-GLIF)

Parameters:

I_{stim} = external stimulation current;
C_m = membrane capacitance;
τ_m = membrane time constant;
E_L = resting potential;
I_e = endogenous current;
k_{adap}, k_2 = adaptation constants;
k_1 = I_{dep} decay rate;
V_{th} = threshold potential;

λ_0, τ_V = escape rate parameters;

t_{spk}^{+} = time instant immediately following the spike time t_{spk}

V_r = reset potential;

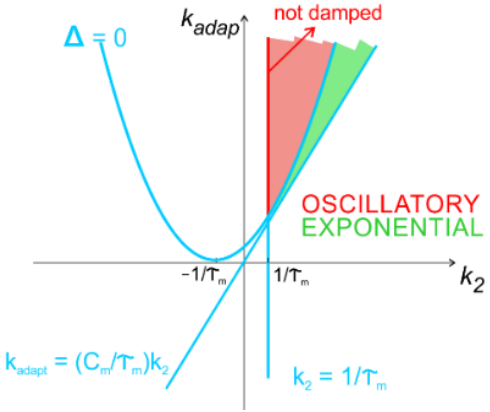
A_2, A_1 = model currents update constants.

Biological quantities/parameters

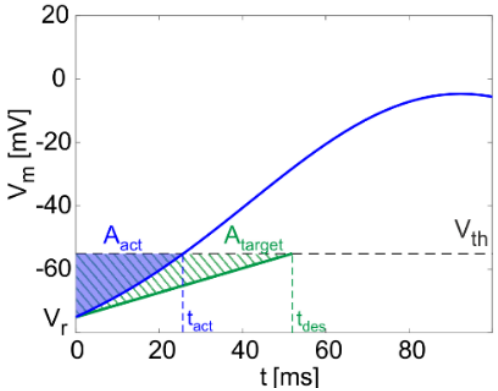
Artificial parameters

[Geminiani et al, *Front Neuroinform*, 2018]

Optimization



Different solution regimes depending on parameters k_2 and k_{adap}

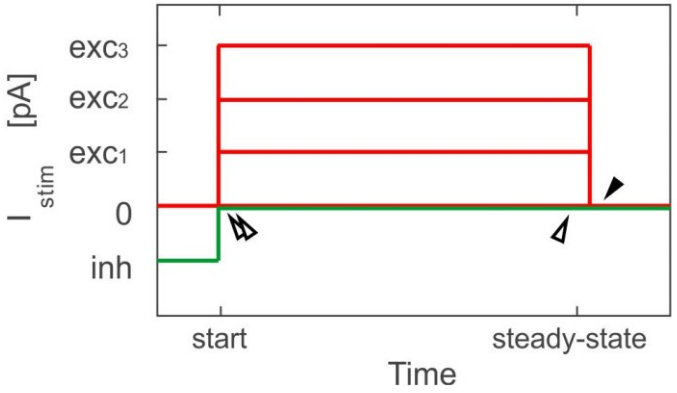


Model analytical solution (blue)
vs
simplified target area (green)

Evaluation of error on spike times during different stimulus (I_{stim}) conditions
+
Mathematical and electrophysiological constraints

Sequential Quadratic Programming optimization algorithm

Cost function



Input current step	Expected output	Corresponding property
I_{stim} = *zero_stim*	Firing at *tonic_freq*	Autorhythm
I_{stim} = *exc1* > *0*	Firing at *freq1* and adaptation with *gain1*	*f*-I_{stim} relationship Depolarization-induced excitation Spike-frequency adaptation
I_{stim} = *exc2* > *exc1*	Firing at *freq2* and adaptation with *gain2*	
I_{stim} = *exc3* > *exc2*	Firing at *freq3* and adaptation with *gain3*	
I_{stim} = *inh* <*0*	Silence period during hyperpolarization and return to spiking with at least 2-spike burst (faster than *tonic_freq*) when hyperpolarization stops.	Post-inhibitory rebound bursting

E-GLIF – implementation, optimization and validation

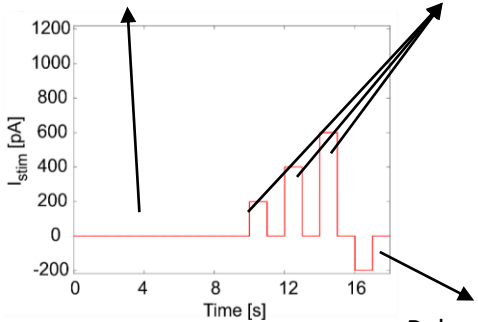
Model DEVELOPMENT

Cell-specific model CHARACTERIZATION
- Analytical SOLUTION
- CONSTRAINTS

Model OPTIMIZATION (spike time error minimization - *literature*)

Model VALIDATION against *electrophysiological in vitro* data

- Autorhythm frequency (*zero_stim*)
- SubThreshold Oscillations
- Starting frequency, f_{in}, and ending frequency, f_{1s}, during three excitatory phases ($exc_{1,2,3}$)
- Adaptation ($gain_{1,2,3}$)



- Rebound burst

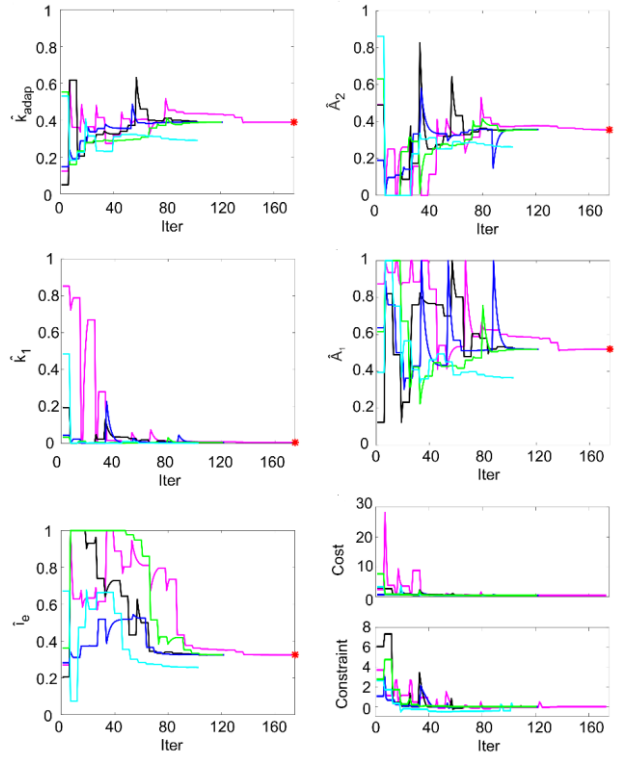
[Geminiani et al, *Front Neuroinform*, 2018]

Optimization – cerebellar Golgi cells

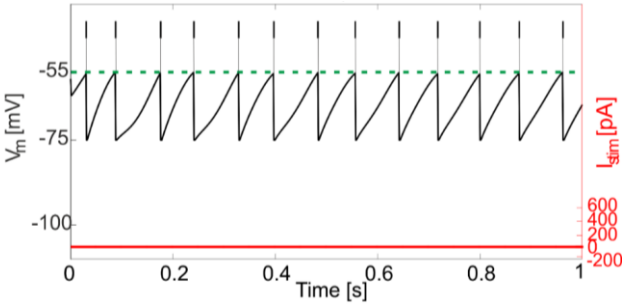
Optimized input-output relationships

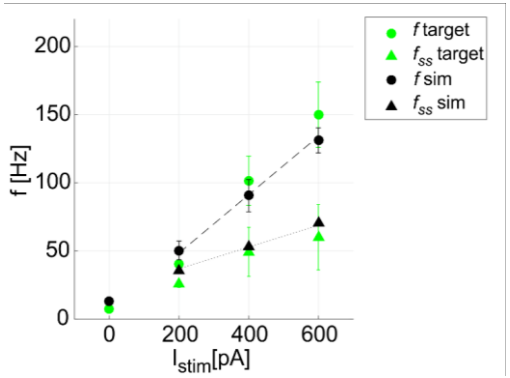
Input current step (I_{stim})	Expected output (mean ± SD)	Desired spike times	Corresponding properties
zero_stim = 0 pA	*tonic_freq* = 8 ± 1 Hz	$\Delta t_{1\_des}^{(zero\_stim)} = \Delta t_{2\_des}^{(zero\_stim)} = \Delta t_{ss\_des}^{(zero\_stim)}$ = 125 ms (mean)	Autorhythm
exc1 = 200 pA	*freq1* = 40 ± 2 Hz	$\Delta t_{1\_des}^{(exc1)} = \Delta t_{2\_des}^{(exc1)}$ = 25 ms (mean)	*f-I*$_{stim}$ relationship
	adaptation *gain1* = 0.7	$\Delta t_{ss\_des}^{(exc1)}$ = 35 ms (mean)	
exc2 = 400 pA	*freq2* = 100 ± 15 Hz	$\Delta t_{1\_des}^{(exc2)} = \Delta t_{2\_des}^{(exc2)}$ = 10 ms (mean)	Depolarization-induc bursting
	adaptation *gain2* = 0.5	$\Delta t_{ss\_des}^{(exc2)}$ = 20 ms (mean)	Spike-frequency adaptation
exc3 = 600 pA	*freq3* = 150 ± 20 Hz	$\Delta t_{1\_des}^{(exc3)} = \Delta t_{2\_des}^{(exc3)}$ = 6.6 ms (mean)	
	adaptation *gain3* = 0.4	$\Delta t_{ss\_des}^{(exc3)}$ = 16 ms (mean)	
0 pA after ***inh = -200 pA***	Latency of 1st spike lower than 0.5·(1/*tonic_freq*)	$\Delta t_{lat\_reb\_des}^{(inh)}$ < 62.5 ms (uniform distribution)	Post-inhibitory rel bursting
	rebound_freq > 2·*tonic_freq*	$\Delta t_{1st\_reb\_des}$ < 62.5 ms (uniform distribution)	

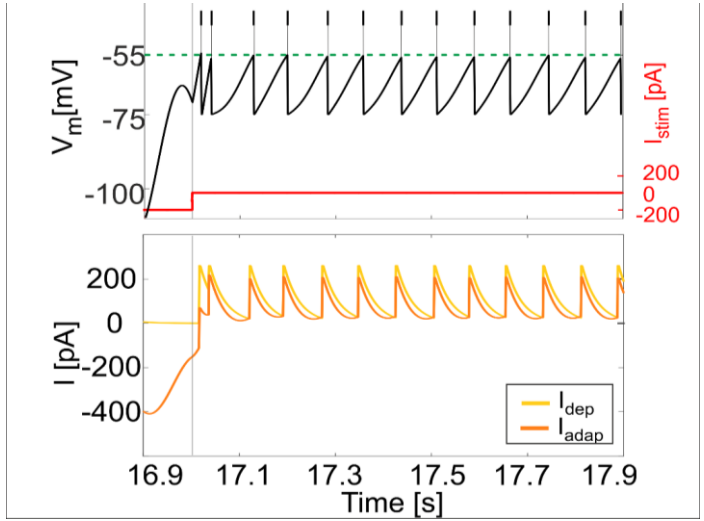
Parameters, cost and constraints along 5 optimization runs



E-GLIF – cerebellar Golgi cells







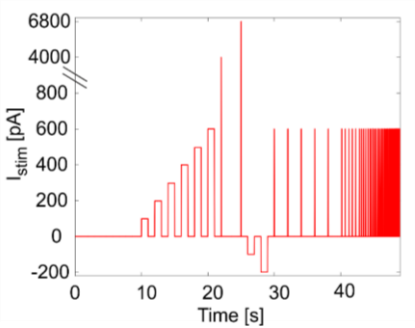
- Autorhythm at 12 Hz
- SubThreshold Oscillations at 5 Hz
- Linear I_{stim} – frequency relationship
- Spike Frequency Adaptation
- Post-inhibitory rebound burst

Model OPTIMIZATION

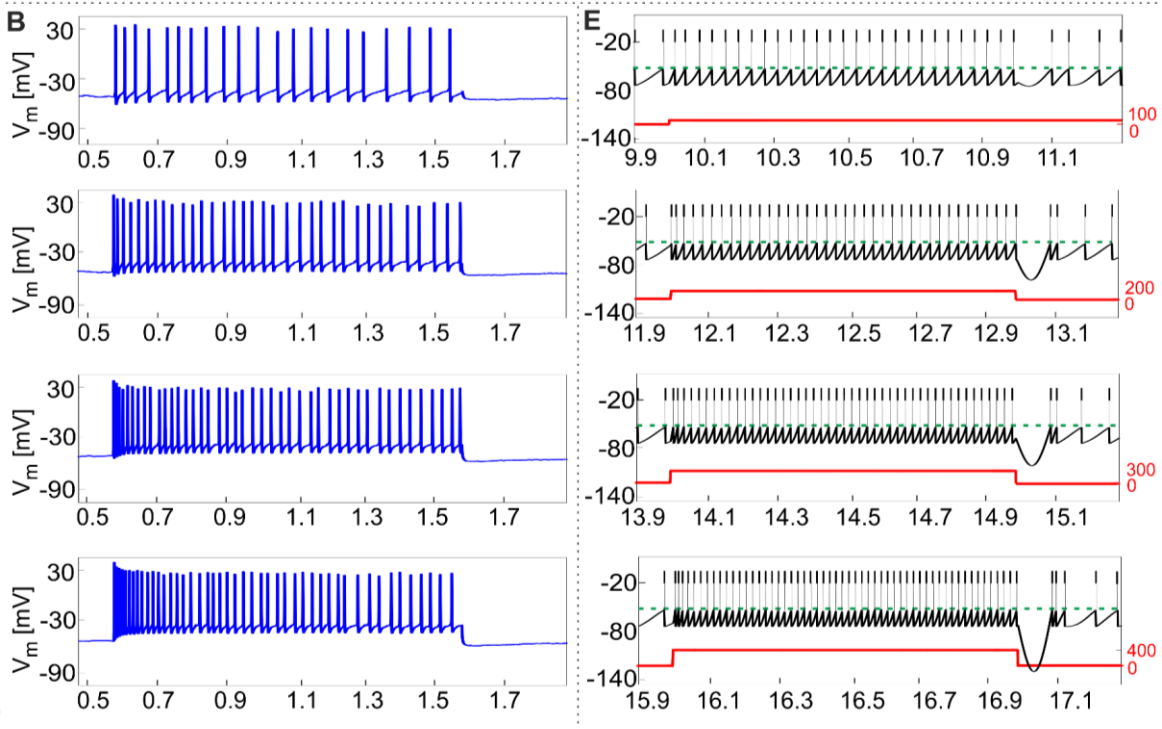
E-GLIF for cerebellar Golgi cells: validation against experimental data

V_m recordings from mice acute cerebellar slices through whole-cell patch-clamp

- Data: 5 Golgi cells from 3 mice
- Stimulation protocol with multiple current steps:

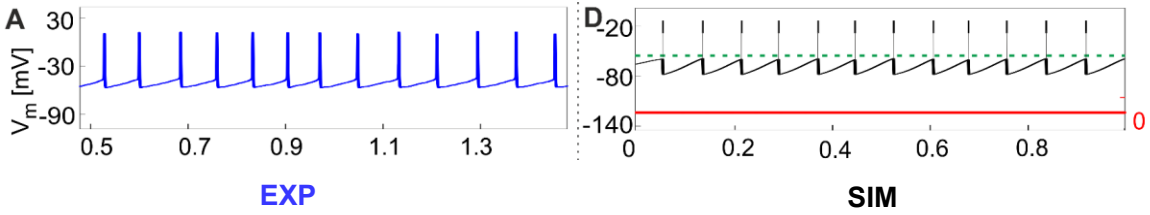


- *<u>f-I_{stim} relationship and adaptation</u> (membrane potential and spikes)*

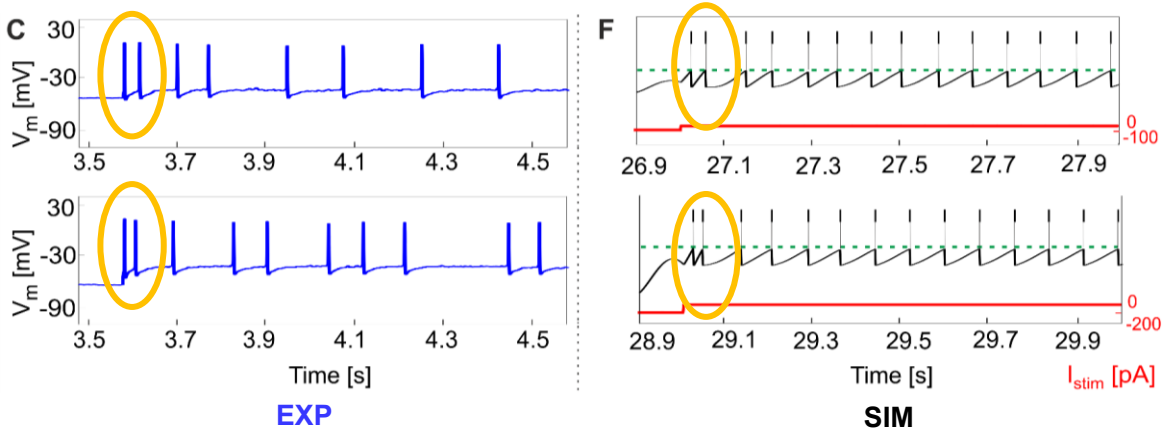


E-GLIF for cerebellar Golgi cells

- *Autorhythm (caused in E-GLIF by model current Ie)*

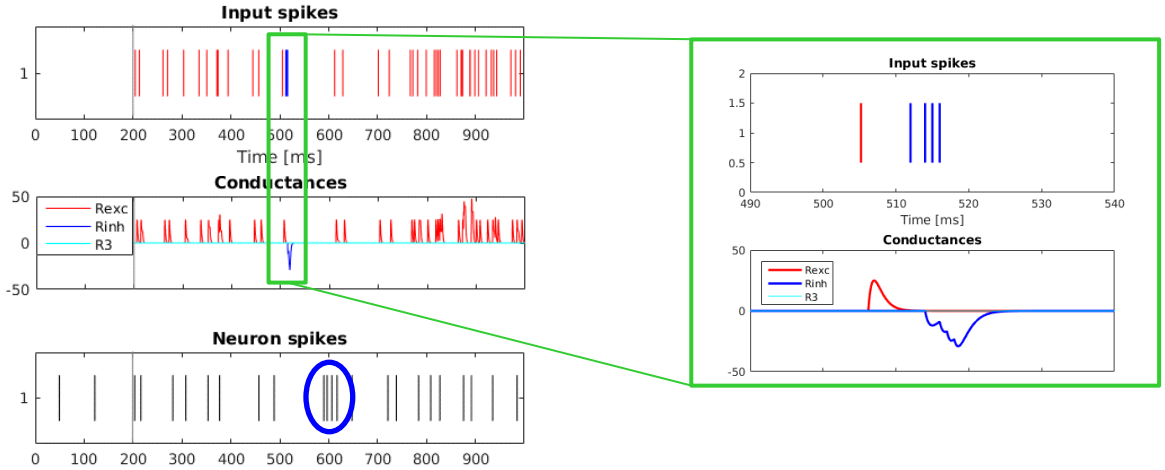


- *Rebound excitation/bursting (caused in E-GLIF by the coupling between Vm and model current Iadap)*



E-GLIF – synaptic inputs

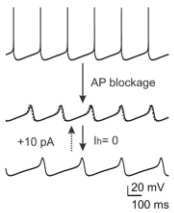
3 receptors with **alpha conductance-based** synapses, for synaptic inputs from different neural populations:

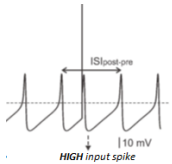


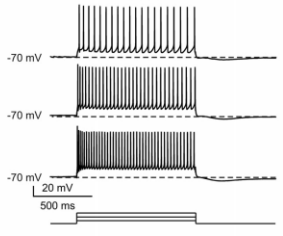
- Increased firing irregularity (Coeff Variation = 39%)
- Rebound burst following inhibitory input burst

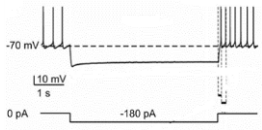
E-GLIF for the other cerebellar neurons

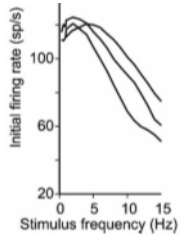
CEREBELLAR CELLS	Auto-rhythm	Sub-Threshold Oscillations	Phase reset	Depolarization-induced burst	Adaptation (SFA)	Post-inhibitory rebound burst	Resonance
Golgi Cell	✓ 12 Hz	✓	✓	✓	✓	✓	✓ (ϑ band)
Granule Cell	-	✓	-	✓	-	-	✓ (ϑ band)
Purkinje Cell	✓ 40-80 Hz	-	-	✓	✓	✓	-
Deep Cerebellar Nuclei	✓ 10-30 Hz	-	-	✓	✓	✓	-
Inferior Olive	-	✓ 1-4 Hz	✓	n.a.		(✓) spike	✓











Adapted from Solinas et al, Front Cell Neurosci, 2007a,b

E-GLIF for the other cerebellar neurons

- *cell-specific input-output relationships to fit*

	Autorhythm	f-I_{stim} relationship		Rebound excitation	
	tonic_freq	$I_{stim} = [exc_1, exc_2, exc_3]$	$f = [freq_1, freq_2, freq_3]$ $(factor_1, factor_2, factor_3)$	*inh*	*lat_rebound; rebound_freq*
GR (D'Angelo et al., 1998)	-	[16, 20, 24] pA	[40±1, 70±1, 120±1] Hz (1, 1, 1)	-10 pA	-
MLI (Galliano et al., 2013)	8.5±2.7 Hz	[12, 24, 36] pA	[30±1, 60±5, 90±10] Hz (1, 1, 1)	-24 pA	-
PC (McKay and Turner, 2005)	65±7 Hz	[500, 1000, 2400] pA	[90±1, 130±1, 242±1] Hz (1.1, 1.1, -)	-2000 pA	≤ 31 ms ≥ 130 Hz
DCNnL (Uusisaari et al., 2007)	30±6 Hz	[142, 248, 426] pA	[50±2, 80±5, 110±15] Hz (1.2, 1.2, 1.2)	-213 pA	≤ 66 ms ≥ 60 Hz
DCNp (Uusisaari et al., 2007)	10±1 Hz	[56, 112, 168] pA	[25±2, 40±2, 45±2]	-84 pA	≤ 200 ms ≥ 20 Hz
IO (De Zeeuw et al., 2003; Mathy et al., 2009)	-	[300, -, -] pA	[273±43, -, -] Hz (5, -, -)	-150 pA	20±2 ms 100±10 Hz

[Geminiani et al, *Front Comput Neurosci*, 2019]

E-GLIF for the other cerebellar neurons

- *cell-specific input-output relationships to fit*

	Model currents constraints	V_{m_inh} range	Solution type	Oscillation limits
GR	$-5 < I_e < 5$ pA $-10 < A_2 < 30$ pA $0.01 < A_1 < 30$ pA	-	oscillatory	$3 < f_{osc} < 8$ Hz $V_{m\_ss\_tonic} < 0.9 \cdot E_L$ $A_{osc\_tonic} < 10$ mV
MLI	$0.01 < I_e, A_2, A_1 < 10$ pA $A_2 < A_1$	$-150 < V_{m\_inh} < -80$ mV	exponential	-
PC	$0.01 < I_e, A_2, A_1 < 1500$ pA $A_2 < A_1$	$-175 < V_{m\_inh} < -45$ mV	oscillatory damped/exponential	-
DCNnL	$0.01 < I_e < 100$ pA $0.01 < A_2, A_1 < 500$ pA $A_2 < A_1$	$-150 < V_{m\_inh} < -40$ mV	oscillatory damped/exponential	-
DCNp	$0.01 < I_e < 100$ pA $0.01 < A_2 < 200$ pA $0.01 < A_1 < 200$ pA	$-155 < V_{m\_inh} < -60$ mV	oscillatory damped/exponential	-
IO	$-30 < I_e < -5$ pA $0.01 < A_2 < 1500$ pA $0.01 < A_1 < 2000$ pA	-	oscillatory	$3 < f_{osc} < 7$ Hz $1.5 \cdot E_L < V_{m\_ss\_tonic} < E_L$ $A_{osc\_tonic} < 10$ mV

[Geminiani et al, *Front Comput Neurosci*, 2019]

E-GLIF for the other cerebellar neurons

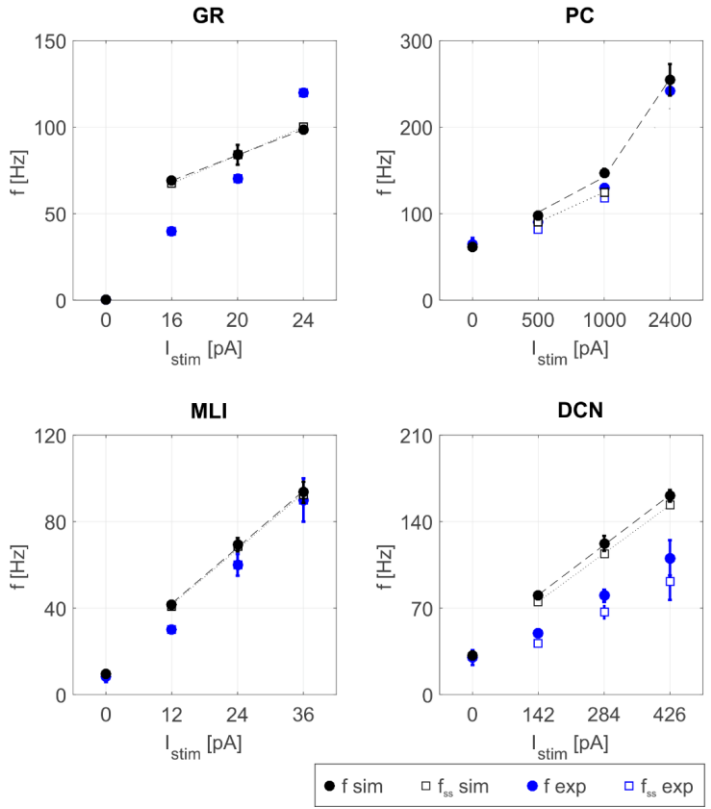
- *ONE parameter SET per neuron type:*

	k_{adap} (MH^{-1})	k_2 (ms^{-1})	A_2 (pA)	k_1 (ms^{-1})	A_1 (pA)	I_e (pA)
GoC (Geminiani et al., 2018)	0.217	0.023	178.01	0.031	259.988	16.214
GR	0.022	0.041	−0.94	0.311	0.01	−0.888
MLI	2.025	1.096	5.863	1.887	5.953	3.711
PC	1.491	0.041	172.622	0.195	157.622	742.534
DCNnL	0.408	0.047	3.477	0.697	13.857	75.385
DCNp	0.079	0.044	176.358	0.041	176.358	2.384
IO	1.928	0.091	1358.197	0.191	1810.923	−18.101

[Geminiani et al, *Front Comput Neurosci*, 2019]

E-GLIF for the other cerebellar neurons

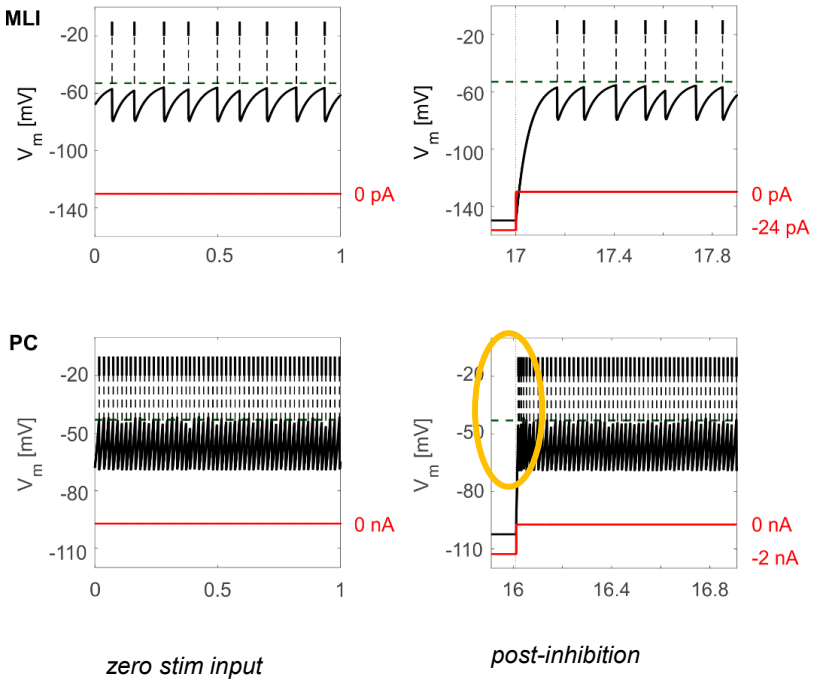
- *f-I_{stim} relationship and adaptation:*
  - Adaptation OK for all neurons
  - F-Istim slope OK for all neurons except GR (but within acceptable experimental ranges)
  - Frequency ranges OK for all neurons; slight increase for DCN, but within physiological ranges



[Geminiani et al, *Front Comput Neurosci*, 2019]

E-GLIF for the other cerebellar neurons

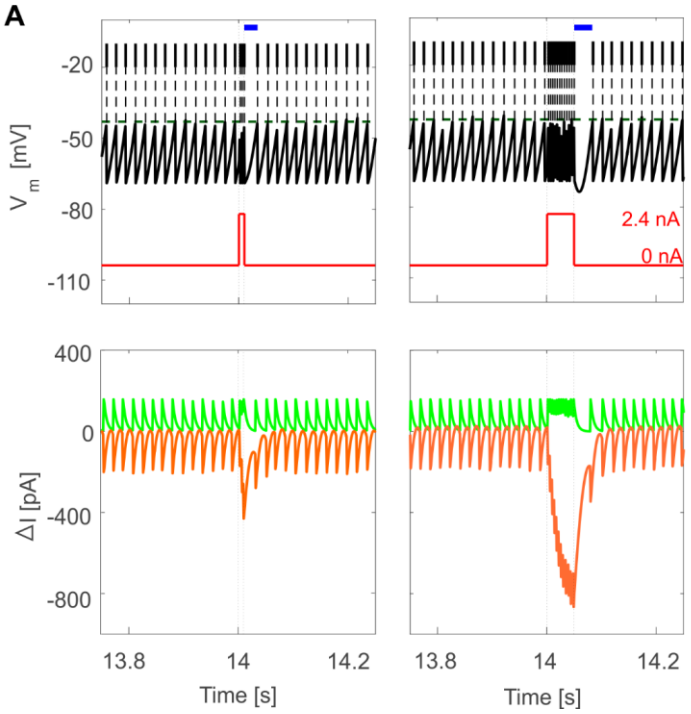
- *Autorhythm/Subthreshold Oscillations and post-inhibitory response*



zero stim input

post-inhibition

- *Neuron-specific properties: bursting*

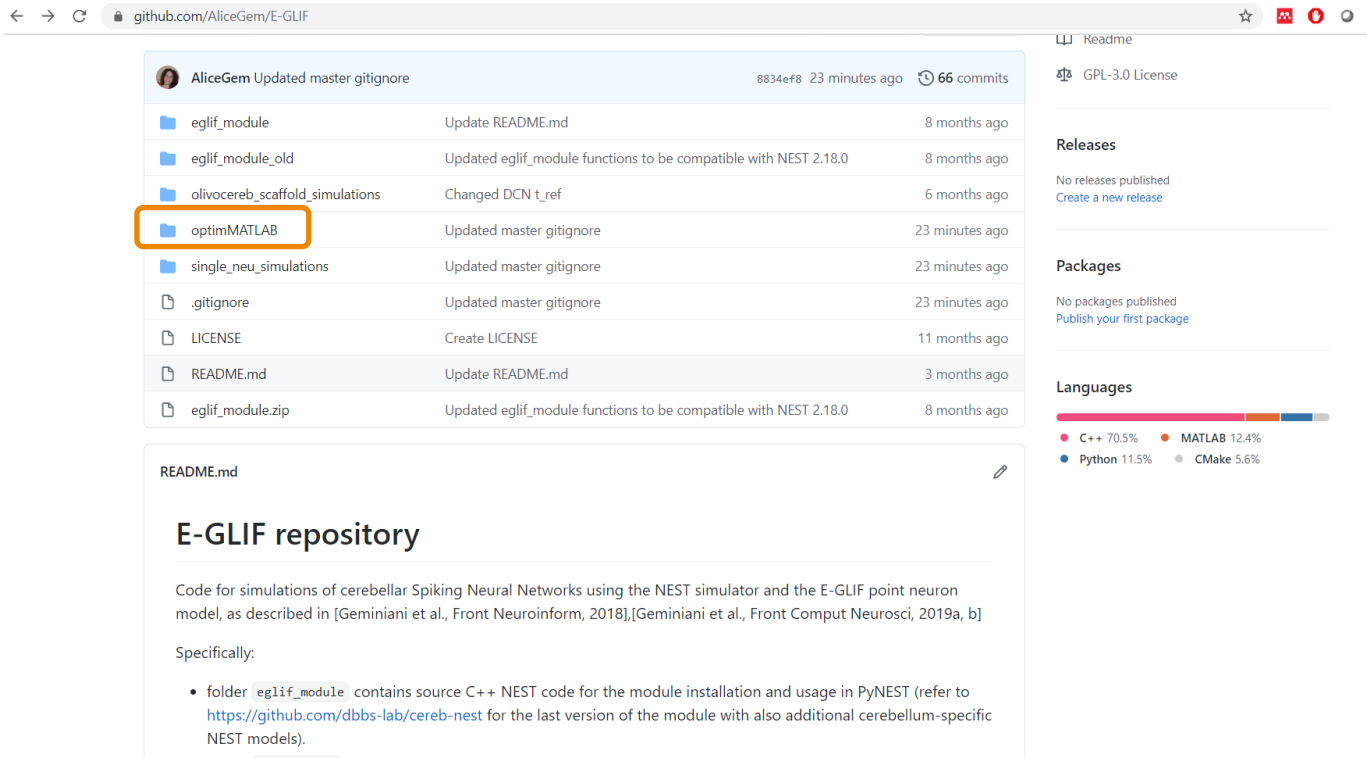


Purkinje cell burst-pause

Break

A deep dive into optimization

https://github.com/AliceGem/E-GLIF



main_optimization.m

Passive membrane properties from literature or experiments:

Editor - /home/nrp/workspace/E-GLIF/optimMATLAB/main_optimization.m

main_optimization.m | objfun_eglif.m | confun_eglif.m

```
16      % Neurons to optimize.
17      % In this example:
18      % MLI = Molecular Layer Interneurons from the cerebellum,
19      % DCN = Deep Cerebellar Nuclei neurons (glutamatergic/GAD-negative large neurons)
20 -    neu_names = { 'MLI', 'DCN'};
21 -    i = 1;      % i-th neuron selected - 1 is MLI, 2 is DCN in this case
22 -    nn = length(neu_names);    % Number of neurons
23
24      % Step 1: set passive membrane parameters (e.g. Cm, tau_m, etc) from neurophysiology experiments
25      % for each of the neurons to be optimized. Reference values should be taken from papers
26      % (for consistency with reference stimulation protocol) or neuroelectro.org
27      % (if not available in papers).
28      % Each parameter is saved in an array of values for each neuron
29 -    Cm = [14.6, 142.0];          % [pF]
30 -    tau_m = [-9.125, -33.0];     % [ms] should be the opposite of the given value
31 -    t_ref = [1.59, 1.5];          % [ms]
32 -    E_L = [-68.0, -45.0];        % [mV]
33 -    Vth = [-53.0, -36.0];        % [mV]
34 -    Vr = [-78.0, -55.0];         % = E_L-10 [mV]
35
```

main_optimization.m

Input-output relationship (autorhythm and depolarizing phases):

Editor - /home/nrp/workspace/E-GLIF/optimMATLAB/main_optimization.m

main_optimization.m

```
34 -    Vr = [-78.0, -55.0];       % = E_L-10 [mV]
35
36 -    m_IF = [8.5 20.0];         % Mean intrinsic frequency - Lachamp, 2009
37 -    sd_IF = [2.7 0 3.0];      % SE o Standard deviation intrinsic frequency - Lachamp, 2009 (reports SE ove
38
39
40      % Step 2: set the target input-output (Istim-firing frequency) relationship from literature stimulatic
41      % For target frequencies, mean and Standard Deviation (SD) values are considered, in order to fit a di
42
43      % Intrinsic firing frequency (/autorhythm/spontaneous firing):
44 -    m_IF = [8.5, 30.0];         % Mean intrinsic frequency
45 -    sd_IF = [2.7, 6.0];      % Standard Deviation of intrinsic frequency (!SE is reported in some studies!)
46
47      % Depolarization phases:
48      % * input current Istim = [3xnn] in [pA], so we use 3 values of input current for 3 depolarazion phase
49      % for the nn neurons considered for optimization
50      % * mean target frequency during depolarization m_Fdep = [3xnn] in [Hz]
51      % * SD of target frequency during depolarization sd_Fdep = [3xnn] in [Hz]
52 -    Istim = [[12.0; 24.0; 36.0], ...      % MLI - Stellate example from [Galliano et al., 2013 - Fig.S3]
53             Cm(i)*[1.0; 2.0; 3.0]];      % DCN - [Uusisaari et al., 2007 - Fig. 7]
54
55 -    m_Fdep = [[30.0; 60.0; 90.0]...
56               [50.0; 80.0; 110.0]];
57
58 -    sd_Fdep = [[1.0; 5.0; 10.0]...
59                [2.0; 5.0; 15.0]];
60
61      % Target frequency at the end of a depolarization step should take into account SFA,
62      % using the parameter SFA_gain = ratio between initial and steady-state firing rate [nnx3]
63      % (it should be set to 1 if no SFA is present)
64 -    SFA_gain = [1.0, 1.0, 1.0; ...
65                 1.2, 1.2, 1.2];
66
```

main_optimization.m

Input-output relationship (hyperpolarizing phase):

```
% Hyperpolarization phase:
% * input current Iinh [pA]
% * minimum Vm value during hyperpolarization Vinh_min [mV]
% * steady-state Vm value during hyperpolarization Vinh_ss [mV]
Iinh = [-24.0, -Cm(i)*1.5];
Vinh_min = [-125, -110];
Vinh_ss = [-115, -95];

% Following hyperpolarization, a rebound burst is present in some neuron types
% Burst frequency is equal to intrinsic frequency if no rebound burst is present (e.g. for MLI)
m_Fburst = [m_IF(1), m_IF(2)*2];          % [Hz]
sd_Fburst = [sd_IF(2), sd_IF(2)];
```

main_optimization.m

Spike times to fit:

```
% Step 3: deriving a spike times distribution of 'ne' samples to fit during optimization

ne = 10;      % 'ne' target values for each stimulation step

% Target spiking times during spontaneous firing (Istim = 0)
T_tonic(i,:) = (1./(sd_IF(i).*randn(1,ne) + m_IF(i)))*1000;

% Avoid negative values in the distribution
while ~isempty(find(T_tonic<0))
    T_tonic(i,find(T_tonic<0))=(1./(sd_IF(i).*randn(1,length(find(T_tonic<0))) + m_IF(i)))*1000;
end

% Target spiking times during depolarization phases (Istim > 0)
T_dep1(i,:) = (1./(sd_Fdep(1,i).*randn(1,ne) + m_Fdep(1,i)))*1000;
T_dep2(i,:) = (1./(sd_Fdep(2,i).*randn(1,ne) + m_Fdep(2,i)))*1000;
T_dep3(i,:) = (1./(sd_Fdep(3,i).*randn(1,ne) + m_Fdep(3,i)))*1000;

Tdep = {T_dep1, T_dep2, T_dep3};

% Target spiking times following hyperpolarization (Istim < 0) - to fit rebound bursting if present
Tburst = (1./(sd_Fburst(i).*randn(1,ne) + m_Fburst(i)))*1000;
Tlb = 5.*randn(1,ne) + 1000*(1/mean(m_IF(i)));

% Target spiking times in the afterhyperpolarization (AHP) when returning to instrinsic firing after
% In the cerebellum, taken into account only for optimization tests on the Golgi cell
Tahp = [5.*randn(ne,1) + 80, 5.*randn(ne,1) + 100, 5.*randn(ne,1) + 120];
```

main_optimization.m

Model solution:

```
%% Model definition - 3D linear ODE system
syms Vm(t) I1(t) I2(t)
syms Ie k_adap k1 k2 A1 A2 ...          % Parameters to be optimized
     Ist


ode1(i) = diff(Vm) == (-1/tau_m(i))*Vm + Ie/Cm(i) + Ist/Cm(i) + I1/Cm(i) - I2/Cm(i) + E_L(i)/tau_m(i);
ode2(i) = diff(I1) == -k1*I1;
ode3(i) = diff(I2) == k_adap*Vm - k2*I2 - k_adap*E_L(i);


I = Ie+Ist;

% Eigenvalues (l1, l2, l3)
l1 = -k1;
D = (1/tau_m(i)+k2)^2-4*(k2/tau_m(i)+k_adap/Cm(i));   % Discriminante
l2 = 0.5*(-(1/tau_m(i)+k2)+sqrt(D));
l3 = 0.5*(-(1/tau_m(i)+k2)-sqrt(D));

% Eigenvectors (x1, x2, x3)
csi = (k2-k1)*tau_m(i)/((1-k1*tau_m(i))*(k2-k1)*Cm(i)+k_adap*tau_m(i));
csi2 = k_adap*tau_m(i)/((1-k1*tau_m(i))*(k2-k1)*Cm(i)+k_adap*tau_m(i));

x1 = [csi; csi2; 1];      % Associated to l1

L2 = Cm(i)*(-1/tau_m(i)-l2);
x2 = [1; L2; 0];            % Associated to l2
```

/home/nrp/workspace/E-GLIF/optimMATLAB/main_optimization.m

main_optimization.m

Optimization function:

```
%% Optimization
% Optimization uses a multi-objective strategy, minimizing an error that takes into account multiple f
% the same time. So the found solution is a compromise between all the features, while also aiming at
delta = (-1/tau_m(i)-k2)^2-4*(-k2/tau_m(i)+k_adap/Cm(i));     % [1/ms^2]
param3_low = 3/((1/(m_IF(i)+3*sd_IF(i)))*1000);
param3_high = 3/t_ref(i);

% Global variables for saving optimization info
global par cf con error_all

low = [Cm(i)/(tau_m(i)^2)+0.000001,-1/tau_m(i)+0.000001,0.0001,3/((1/m_IF(i))*1000),0.0001,0.0001];
up_2 = 10*low(2);
up = [((up_2-1/tau_m(i))^2)*Cm(i)/4-0.000001,up_2,10.0,3/t_ref(i),10.0,10.0];

% Linear inequality constraints: A2<A1; kadap>(Cm/tau_m)*k2 -> in normalized
% ranges!!
% Att: the first constraints should be modified if A1 and A2 are not in the same ranges
A = [0 0 1 0 -1 0;(low(1)-up(1)) (-Cm(i)/tau_m(i))*(up(2)-low(2)) 0 0 0 0];
b = [0;low(1)+(Cm(i)/tau_m(i))*low(2)-0.000001];

% Linear equality constraints
Aeq = [];
beq = [];

% Lower and upper bounds of normalized parameters
lb = zeros(6,1);
ub = ones(6,1);
```

main_optimization.m

Optimization function:

```
for nopt = 1:10
    par = [];
    cf = [];
    con = [];
    error_all = [];

    start_param = rand(6,1)'
    % Check that the constraints are satisfied at initial point
    con1 = (confun_eglif(start_param,low,up,i,Iinh,Cm,tau_m,E_L,Vth, Vinh_ss,t_ref,L2, L3, Sp, T_tonic, Istim,V1,V2,Vss, T_dep3, S
    while con1(1)>0 || con1(2)>0 || con1(3)>0  || con1(4)>0
        start_param = rand(6,1)'
        con1 = (confun_eglif(start_param,low,up,i,Iinh,Cm,tau_m,E_L,Vth, Vinh_ss,t_ref,L2, L3, Sp, T_tonic, Istim,V1,V2,Vss, T_dep
    end

    equs = [equ1;equ2;equ3;equ4;equ5;equ6;equ7;equ8];

% Optimization algorithm options
    options = optimoptions(@fmincon,'Algorithm','sqp','Display','iter-detailed','TolX',1e-3,'TolCon',1e-3,...
        'TolFun',1e-3,'ObjectiveLimit',0.1,'MaxFunEvals',200,'MaxIter',200);         %,'ScaleProblem','obj-and-constr');


% % Separating linear and non linear constraints - error on area Vm
    [param_all,fval_all] = ...
    fmincon(@(param)objfun_eglif(param,low,up,equs,[1.2 1/2 1/4 1/6 1.1 1 1.1 1.1 1 1 1 1 1],Istim,i,T_tonic, Tdep, Tburst, Tlb, t
        A,b,Aeq,beq,lb,ub,@(param)confun_eglif(param,low,up,i,Iinh,Cm,tau_m,E_L,Vth, Vinh_ss,t_ref,L2, L3, Sp, T_tonic, Istim,V1,V
```

objfun_eglif.m

Autorhythm:

```
% Autorhythm step
for ind = 1:length(Tton(i,:))

    % Phase 1
    sol1 = @(t,c1,c2,c3,x1,x2,x3,l1,l2,l3,Sp_1) Vth(i)-vpa(real(((c1*x1*exp(l1*t)+c2*x2*exp(l2*t)+c3*x3*exp(l3*t)+Sp_1))),2);



    area_act_ton(ind,1) = integral(@(t) 0.5*(double(sol1(t,c1,c2,c3,x1,x2,x3,l1,l2,l3,Sp_1))).*(sign(double(sol1(t,c1,c2,c3,x1,x

    error_ton(ind,1) = (abs(area_act_ton(ind,1)-0.5*Tton(i,ind)*(Vth(i)-E_L(i))))/(0.5*Tton(i,ind)*(Vth(i)-E_L(i)));



    for p = 1:3
        error(p) = mean(error_ton(:,p).^2);
    end
```

objfun_eglif.m

Depolarization steps:

```
% Depolarization steps
for dp = 1:3            % Number of depolarizing phases
    clear Sp_1
    Sp_1 = double(subs(Sp(1),{Ist,k_adap,k2,A2,k1,A1,Ie},{Istim(dp,i),norma(param(1),low(1),up(1)),norma(param(2),low(2),up(2)),nor
        norma(param(4),low(4),up(4)),norma(param(5),low(5),up(5)),norma(param(6),low(6),up(6))}));

    for ind = 1:length(T{1}(i,:))

        % Phase 1
        sol1 = @(t,c1,c2,c3,x1,x2,x3,l1,l2,l3,Sp_1) Vth(i)-vpa(real(((c1*x1*exp(l1*t)+c2*x2*exp(l2*t)+c3*x3*exp(l3*t)+Sp_1))),2);


  area_act_dep(ind,1,dp) = integral(@(t) 0.5*(double(sol1(t,c1,c2,c3,x1,x2,x3,l1,l2,l3,Sp_1))).*(sign(double(sol1(t,c1,c2,c

  error_dep(ind,1,dp) = (abs(area_act_dep(ind,1,dp)-0.5*T{dp}(i,ind)*(Vth(i)-E_L(i))))/(0.5*T{dp}(i,ind)*(Vth(i)-E_L(i)));


    for p = 1:3
        error_depol1(p) = mean(error_dep(:,p,1).^2);
    end

    for p = 1:3
        error_depol2(p) = mean(error_dep(:,p,2).^2);
    end

    for p = 1:3
        error_depol3(p) = mean(error_dep(:,p,3).^2);
    end
```

objfun_eglif.m

Post-Hyperpolarization step:

```
error_lat_burst = mean(error_lb.^2);
error_first_burst = mean(error_rb.^2);
```

Overall error and cost function:

```
sq_err = double(sqrt(mean([error error_depol1 error_depol2 error_depol3 error_lat_burst error_first_burst])));

error_all = [error_all; error error_depol1 error_depol2 error_depol3 error_lat_burst error_first_burst];
cf = [cf;sq_err];
```

Can be customized! – e.g.
adding or removing terms,
weighting some terms, etc

confun_eglif.m

```
    % Oscillation parameters
    % Frequency
    delta = (-1/tau_m(neu_ind)-norma(param(2),low(2),up(2)))^2-4*(norma(param(2),low(
    Vinh_act=double(subs(Sp(1),{Ist,k_adap,k2,Ie},{Iin,norma(param(1),low(1),up(1)),n
    Sp_ton=double(subs(Sp(1),{Ist,k_adap,k2,Ie},{0,norma(param(1),low(1),up(1)),norma

   % Nonlinear inequality constraints
  c = [


        % delta positive!
  -delta/abs(delta)+0.000001;


  % Steady state during hyperpolarization
    Vinh_act/(Vinh_ss-35)-1;
    Vinh_act/abs(Vinh_ss+35)+1;


 % SS Vm during tonic above Vth
    Sp_ton/Vth(neu_ind)-1;

];
```

c = []
to have no constraints

c<0

→ To respect all constraints, the *constraints* variable should contain only negative values!

Saved data and results

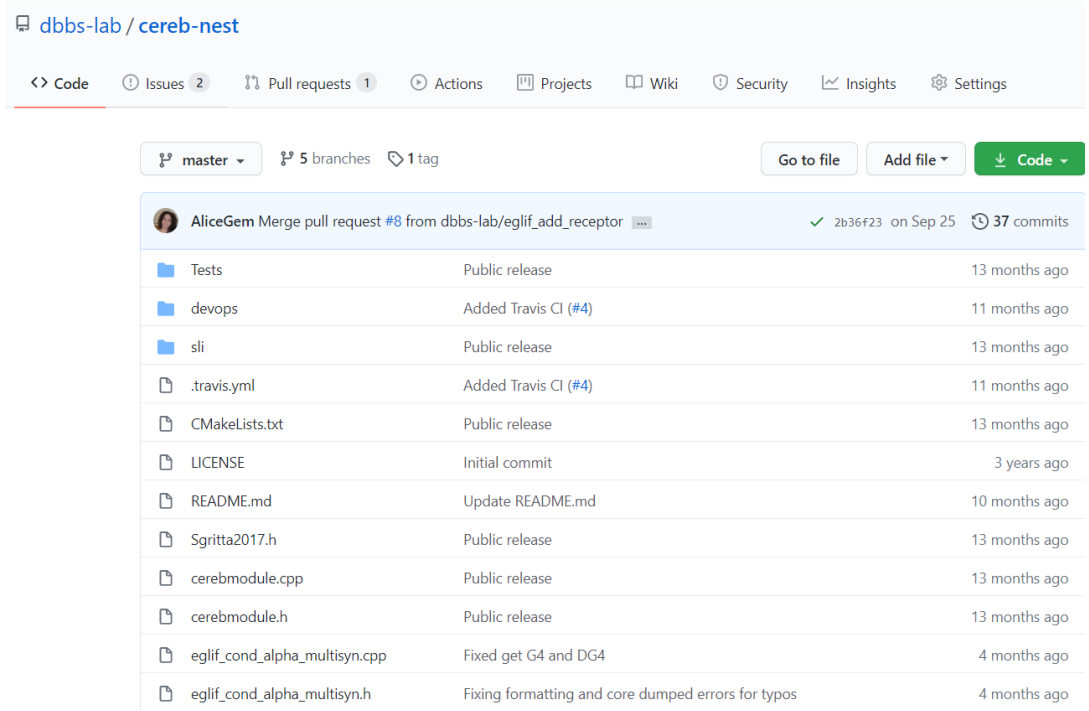
```
    parametri{nopt} = par;
    cost_function{nopt} = cf;
    par_init(nopt,:) = start_param;
    constraints{nopt} = con;
    err_all{nopt} = error_all;
    nopt
end

% Saving optimization data
save param.mat parametri
save cost.mat cost_function
save init_par.mat par_init
save constr.mat constraints
save err_all.mat err_all
```

The algorithm aims at minimizing cost function and constraints at the same time.
Check the values of saved variables throughout optimization to verify properties of the found solution (e.g. converging to minimum value of cost function but not respecting all constraints, etc).

A deep dive into simulation

https://github.com/dbbs-lab/cereb-nest

- ✓ Compatible with NEST 2.18
- ✓ Compatible with NEST3 work in progress



PyNEST simulation

single_neu_simulations/single_neuron_simulation.py

```
#From MATLAB optimization
param_all = [2.0246, 1.0959, 5.8634, 1.8868, 5.9532, 3.7113]

#MLI neuron: instrinsic tonic activity;
nest.SetStatus(single_neuron_MLI, {'t_ref' : 1.59,
                                   'C_m' : Cm_MLI,
                                   'tau_m' : 9.125,
                                   'E_L' : -68.0,
                                   'I_e': param_all[5],
                                   'kadap' : param_all[0],
                                   'Vinit':-68.0,
                                   'V_reset' : -78.0,
                                   'V_th' : -53.0,
                                   'lambda_0' : 1.8,
                                   'tau_V' : 1.1,
                                   'k1' : param_all[3],
                                   'k2': param_all[1],
                                   'A1' : param_all[4],
                                   'A2' : param_all[2],
                   })
```

→ $[k_{adap}, k_2, A_2, k_1, A_1, I_e]$

Simulation analysis

single_neu_simulations/single_neu_analysis.m

```
figure;
plot(mult(:,2),mult(:,3),'k','LineWidth',2)
xlim([0 max(mult(:,2))])
hold on
plot(mult(:,2),mult(:,4),'--','Color',[7 93 13]/255,'LineWidth',2)
hold on
plot(mult(:,2),-85+I/32,'r','LineWidth',2)
hold on
for sp = 1:length(spk)
   line([spk(sp,2) spk(sp,2)],[-20 -10],'Color','k','LineWidth',2)
   line([spk(sp,2) spk(sp,2)],[mult(1,4)-0.5 -20],'LineStyle','--','Color','k','LineWidth',1)
end
xlabel('Time [s]')
ylabel('V_m [mV]')
set(gca,'FontSize',12)
```

Plot state variables

```
%% Plot frequencies, SFA and resonance parameters

% Frequencies

xax = [1:4]

% Plot REFERENCE values from literature, used as targets in optimization
fI_PC = 0.1;        % Masoli
mean_des_all = [60.0 ([800.0 1600.0 2400.0]*fI_PC+80).*[1 1 2.5];  ...   % PC
    8.5 30 60 90;...
    30 50 80 110];
mean_des_all = [60.0 ([400.0 800.0 2400.0]*fI_PC+80).*[1 1 2.5];  ...   % PC
    8.5 30 60 90;...
    30 50 80 110];
sd_des_all = [7 1 1 1;...
    2.7 1 5 10;...
    6 2 5 15];
mean_des_end_all = [mean_des_all(:,2:end)./[1 1.25 10;1 1 1;1.1 1.1 1.4]];              % DCN with linear adaptation
mean_des_end_all = [mean_des_all(:,2:end)./[1 1.25 10;1 1 1;1.2 1.2 1.2]];              % DCN with constant adaptation
sd_des_end_all = sd_des_all(:,2:end);
mean_des = mean_des_all(neu_type,:);
sd_des = sd_des_all(neu_type,:);
mean_des_end = mean_des_end_all(neu_type,:);
sd_des_end = sd_des_end_all(neu_type,:);
figure
plot(xax,mean_des,'go','MarkerFaceColor','g')
hold on
errorbar(xax,mean_des,sd_des,'g','LineStyle','none')
plot(xax(2:end),mean_des_end,'gsquare')
```

Plot frequencies compared to desired values

RisingNet use cases

- Hippocampus:

  Neurons with strong gradual spike-frequency adaptation

→ *ad hoc* modifications:

  - Cost function modified to fit the first 5 spike times
  - Variable A1 and A2 (intrinsic currents updates)

- Basal Ganglia
- Cerebral Cortex

RisingNet use cases - discussion

- NOISE sources

  present:

  - ✓ Stochastic threshold
  - ✓ Random V_{init}
  - ✓ Random synaptic input (spike train Poisson)

  to add:

  - ○ Adding variability to parameters around the optimized value (sensitivity analysis)
  - ○ For each neural population, we could have multiple families of neuron types with different parameter sets.

- CONDUCTANCE:

  Current-based vs conductance-based conditions → record V_m in network simulations with syn inputs to test conductance responses; adaptation conductance instead of adaptation currents (accumulation of Calcium != plasticity mechanisms)

- Systematic Parameter sensitivity analysis needed

- Systematic Validation; currently tested:
  - ✓ Different current inputs wrt ones in optimization
  - ✓ Network simulations

- Depolarization block – saturation and other features we want to fit → let's list and characterize for each neuron type

- Customizing features (e.g. cost function etc) on same optimization algorithm

UNIPV

Egidio D'Angelo (DIRECTOR)

Neurocomputation lab

Claudia Casellato
Robin De Schepper
Alice Geminiani
Stefano Casali
Cristiano Alessandro
Stefano Masoli
Martina Rizza

Neuroimaging lab

(Center for Health and Technology)
Claudia Gandini Wheeler-Kingshott
(UCL)
Fulvia Palesi
Roberta Lorenzi

POLIMI (NEARLab)
Alessandra Pedrocchi
Alessandra Trapani
Francesco Sheiban
Massimo Grillo

Thanks
for your attention!