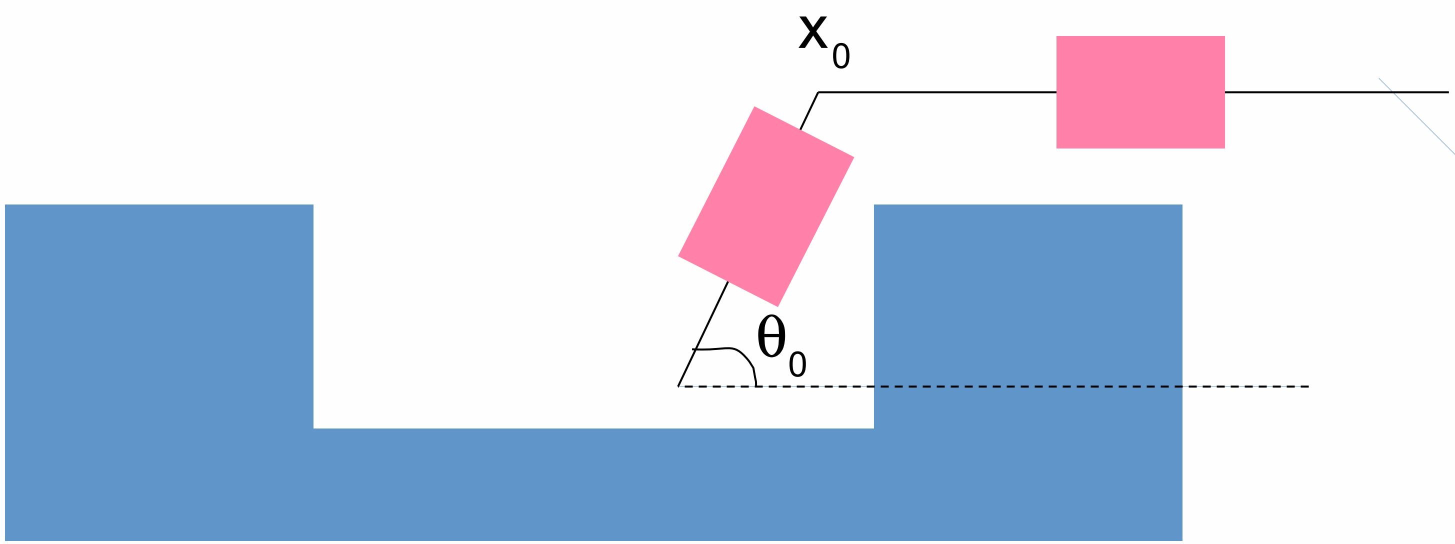
TECHNISCHE UNIVERSITÄT WIEN
Vienna | Austria
Institute of Computer Engineering
Cyber-Physical Systems Group

From Artificial to Biological Neural Networks and Back

Radu Grosu

Parallel Parking: Deterministic Algorithm



while $(x > x_0)$ go back;

while $(\theta > \theta_0)$ turn;

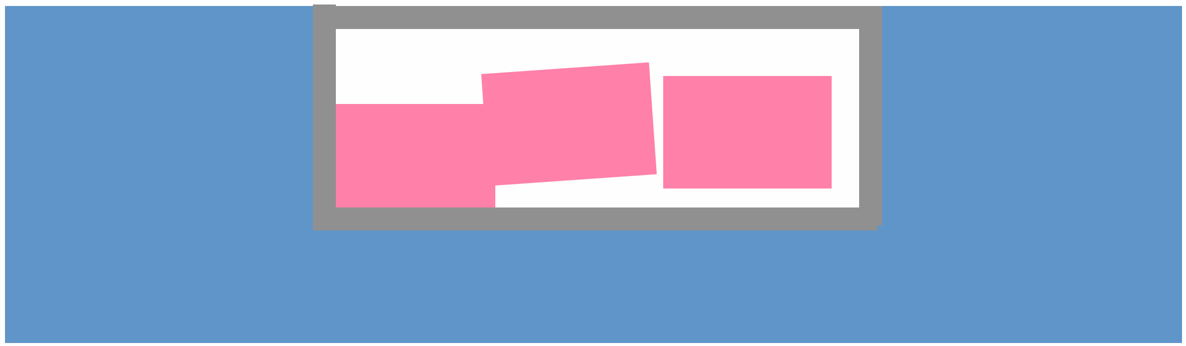
...

Not smart/robust

- Optimization tough
- Sliding problem

Optimization goal: Learn x_0, θ_0, …

Parallel Parking: Deterministic Algorithm



while $(x > x_0)$ go back;

while $(\theta > \theta_0)$ turn;

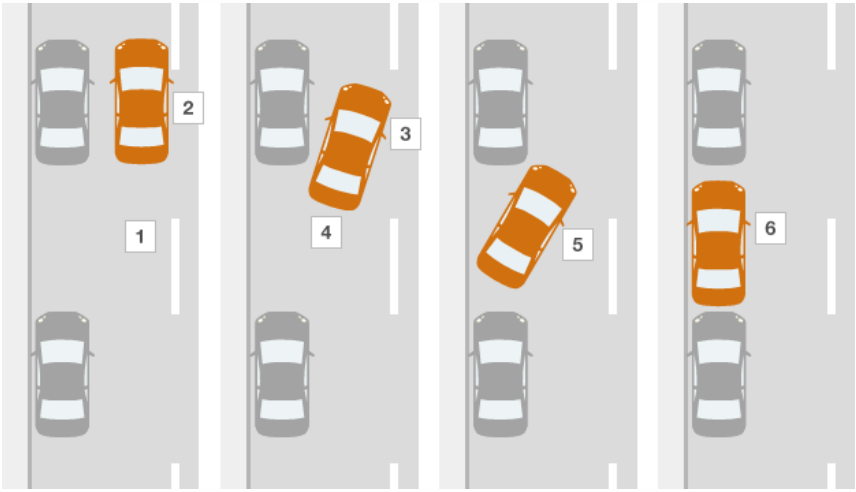
...

Not smart/robust

- Too restrictive
- Many correct solutions

Optimization goal: Learn x_0, θ_0, …

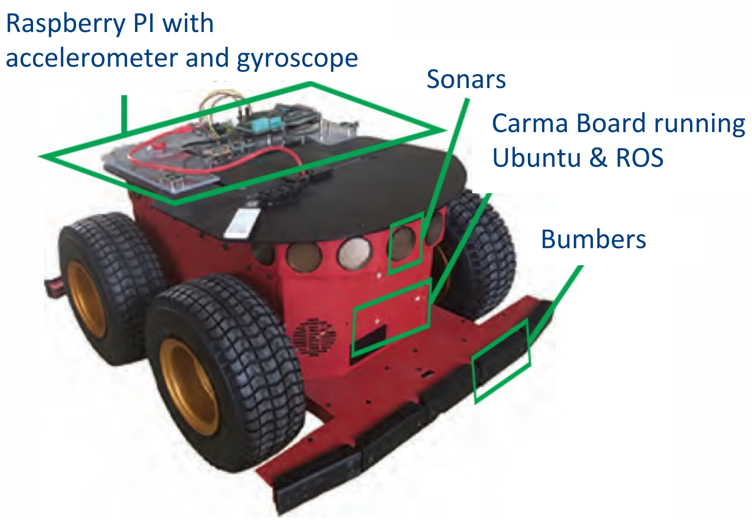
Parking: where our journey starts



(http://www.bbc.com/news/magazine-22350646)

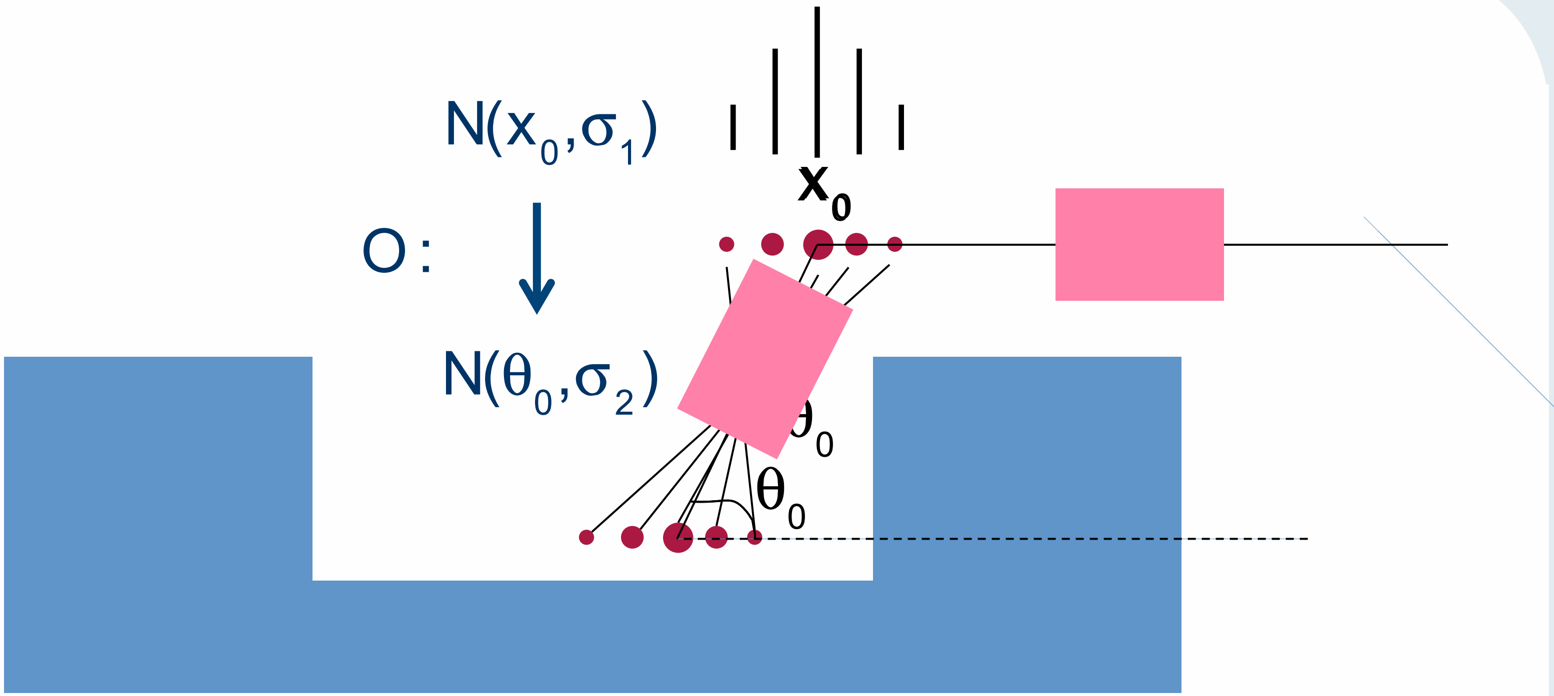
When we park:

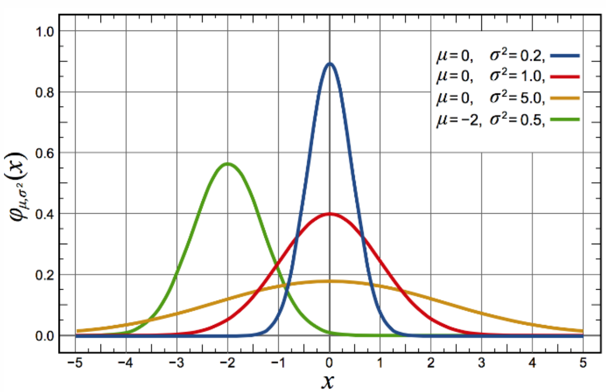
- do it differently
- adapt to environment

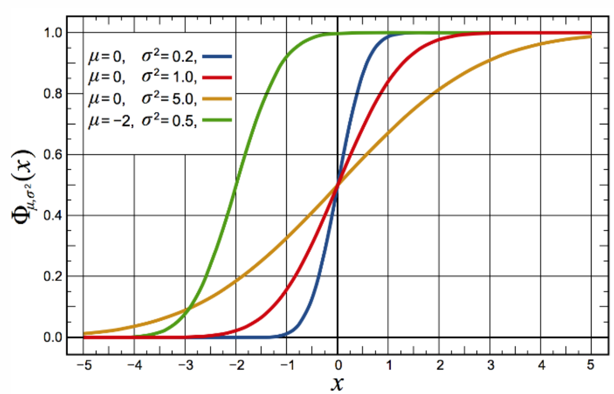


What can we learn from biological systems to do better engineering?

Capturing freedom: Neural Network

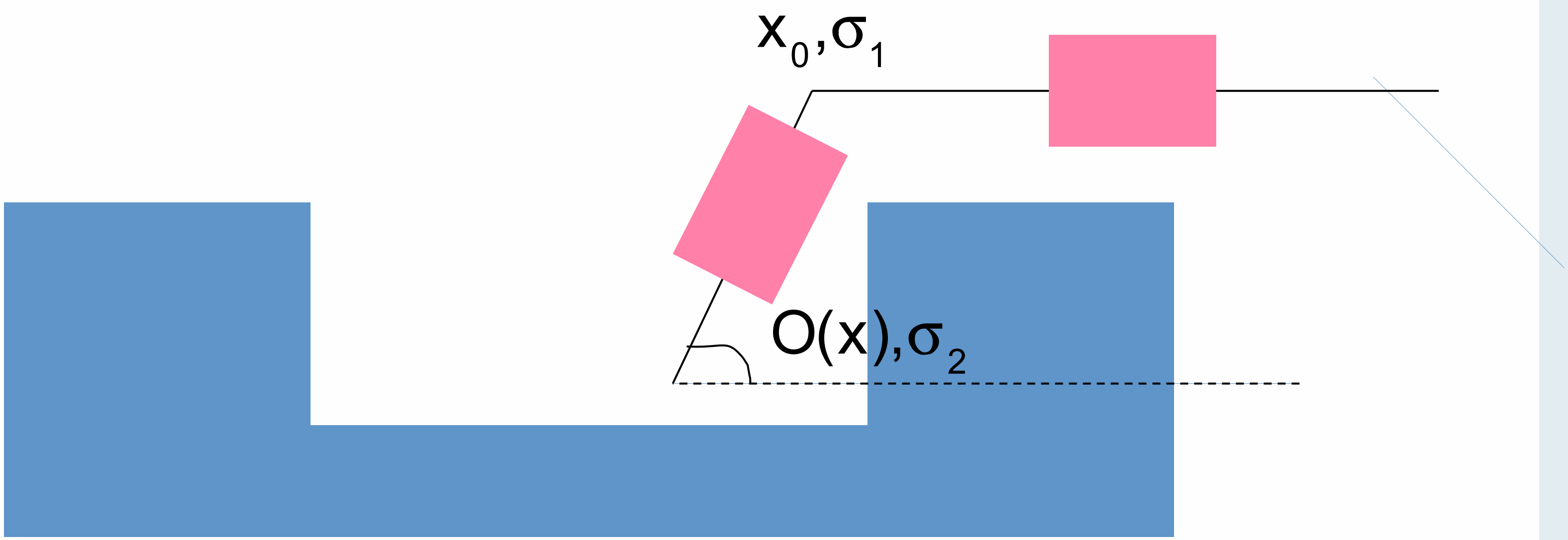






Ontology O: NN of guard dependencies

- nwhile: Between neural-**switch decisions**



nwhile $(x > x_0, \sigma_1)$ go back;

nwhile $(\theta > O_\theta(x), O_\sigma(x))$ turn;

...

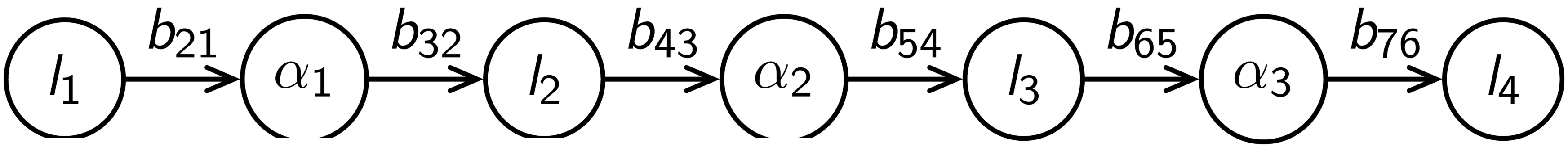
Gaussian-Bayesian Network to learn the parameters

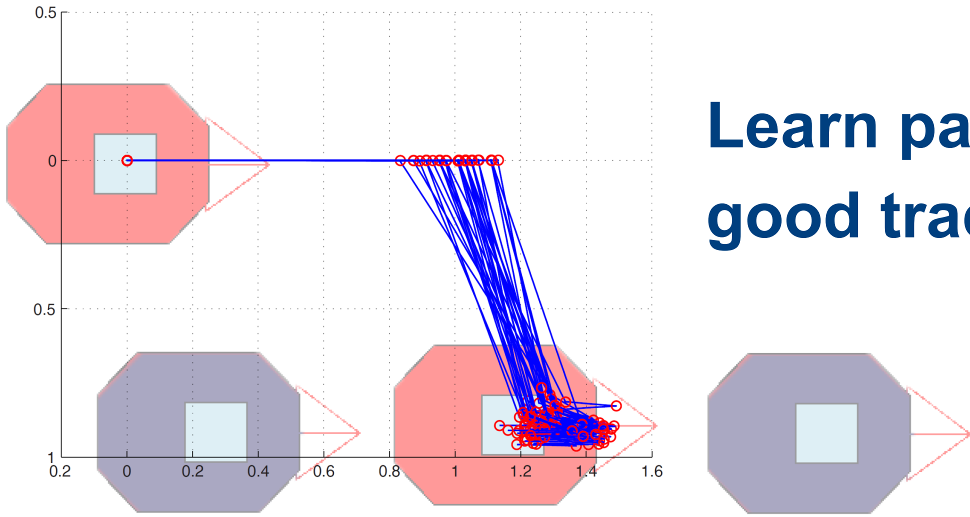
Parallel Parking: Neural Program Sketch

```
nwhile(currentDistance < targetLocation1, sigma1){
  moving();
  currentDistance = getPose();
  }

updateTargetLocations();
nwhile(currentAngle < targetLocation2,    sigma2){
  turning();
  currentAngle = getAngle();
  }

updateTargetLocations();
nwhile(currentDistance < targetLocation3, sigma3){
  moving();
  currentDistance = getPose();
  }
...
```

Neural-Program: Learning

Gaussian-Bayesian Network (GBN):





Learn parameters of GBN from good traces:

1. Convert the GBN to a MGD (multivariate Gaussian distr.)
2. Update the covariance matrix Σ^{-1} of the MGD
3. Extract sigmas and b_{ij}s from precision matrix $T=\Sigma^{-1}$

Neural-Program: Learning

Iterative learning procedure:

Incrementally update mean and covariance matrix of the prior

Mean update:

$$\bar{\mathbf{x}} = \frac{\sum_{h=1}^{m} \mathbf{x}^{(h)}}{m}$$

$$\bar{\mathbf{x}}_{m+1} = \frac{\mathbf{x}^{(m+1)} + m\bar{\mathbf{x}}_m}{m+1} = \bar{\mathbf{x}}_m + \frac{1}{m+1}(\mathbf{x}^{(m+1)} - \bar{\mathbf{x}}_m)$$

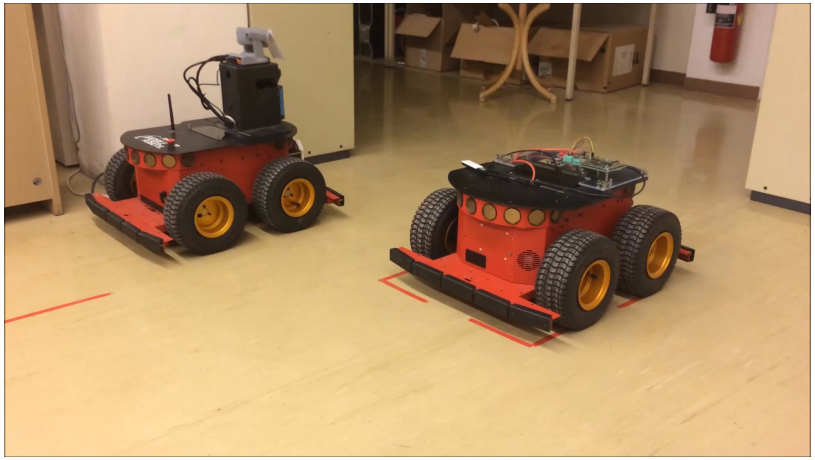
Covariance update:

$$\mathbf{s} = \sum_{h=1}^{m} (\mathbf{x}^{(h)} - \bar{\mathbf{x}})(\mathbf{x}^{(h)} - \bar{\mathbf{x}})^T$$

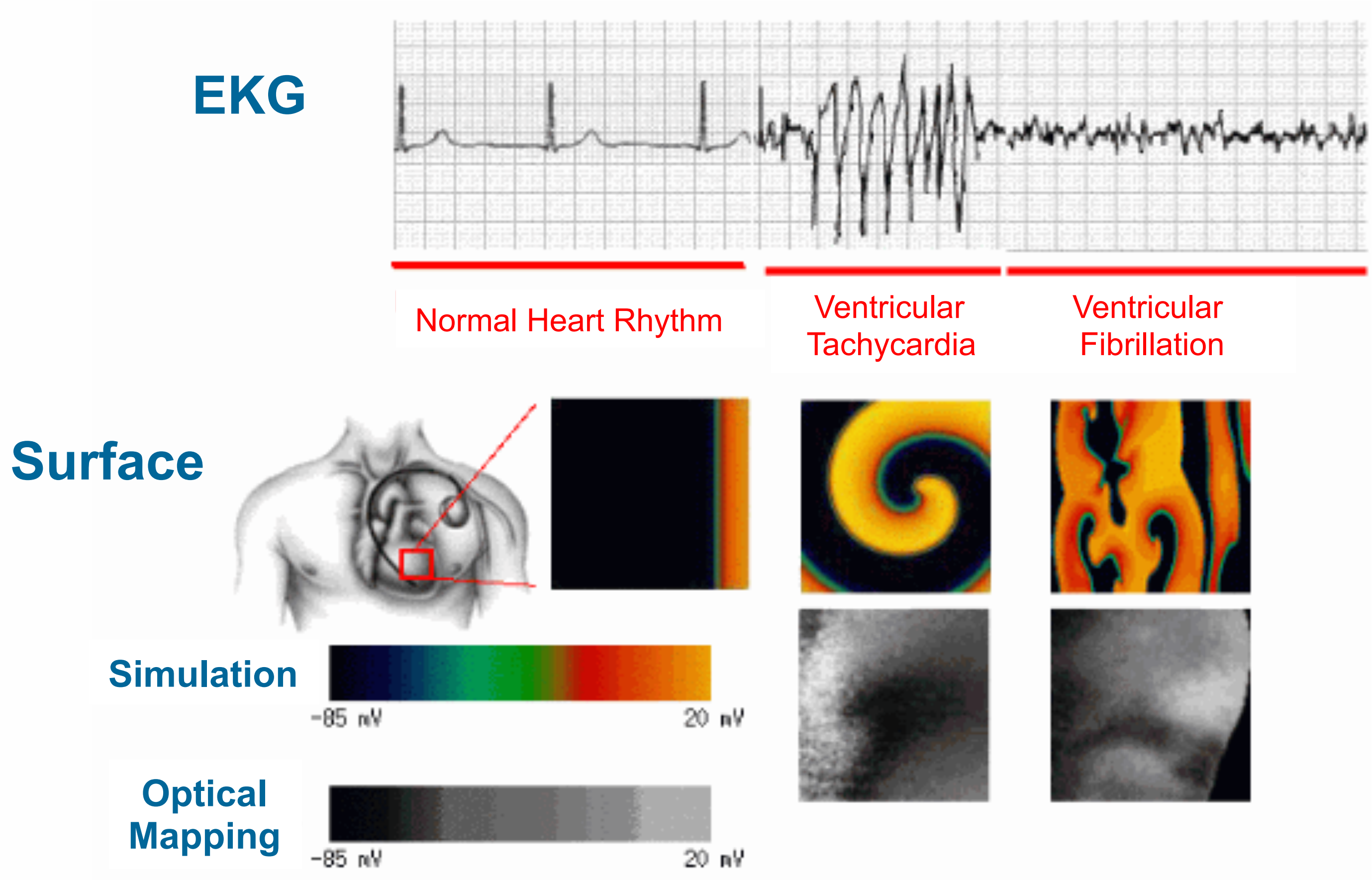
$$\mathbf{s}_{m+1} = \mathbf{s}_m + (\mathbf{x}^{(m+1)} - \bar{\mathbf{x}}_m)(\mathbf{x}^{(m+1)} - \bar{\mathbf{x}}_{m+1})^T$$

$$\mathbf{T}^{-1} = \mathbf{s}$$

Pioneer Rovers: Normal2, Water, Paper



Emergent Behavior in Cardiac Cells



Arrhythmia afflicts more than 3 million Americans alone

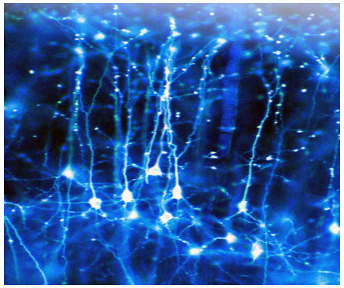
Excitable Cells

Generate action potentials (elec. pulses) in response to **electrical stimulation**

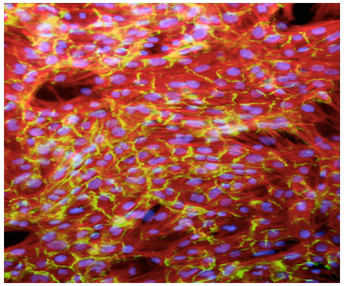
- **Examples:** neurons, cardiac cells, etc.

Local regeneration allows electric signal propagation **without damping**

Building block for electrical signaling in **brain, heart, and muscles**



Neurons of a squirrel
University College London



Artificial cardiac tissue
University of Washington

Single Cell Reaction: Action Potential

Membrane's AP depends on:

Stimulus (voltage or current):

- **External / Neighboring cells**

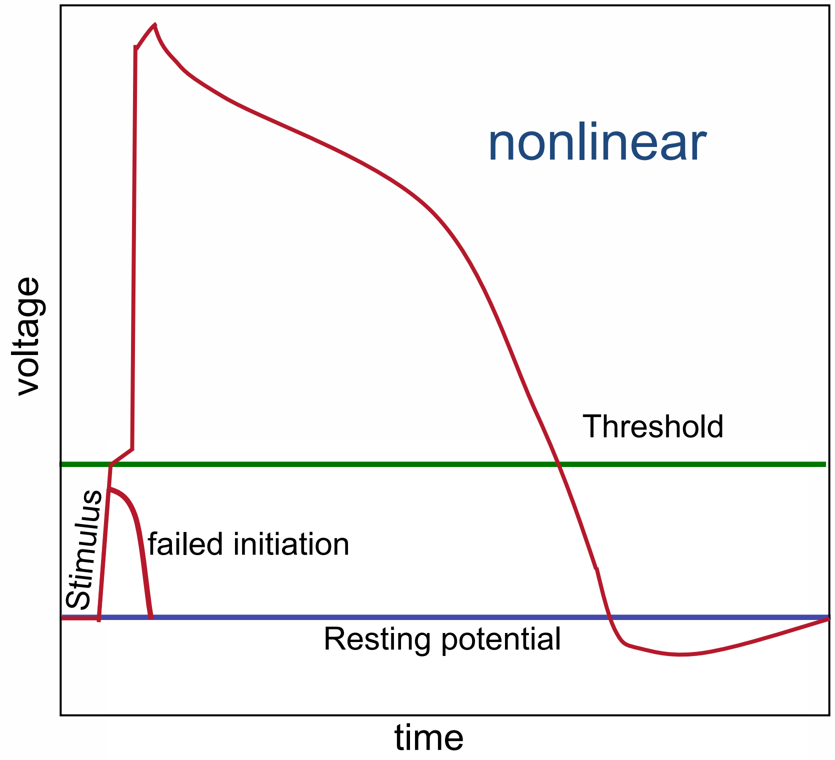
Cell itself (excitable or not):

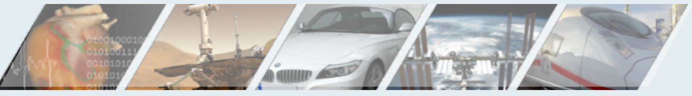
- **State / Parameters value**

Tissue: Reaction / diffusion

$$\frac{\partial \mathbf{u}}{\partial t} = R(\mathbf{u}) + \nabla(D\nabla \mathbf{u})$$

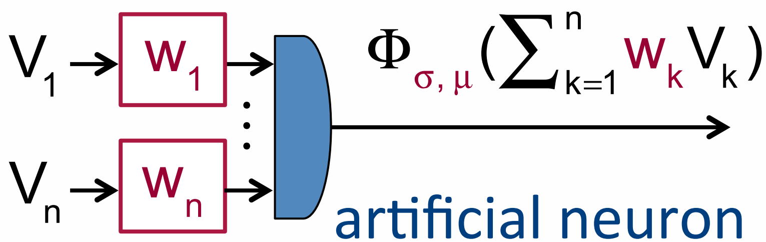
Schematic Action Potential





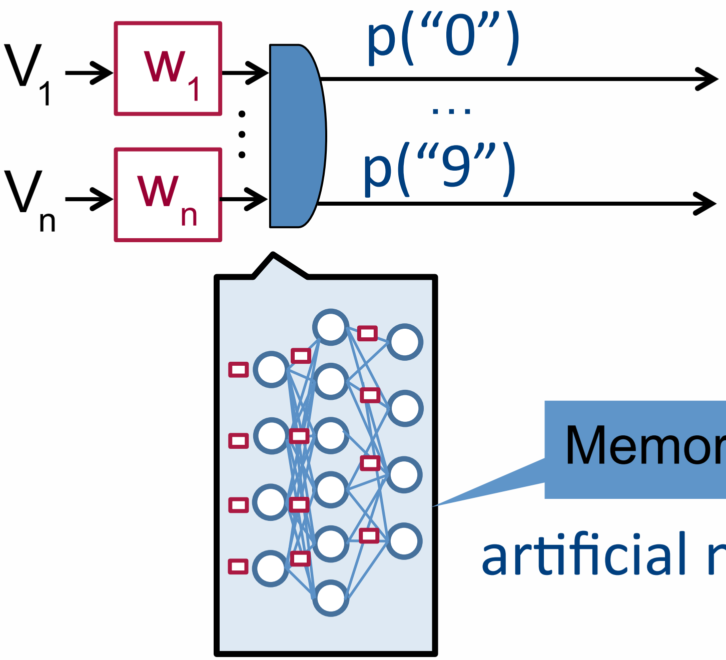
Good Old Artificial Neural Networks (2^{nd} generation)

Combinational Circuit



artificial neuron

$\Phi_{\sigma,\mu}$ nonlinear activation function

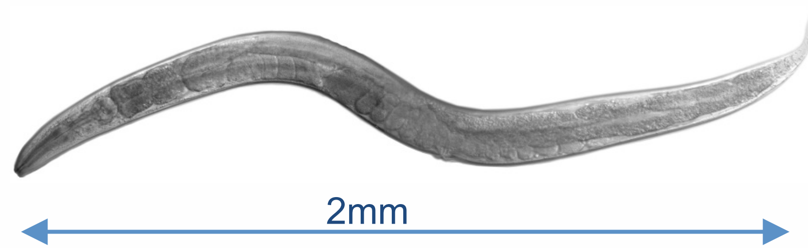


Memoryless Circuit

artificial neural networks

MNIST dataset

C.Elegans as a Model Organism



Human Brain

- 86 billion neurons
- 10 trillion synapses
- 25000 genes

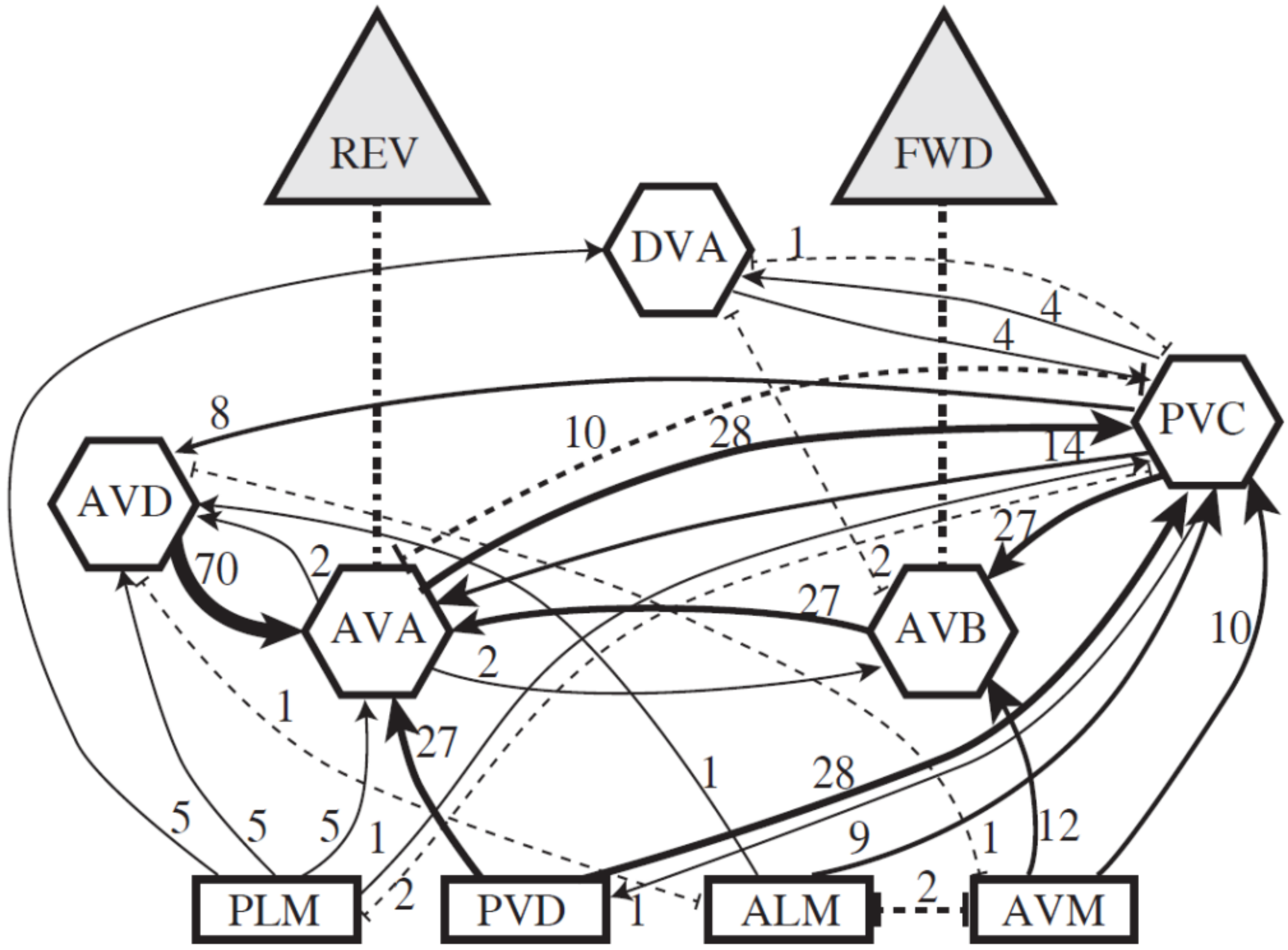
C Elegans Nervous System

- 302 Neurons
- ~7000 synapses
- 20000 genes
- Known connectivity

Striking Similarity

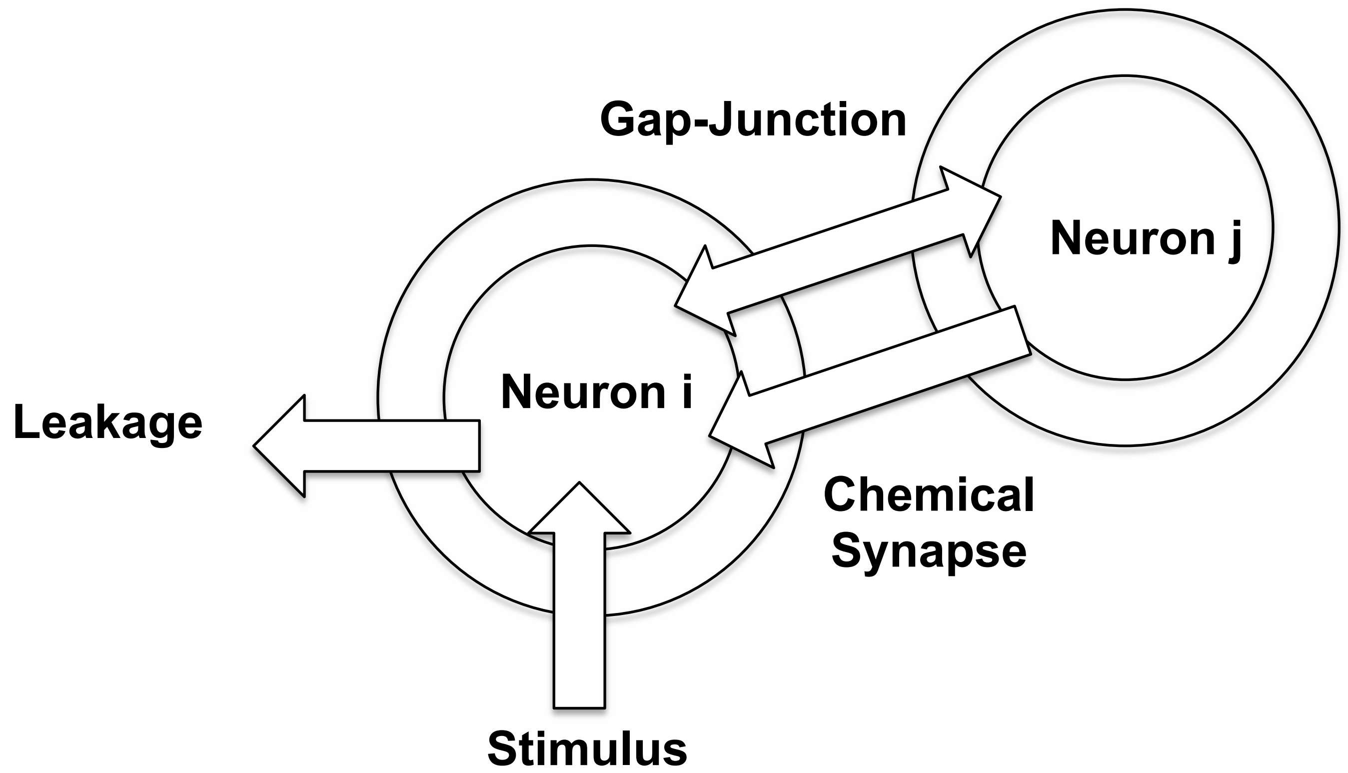
- Neuro-transmitter
- Ionic Channels
- Developmental genes

C.Elegans: Tap Withdrawal Response



Wicks et al., 1996

C.Elegans: Neuronal Dynamics



Dynamics of i^{th} neuron's membrane potential

=

leakage current + gap-junction current
+ synaptic current + stimulus current

Modeling Neuron

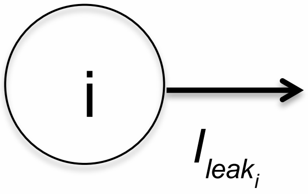
- ## Leakage current

Current flowing out of neuron *i* given by:

$$I_{leak_i} = g_l(V_{leak_i} - V_i)$$

g_l : leakage conductance of neuron *i*

V_{leak_i} : leakage voltage of neuron *i*



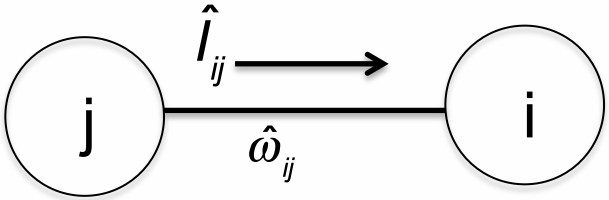
- ## Gap-Junction

current flowing from neuron *j* to neuron *i* is given by:

$$\hat{I}_{ij} = \hat{\omega}_{ij}\hat{g}(V_j - V_i)$$

$\hat{g}$: maximum gap junction conductance

$\hat{\omega}_{ij}$: number of gap-junction connections



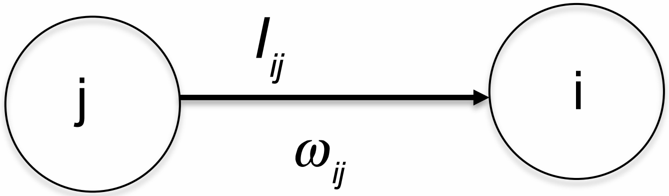
Modeling Neuron

- **Chemical Synapse**

Synaptic current flowing from pre-synaptic neuron *j* to post-synaptic neuron *i* is given as:

$$I_{ij} = \omega_{ij} g_{ij}(V_j)(E_j - V_i)$$

$$g_{ij}(V_j) = \frac{\bar{g}}{1 + e^{-4.39\left(\frac{V_j - V_{EQ_j}}{V_{RANGE}}\right)}}$$



E_j : Reversal potential for synaptic conductance of neuron *j*

$\bar{g}$: maximum synaptic conductance

V_{EQ_j} : Equlibrium potential of V_j

V_{RANGE} : Voltage range over which synapse is activated

ω_{ij} : number of synaptic connections

Model for TW Circuit

The dynamic of i-th neuron of TW circuit:

$$\boxed{C_{m_i}\frac{dV_i}{dt} = I_{leak_i} + \sum_{j=1}^{N}\hat{I}_{ij} + \sum_{j=1}^{N} I_{ij} + I_{stim}}$$

$$I_{leak_i} = g_l(V_{leak_i} - V_i) \text{ (Leakage Current)}$$

$$\hat{I}_{ij} = \hat{\omega}_{ij}\hat{g}(V_j - V_i) \text{ (Gap-junction current)}$$

$$I_{ij} = \omega_{ij} g_{ij}(V_j)(E_j - V_i) \text{ (Synaptic current)}$$

$$g_{ij}(V_j) = \frac{\bar{g}}{1 + \exp\left(-4.39\left(\frac{V_j - V_{EQ_j}}{V_{RANGE}}\right)\right)}$$

C_{m_i} : Capacitance of neuron i

I_{stim} : Stimulus current applied only to sensory neurons

Circuit Output

$$Y = \int_{T_i}^{T_f} (V_{AVB} - V_{AVA})\,dt$$

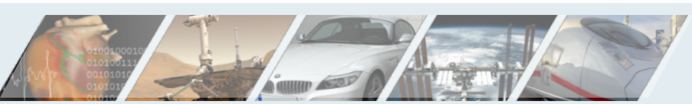
T_i : start time of stimulation

T_f: end time of stimulation

$Y >> 0$: Reversal

$Y << 0$: Acceleration

$Y \sim 0$: No Response



Model Checking for C.Elegans

Given

- *M(x,p)* – mathematical model of TW circuit
- x – state vector
- *p* – parameter vector

Find

- range of *p s.t.*

$$M(x,p) \vDash \phi$$

Behaviors in Temporal Logic

- Reversal:

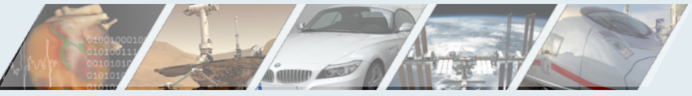
$$\phi :: \forall t \in [T_i, T_f], V_{AVB}(t) > V_{AVA}(t)$$

- Acceleration:

$$\phi :: \forall t \in [T_i, T_f], V_{AVB}(t) < V_{AVA}(t)$$

- No Response:

$$\phi :: \forall t \in [T_i, T_f], \| V_{AVB}(t) - V_{AVB}(0) \| < \delta \wedge \| (V_{AVA}(t) - V_{AVA}(0) \| < \delta$$

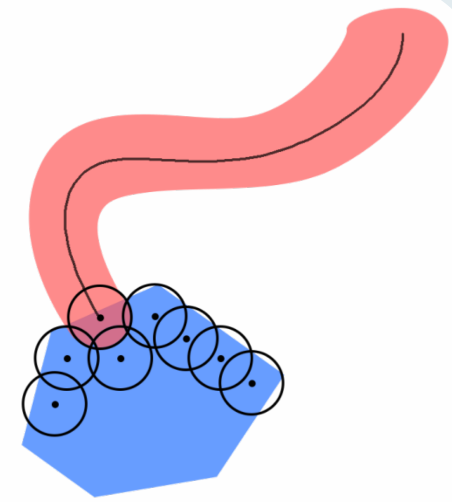


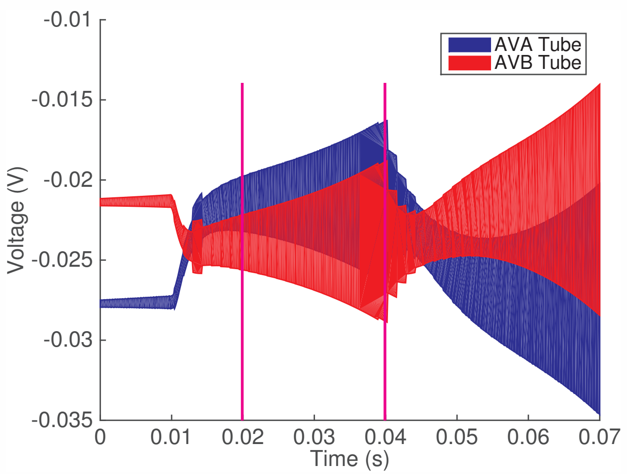
Reach Tube Computation

Finite covers of initial set

Simulate from the center of each cover

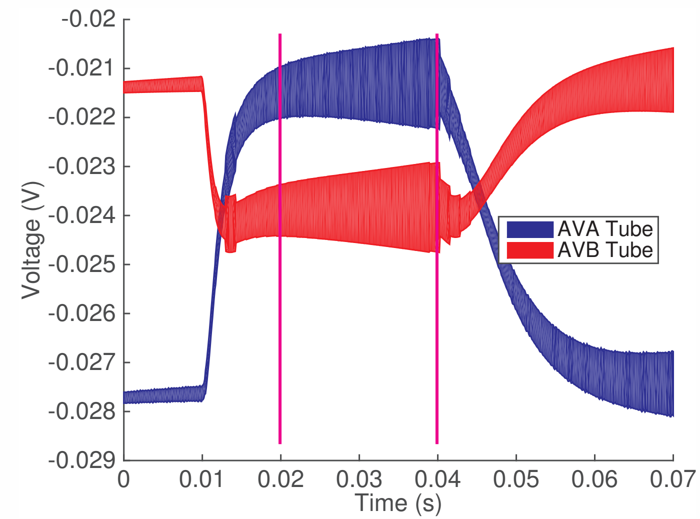
Union of all such tubes gives an over-approximation of the reach set





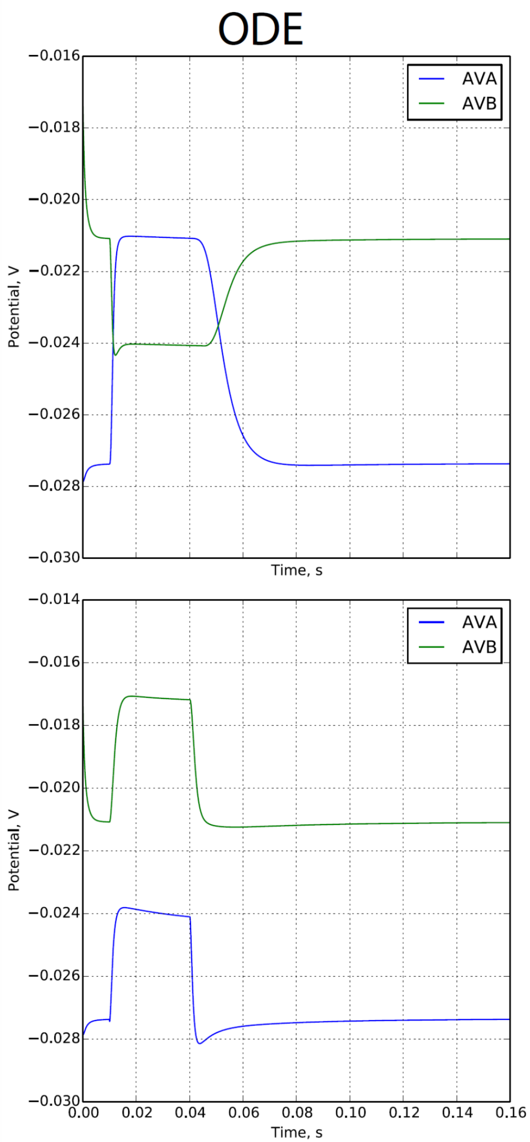
Rev. property not satisfied with $\delta = 1e-4$

$\delta -$refinement



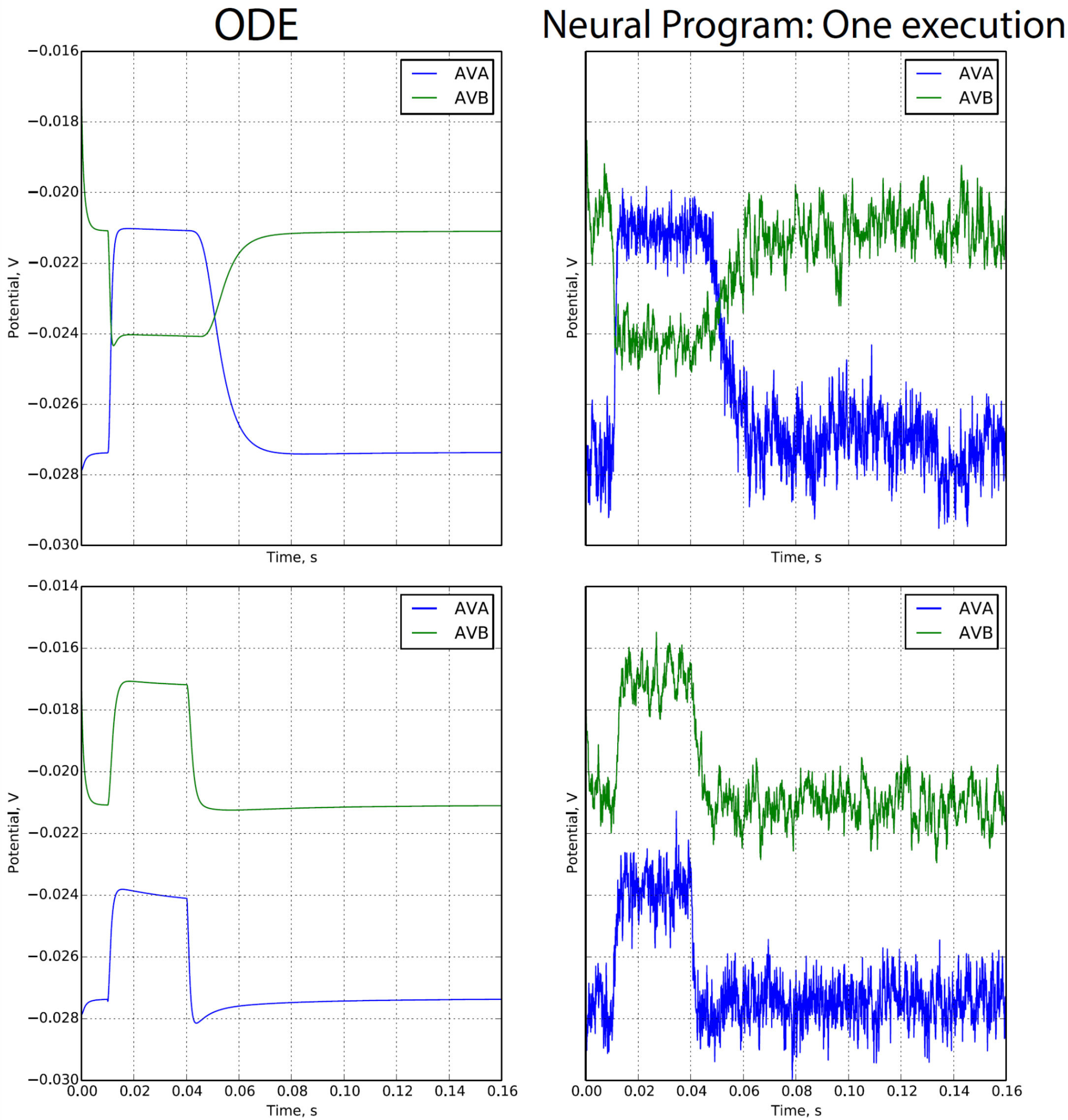
Rev. property satisfied with $\delta = 5e-5$

Tap Withdrawal: ODE Simulations



Is nature like this?

Tap Withdrawal: Simulations with nwhile



Is nature like this?

How to account parameter variance in simulation?

Tap Withdrawal: Simulations with nwhile

Biological model:

$$\frac{dV^{(i)}}{dt} = \frac{V_{Leak} - V^{(i)}}{R_m^{(i)} C_m^{(i)}} + \frac{\sum_{j=1}^{N} (I_{syn}^{(ij)} + I_{gap}^{(ij)}) + I_{stim}^{(i)}}{C_m^{(i)}} \tag{1}$$

$$I_{gap}^{(ij)} = w_{gap}^{(ij)} \, g_{gap}^{(ij)} \, (V_j - V_i) \tag{2}$$

$$I_{syn}^{(ij)} = w_{syn}^{(ij)} \, g_{syn}^{(ij)} \, (E^{(ij)} - V^{(j)}) \tag{3}$$

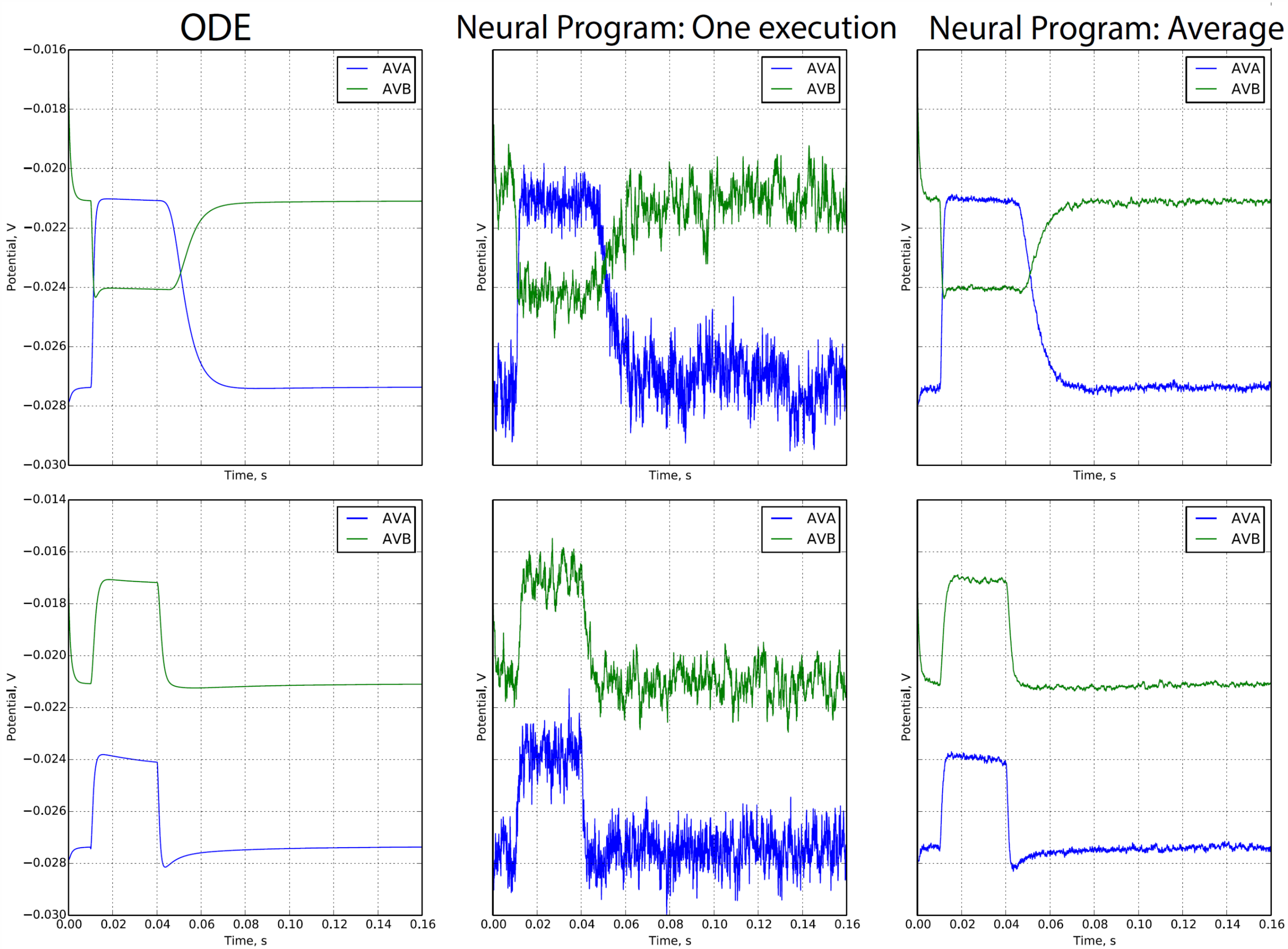
$$g_{syn}^{(ij)}(V^{(j)}) = \frac{\bar{g}_{syn}}{1 + e^{K\left(\frac{V^{(j)} - V_{eq_j}}{V_{range}}\right)}} \tag{4}$$

Neural Program:

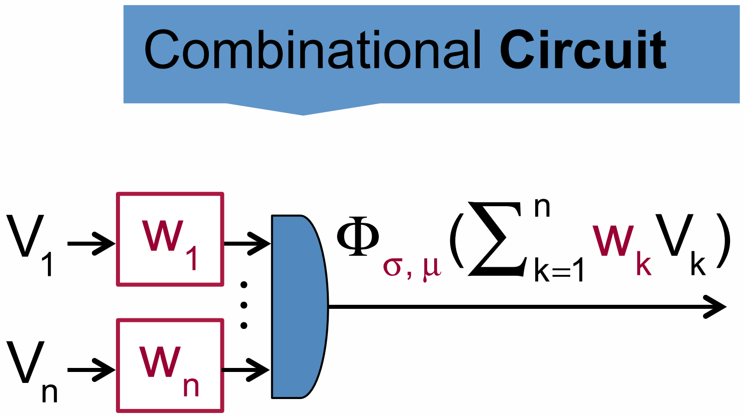
```
nwhile ( t ≤ t_dur , 0)
    compute I_gap^(ij) using equation 2
    nwhile ( k ≤ w_syn^(ij) , 0)
        nif (V^(j) ≤ V_eq , K/V_range)
            g_syn^(ij) ← g_syn^(ij) + g_syn
    compute I_syn^(ij) using equation 3
    compute dV^(i) using equation 1
    V^(i) ← V^(i) + dV^(i)
    t ← t + dt
```

We explicitly allow controllable variance in the program

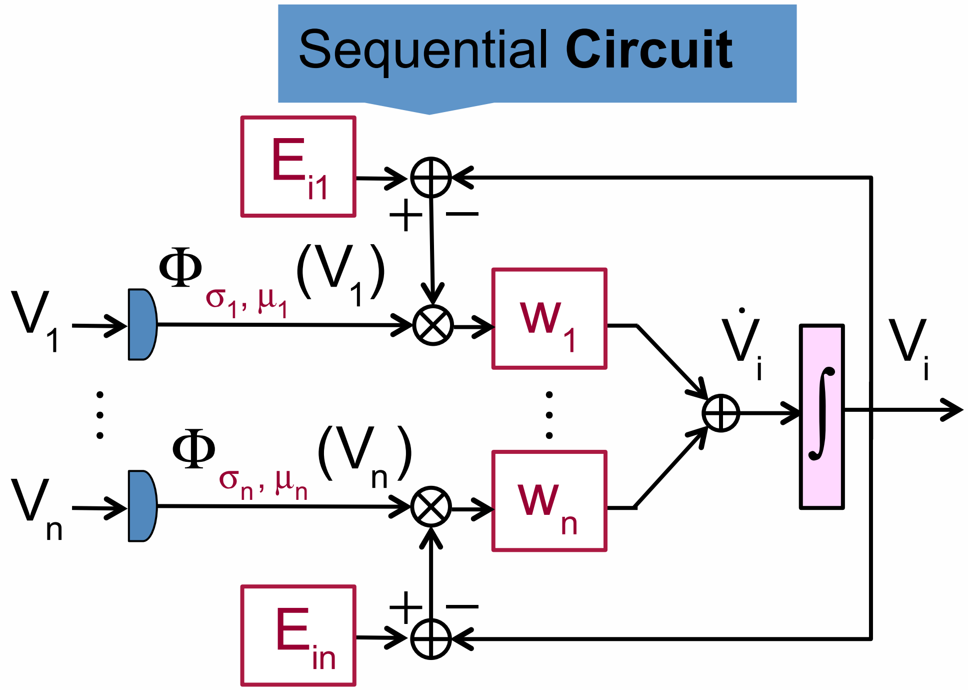
Tap Withdrawal: Simulations with nwhile



Artificial versus Real Neurons



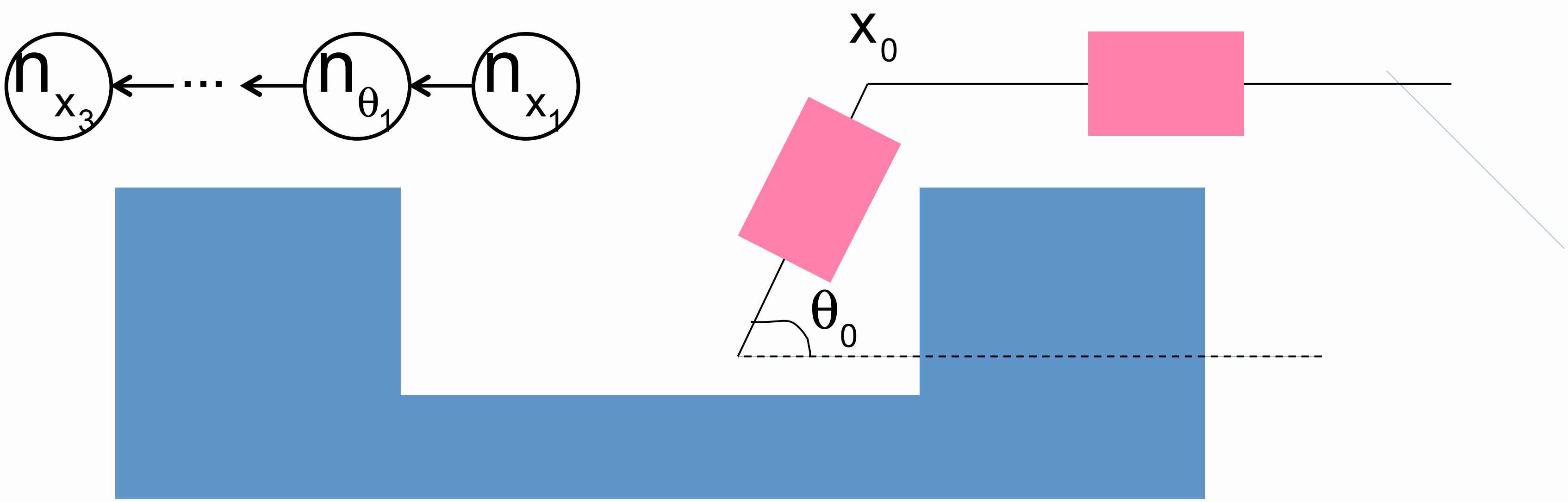
artificial neuron



real neuron

$$\dot{V}_i = \sum_{k=1,\, k\neq i}^{n} \Phi_{\sigma_k,\mu_k}(V_k)\, w_k (E_{ik} - V_i)$$

Neural-Circuit Controller



while (true) { **Proportional Ctrl**

$$x_1 = x_1 + w_{x_1}(x_0 - x_1)\,dt$$

$$\theta_1 = \theta_1 + \Phi_{\sigma, x_0}(x_1)\,w_{\theta_1}(\theta_0 - \theta_1)\,dt$$

... }

Flow Ctrl

for $(i = 1 : w_{\theta_1})$ {

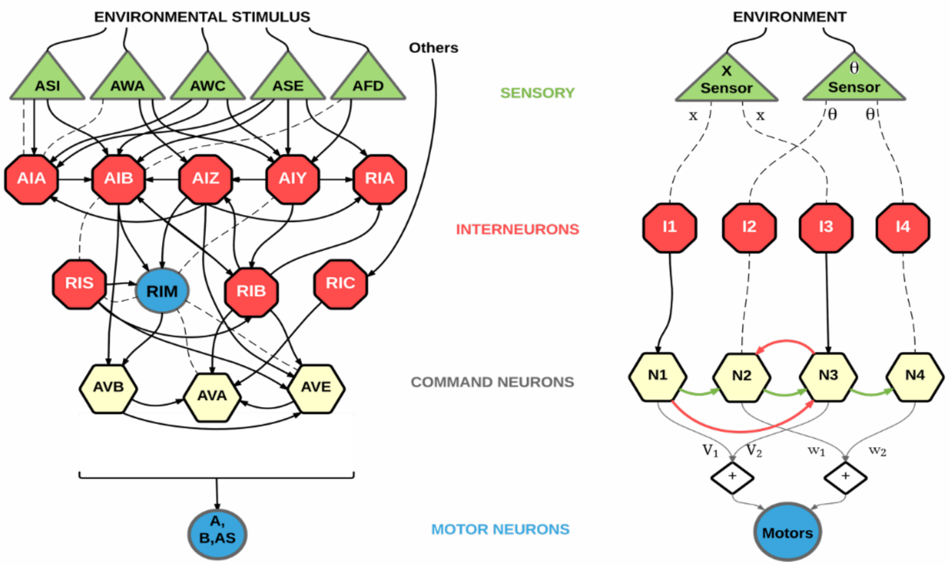
$$s = (x_0 > x_1, \sigma)\,?\,0 : 1$$

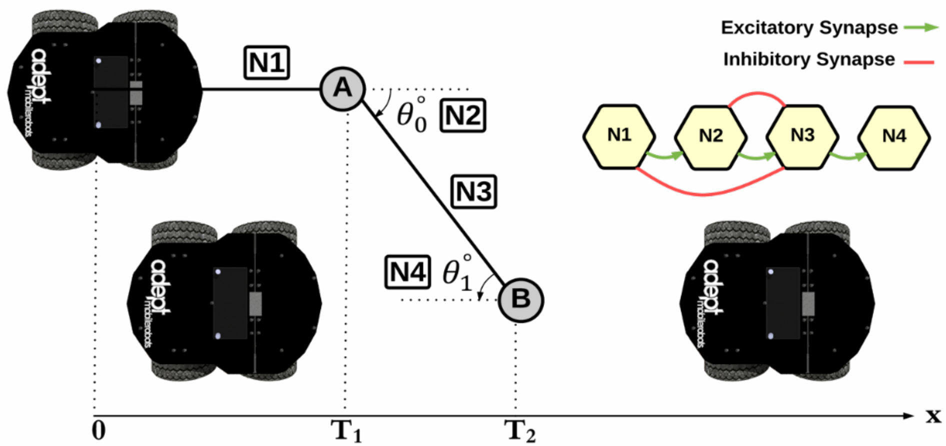
$$\Phi \mathrel{+}= s$$

}

Pioneer Rover with Neural Circuit Control

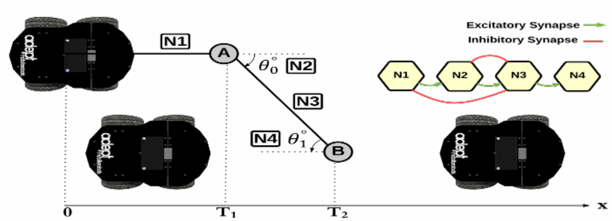
Automatic Parking inspired by part of the mechanosensory neural circuit



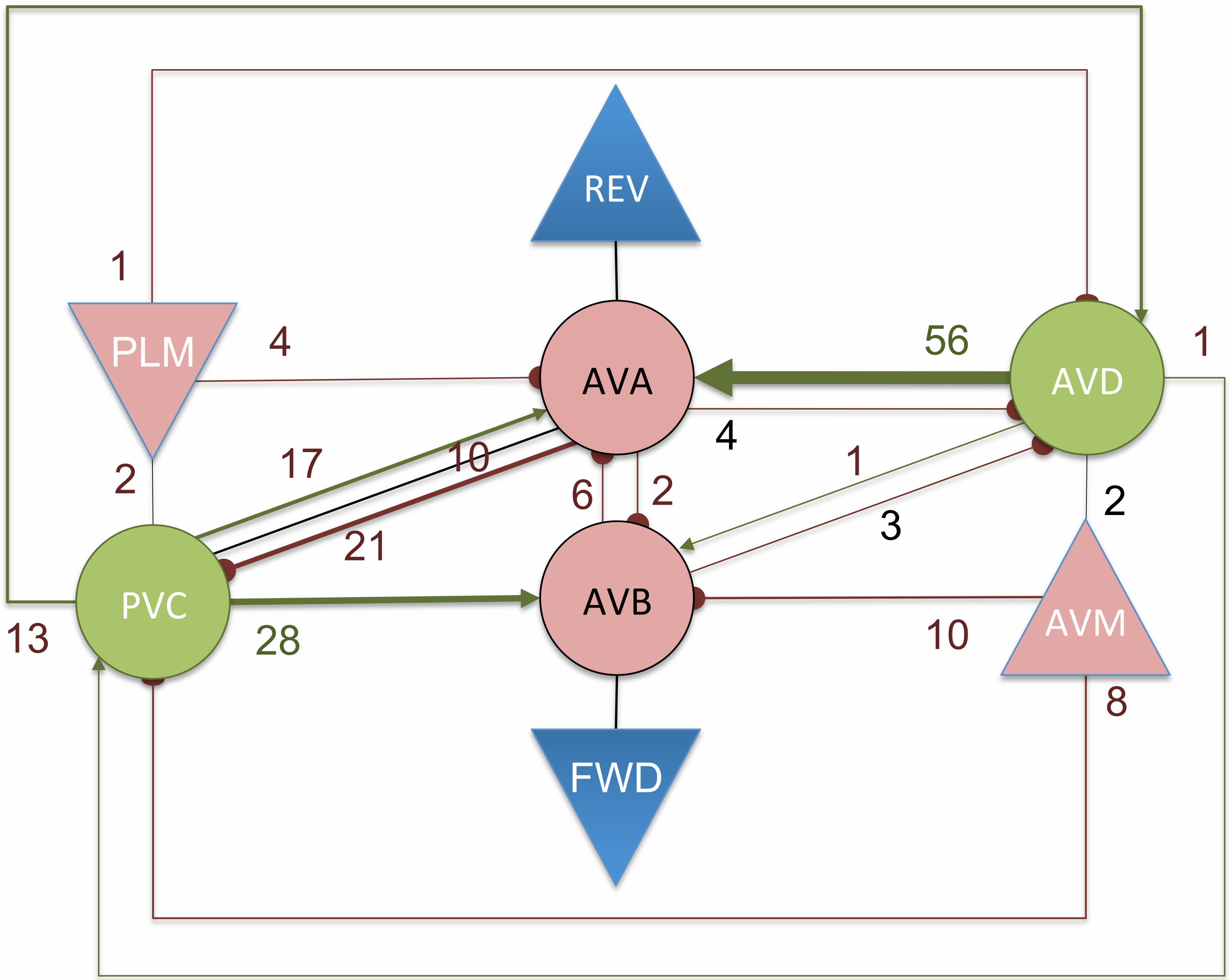


Pioneer Rover with Neural Circuit Control

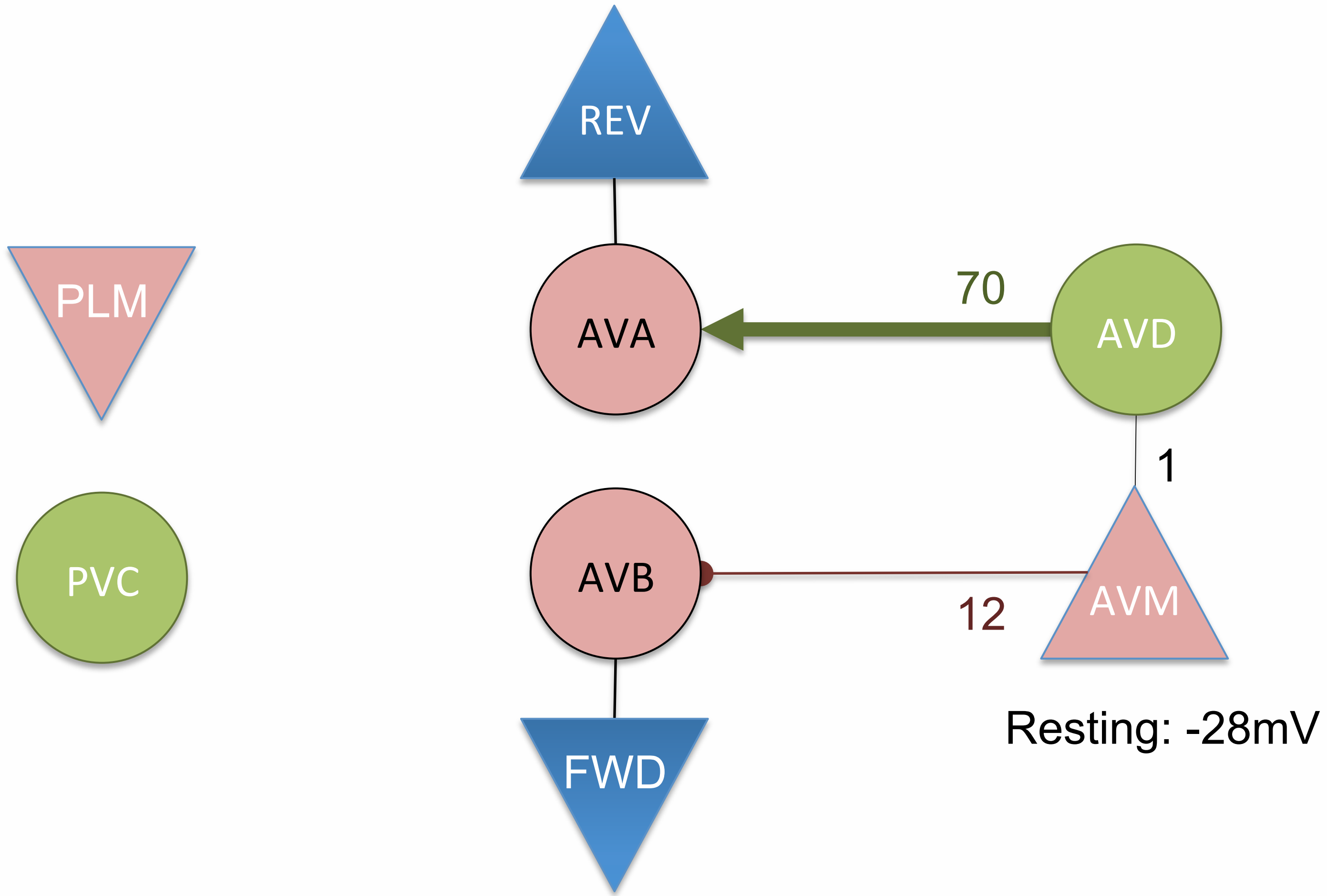
Automatic Parking inspired by part of the mechanosensory neural circuit



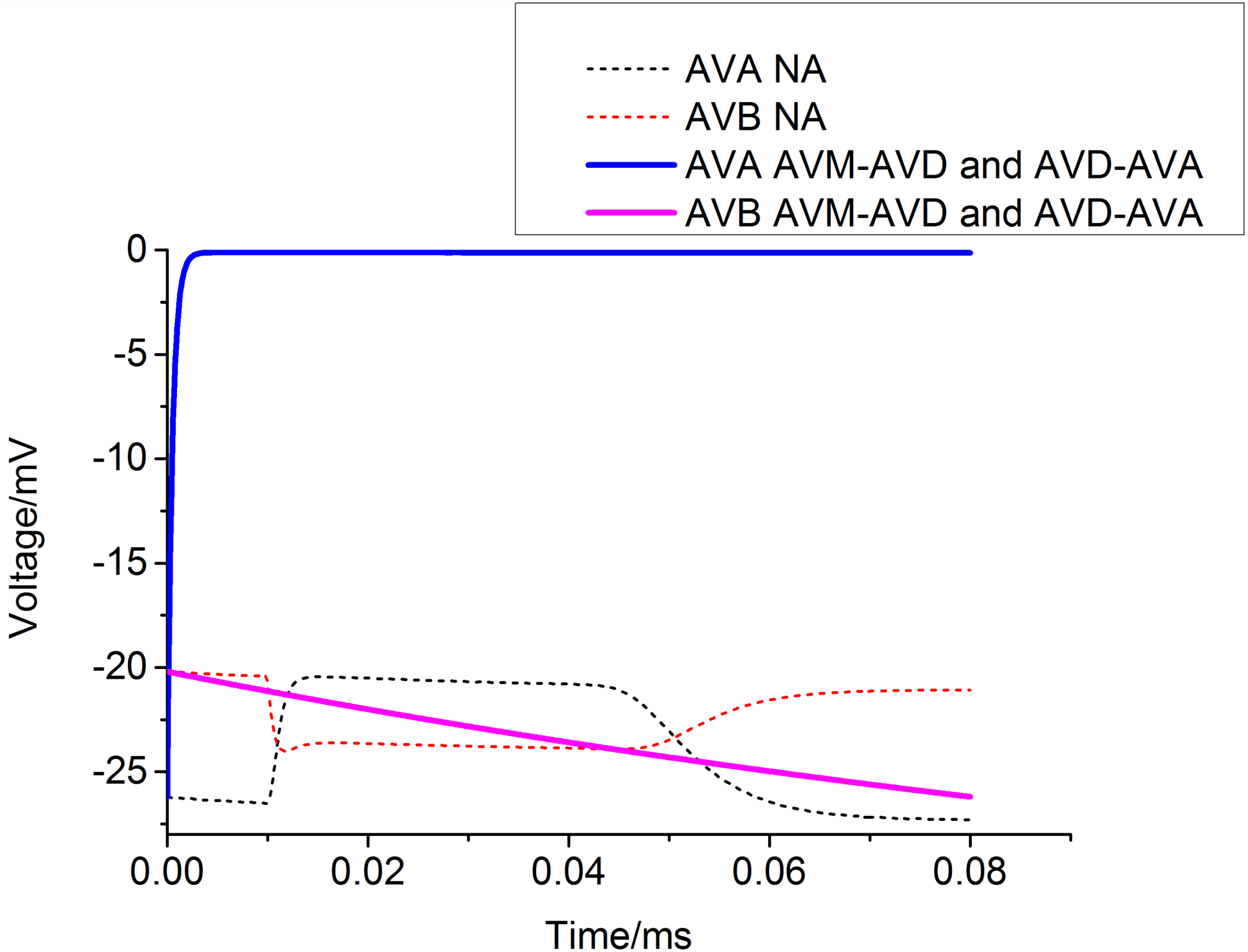
Why is the entire circuit so complicated?



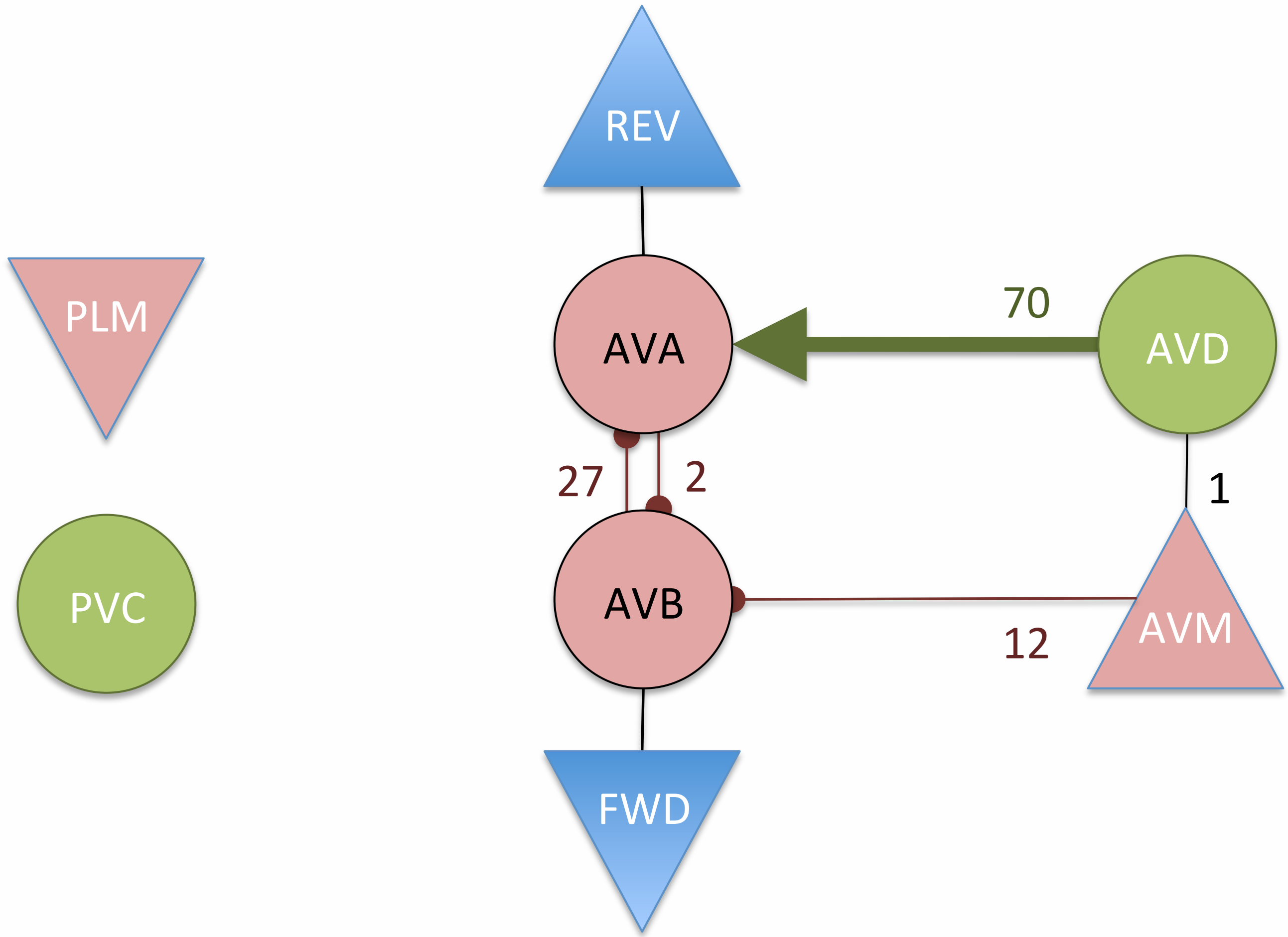
Why nature does not make it simple?



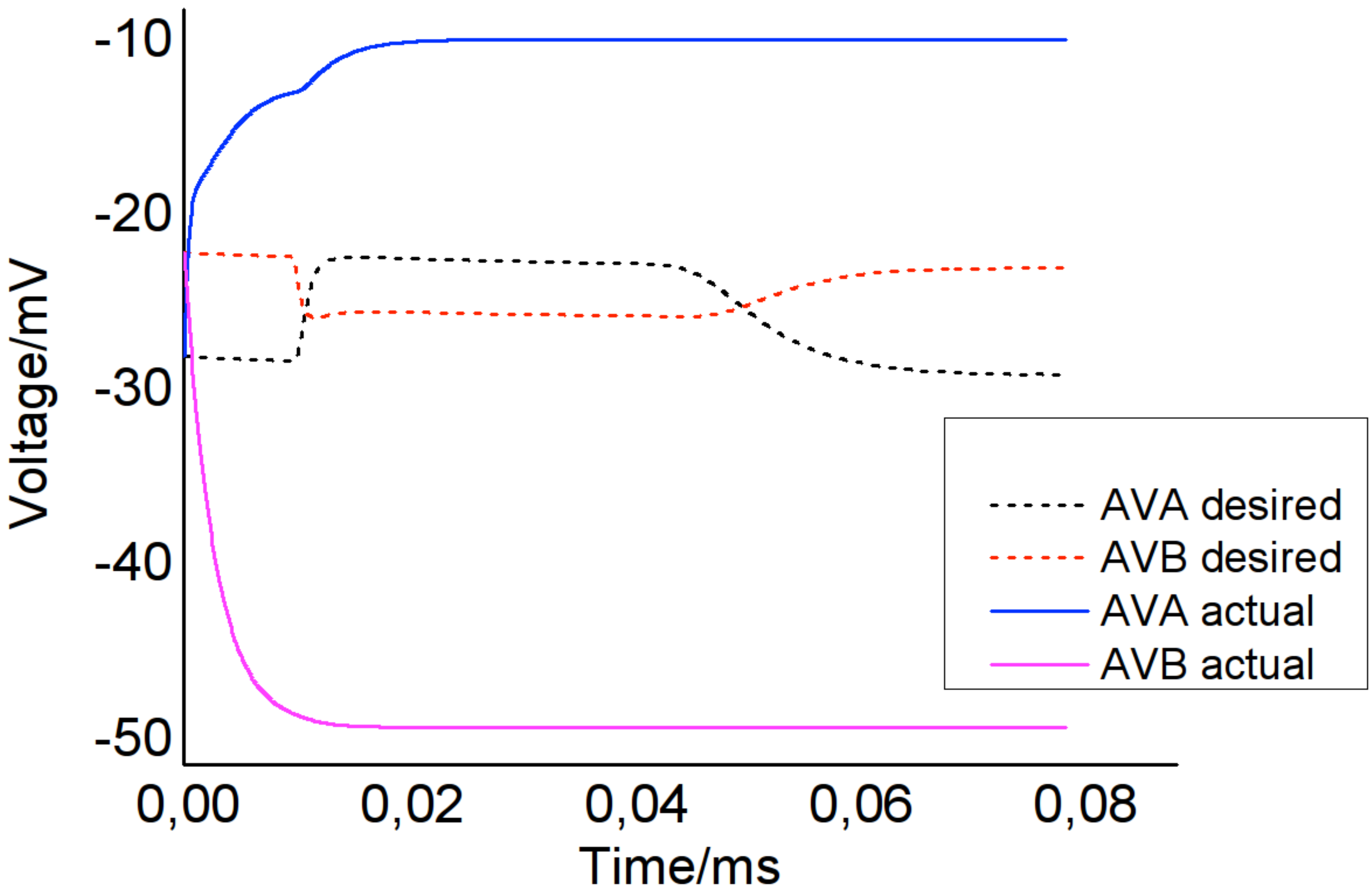
Why nature does not make it simple?



Cycles in Neural Circuit

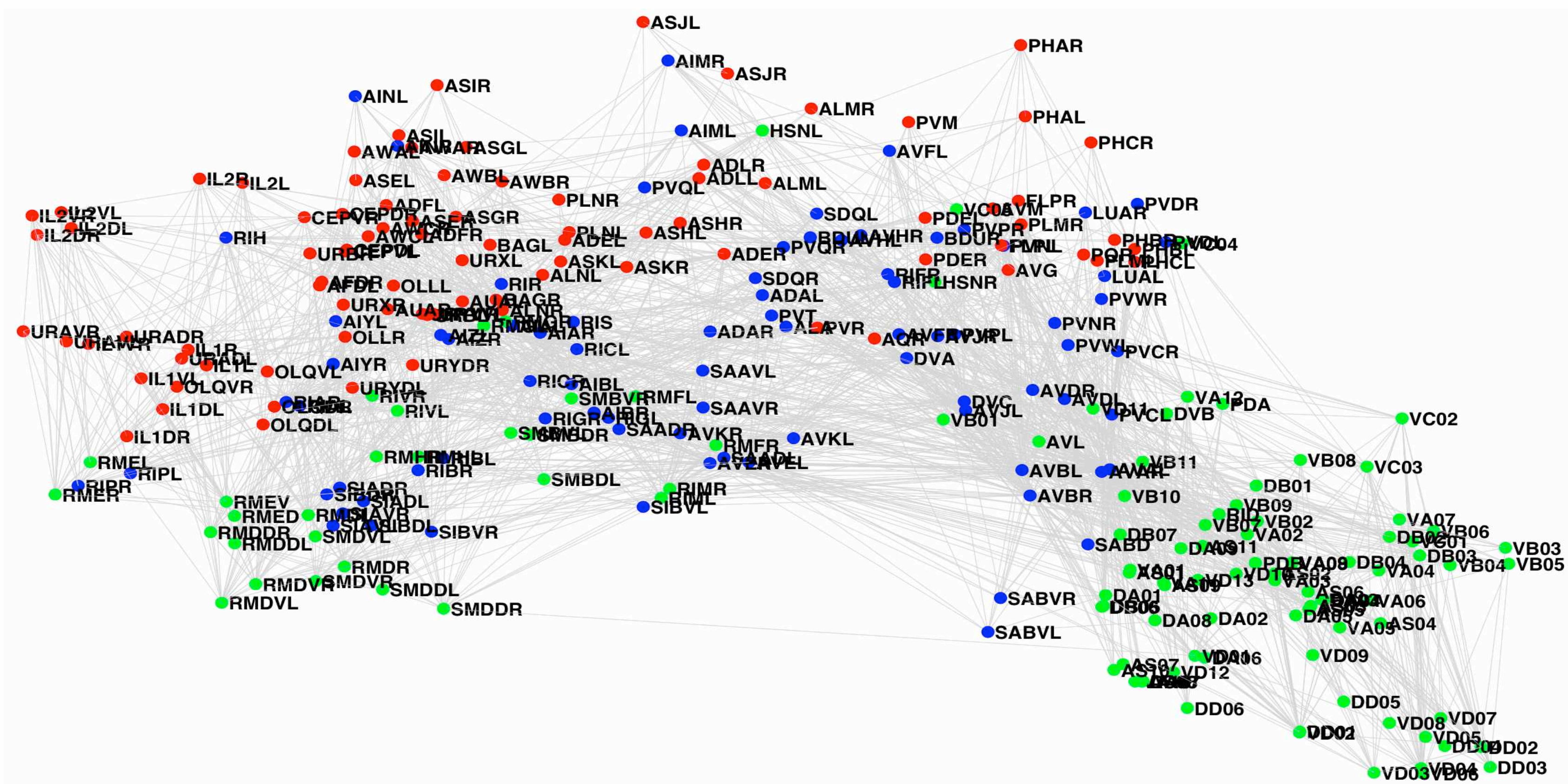


Why is the entire circuit so complicated?



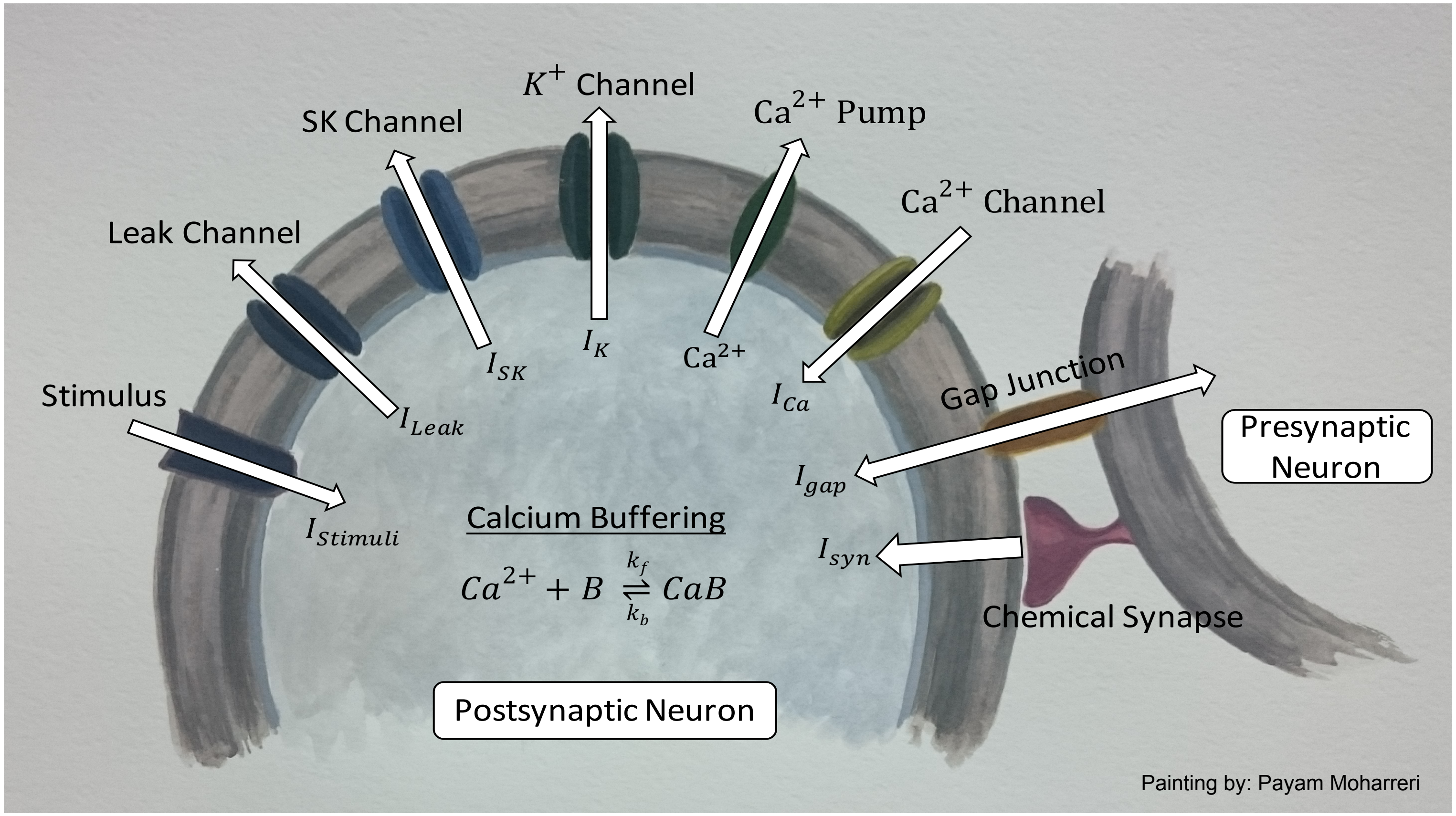
C.elegans Nervous System Modeling

Why is the nervous system of the nematode designed by nature the way it is?



White, John G., et al. (1986)

Model of a Neuron



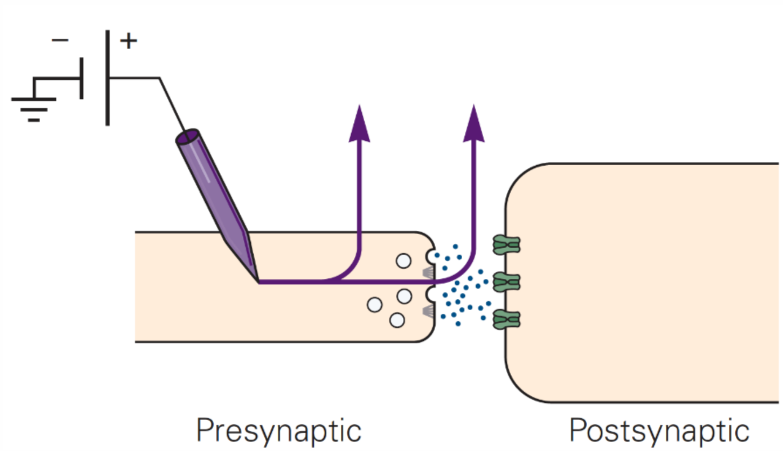
$$C_m \frac{dV}{dt} = -(I_{Ca} + I_K + I_{SK} + I_{Leak}) + \sum I_{Syn} + I_{gap} + I_{Stimu}$$

A. L. Hodgkin, et all. (1952), M. B. Goodman, et al. (1998), M. Kuramochi, et al. (2010)

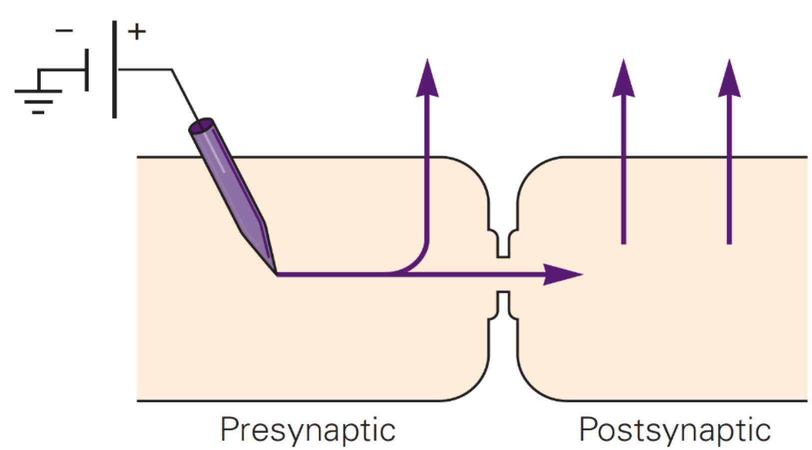
C.elegans Nervous System Modeling

Model of Synapses

Chemical Synapse

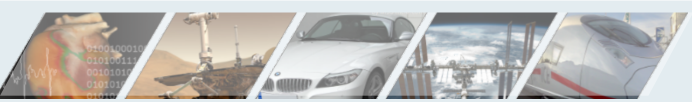


Gap Junction



$$I_{syn} = \frac{n^{ij} G_{syn}}{1 + e^{-(V_{Pre} - V_{shift})/V_{range}}} (E_{syn} - V_{post})$$

$$I_{gap} = n_{gap} G_{gap} (V_{pre} - V_{\ldots}$$

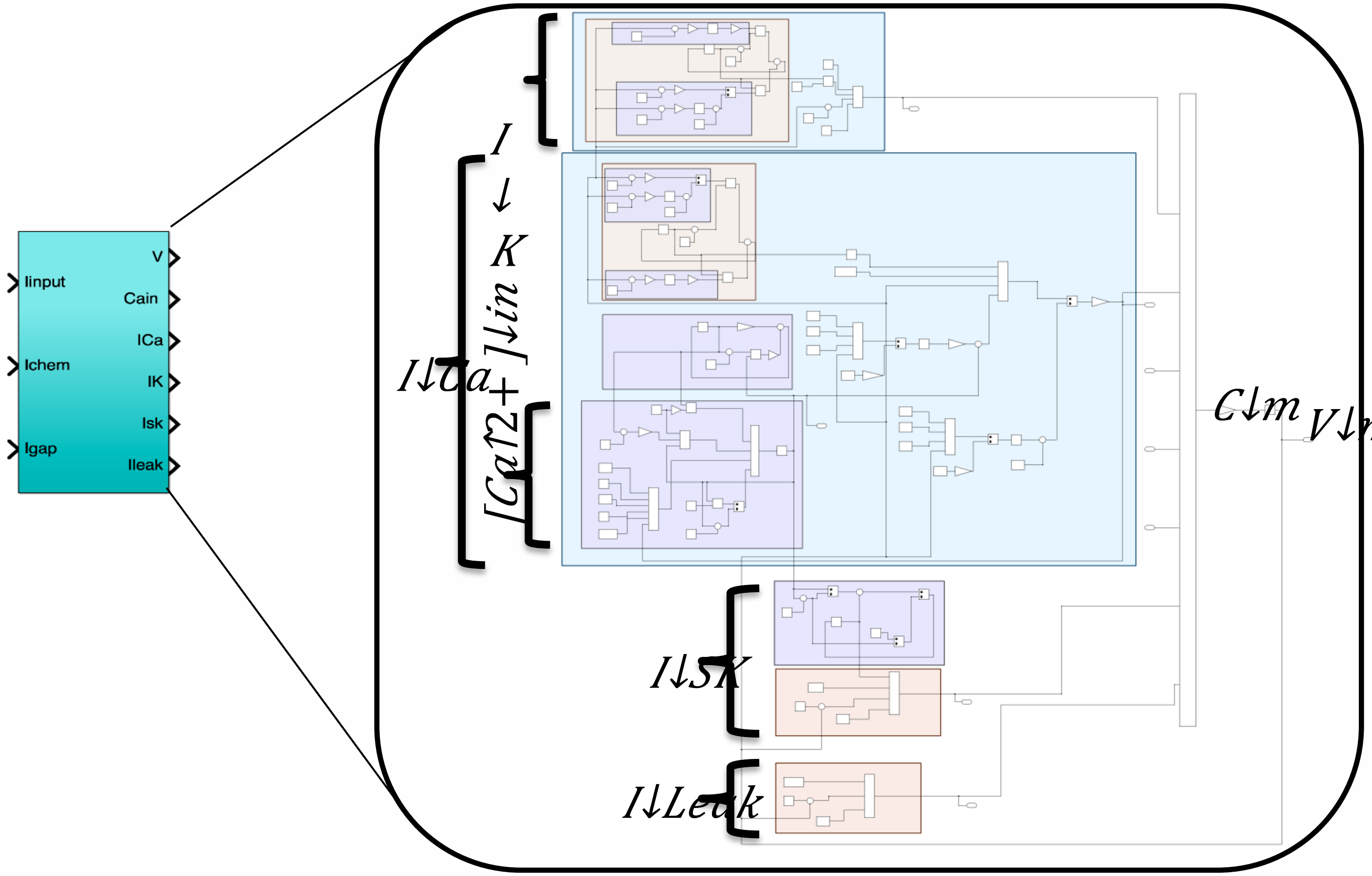


C.elegans Nervous System Modeling

Model Implementation

The neuron

Unpublished



$$C_m \, dV/dt = -(I_{Ca} + I_K + I_{SK} + I_{Leak}) + \sum I_{Syn} + I_{gap} + I_{Stimuli}$$

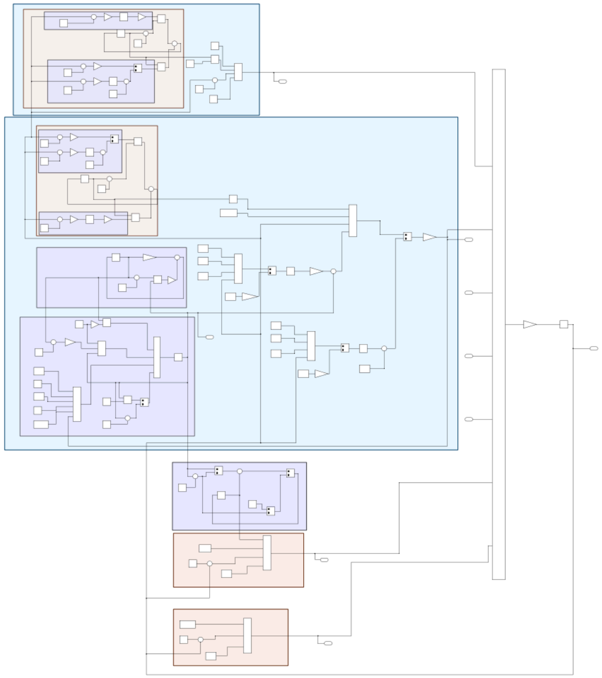
Model Implementation

The neuron

Unpublished

Features:

- ✓ Monitoring the dynamics of every single ion channel current together with its parameters and specially **observing dynamics of intracellular calcium concentration of a neuron**.
- ✓ Easy access to the channel parameters such as **Ionic conductance**, equilibrium potential of channels, gate rate functions, time constant of the activation and inactivation of a gate.
- ✓ Easy access to the **membrane capacitance** and resting potential of the neuron.
- ✓ capable of adding **stochasticity** to the system.

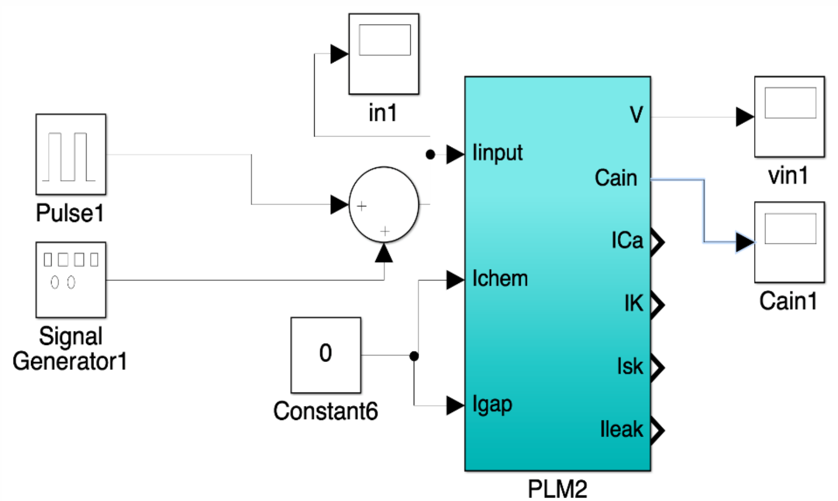


Model Implementation

The neuron – Response of a Single Neuron

Unpublished

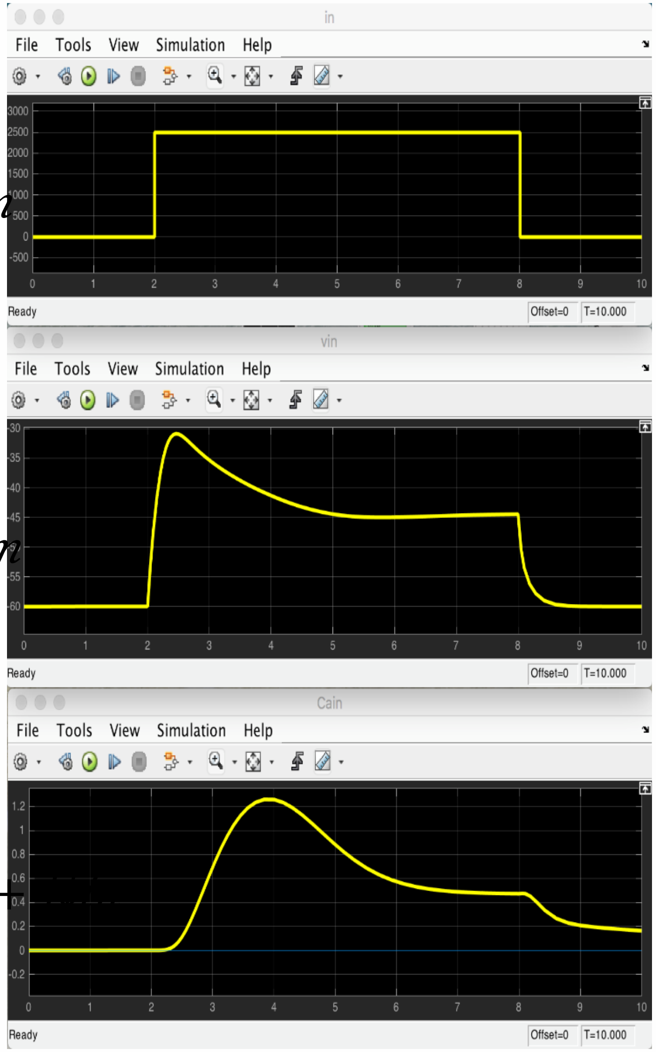
Applying current stimuli to a single neuron and observing its membrane potential and its intracellular Calcium concentration



I_{in}

V_m

$[Ca^{2+}$



Model Implementation

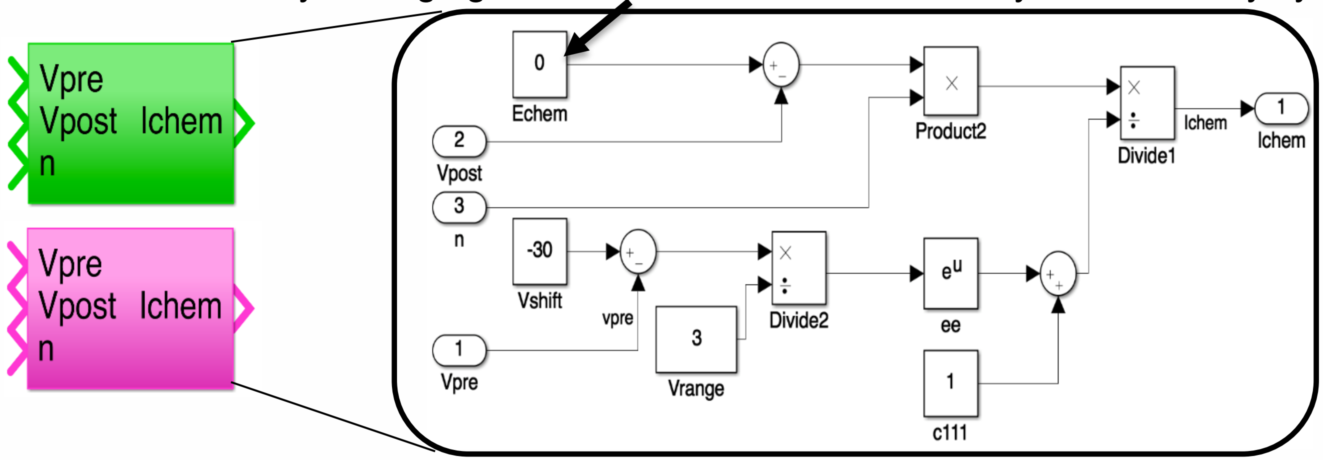
Synapses

Unpublished

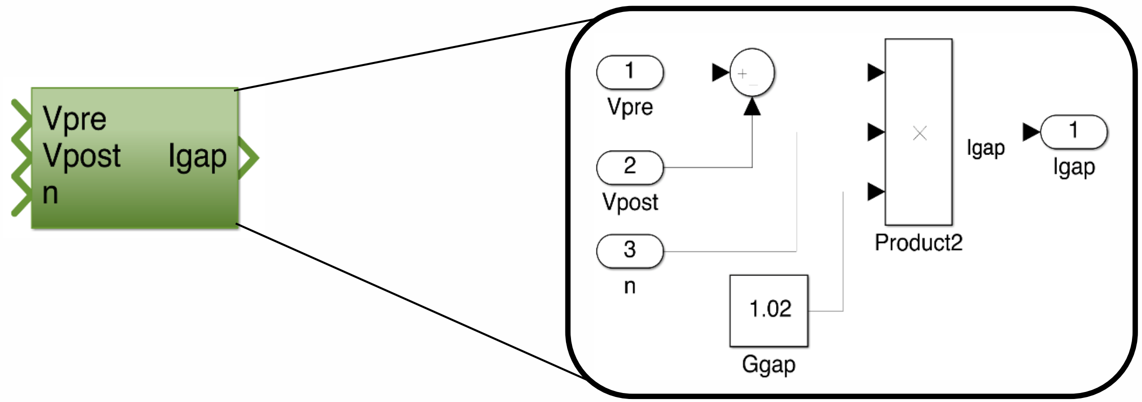
By Changing E_{chem} we can set excitatory and inhibitory synapses

Excitatory Synapse

Inhibitory Synapse



Gap Junction



Model Implementation

Synapses

Unpublished

Features:

- ✓ Monitoring input voltage and output current of a synapse

- ✓ Easy access to the parameters of synapses such as: Number of synaptic connections between two neurons, Synaptic weight, Shift and range voltages of the synapse.

- ✓ Set the level of excitation and inhibition of a chemical synapse by varying E_{chem}.

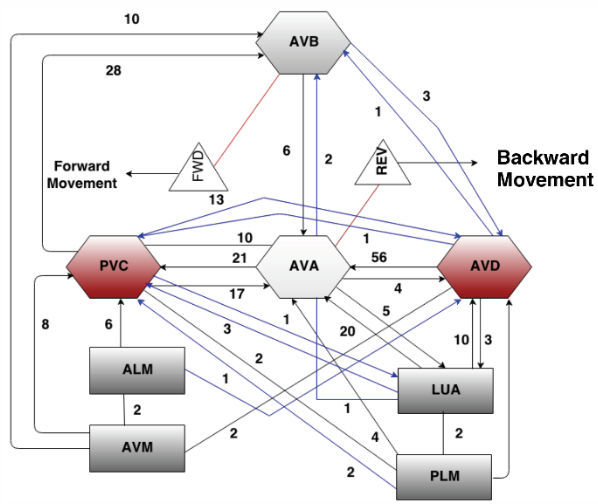
C.elegans Nervous System Modeling

Neural Circuit Implementation

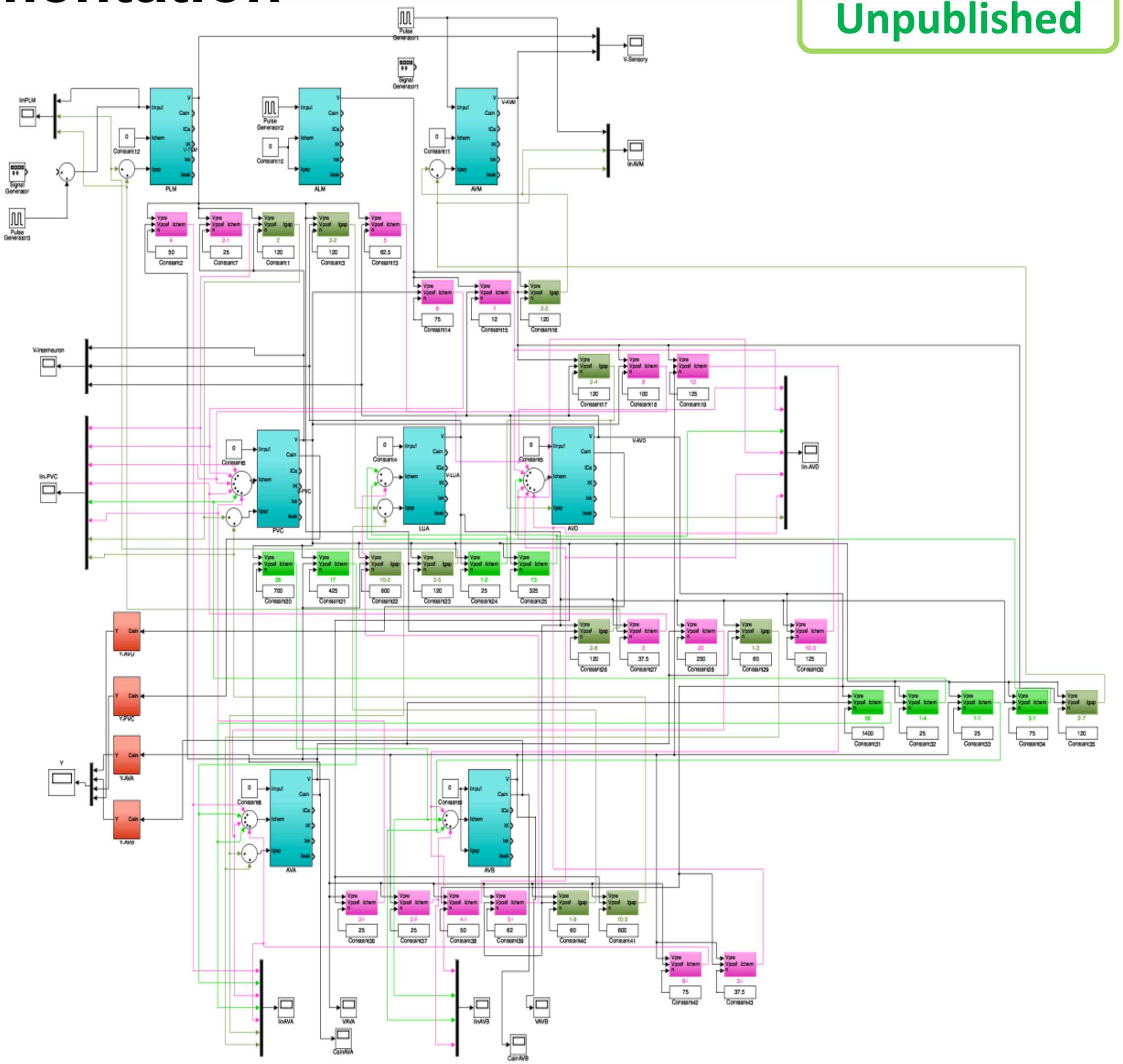
Tap-Withdrawal Circuit

Unpublished

A **Simulink** tool for implementing neural circuits has been developed.



www.wormweb.org



C.elegans Nervous System Modeling

Neural Circuit Implementation

Tap-Withdrawal Circuit

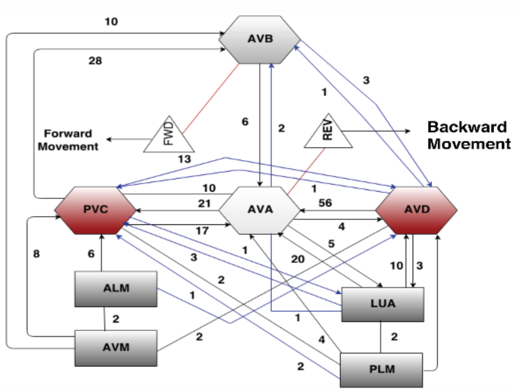
Unpublished

AV

AV

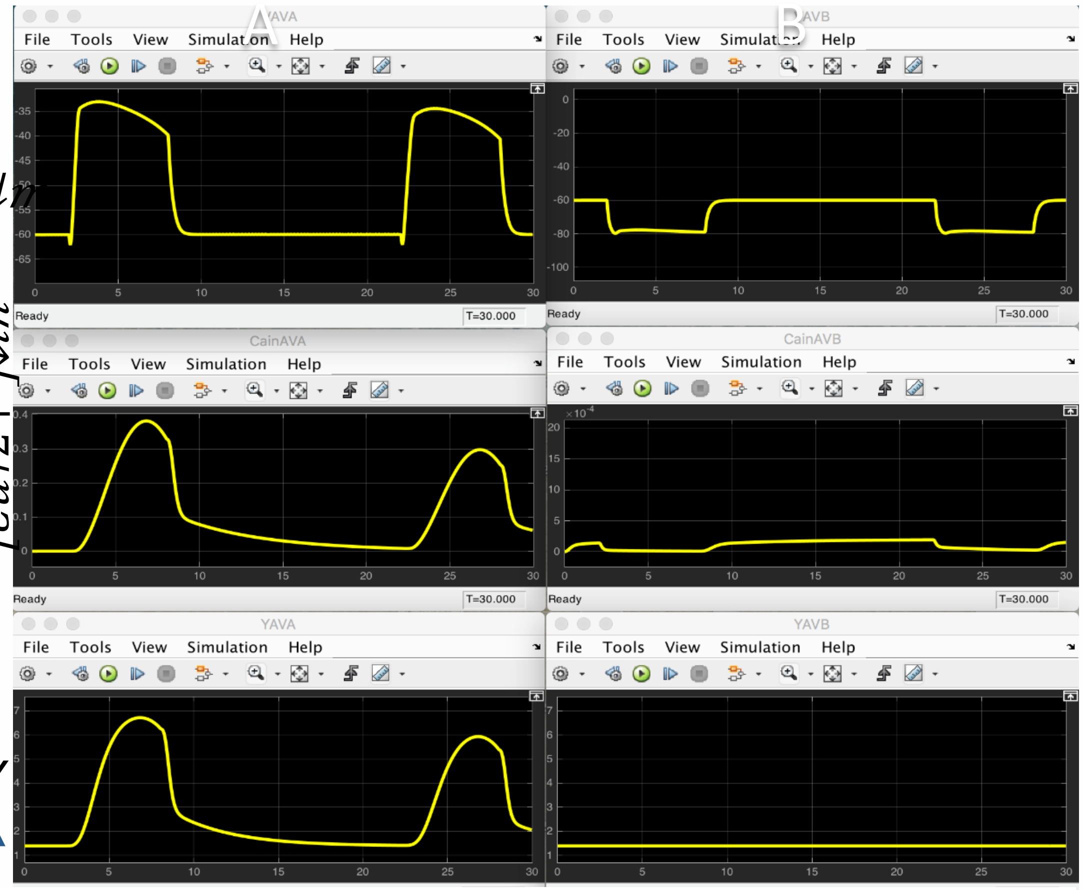
AVM

I_{in}



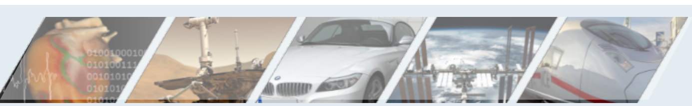
V_{in}

$[Ca^{2+}]_{in}$



Y

fluorescence intensity

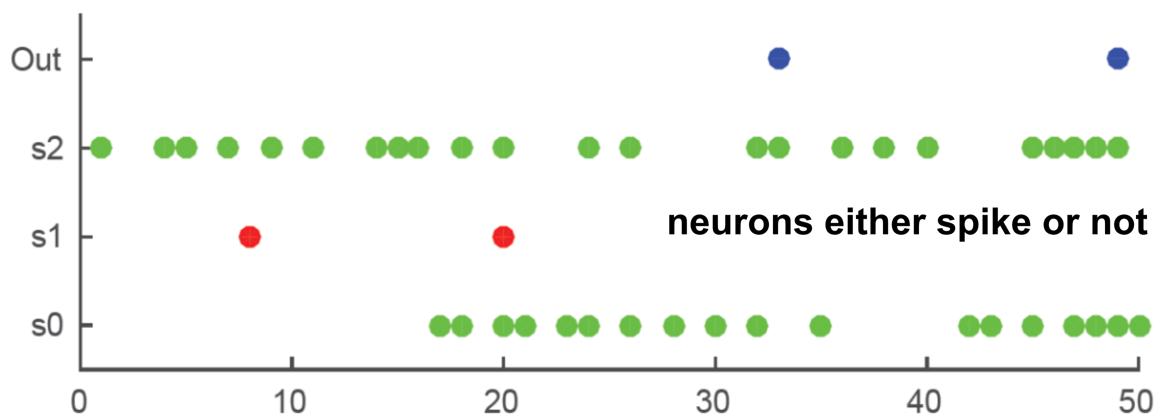


Spiking Neurons in Hardware (TrueNorth)

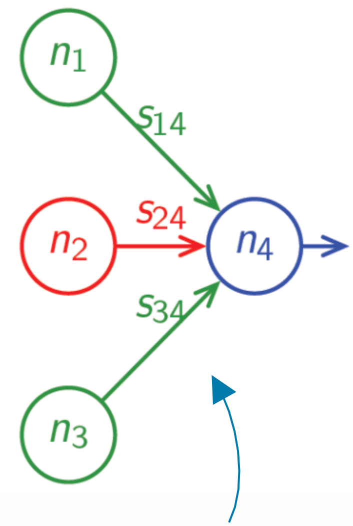
How to capture leaky-integrate-and-fire (LIF) behavior in hardware?

TrueNorth Neural Model from IBM:

- developed specifically for hardware implementation
- does not use floating point computations
- extends the LIF model



(Cassidy et. al. 2013)



only synaptic connections

TrueNorth: Extension of LIF Model

Synaptic Integration:

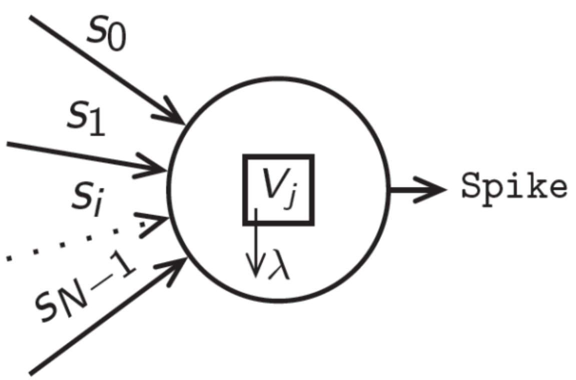
- Take into account outputs of other neurons

Leak Integration

- Model energy loss over time (absence of input)

Threshold, Fire, Reset:

- Fire a spike if membrane potential exceeds threshold



TrueNorth: Extension of the LIF Model

Synaptic Integration:

- Weighted sum of inputs / probabilistic sum
- Integer weights

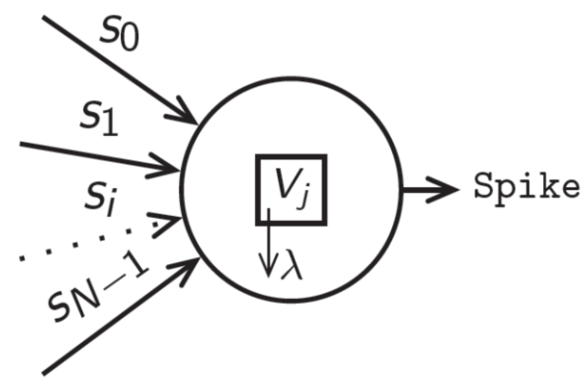
probabilistic flag

random sample

$$V_j(t) \leftarrow V_j(t-1) + \sum_j in_j \left[(1-b_j) w_j + b_j \cdot F(w_j, \rho) \cdot \text{sgn}\left(w_j\right) \right]$$

Membrane Potential (integer)

weights



1st step of Neuron Computation

(Cassidy et. al. 2013)

TrueNorth: Extension of the LIF Model

Leak Integration:

- **Standard / leak reversal mode**
- **Energy loss captured by leak weight**

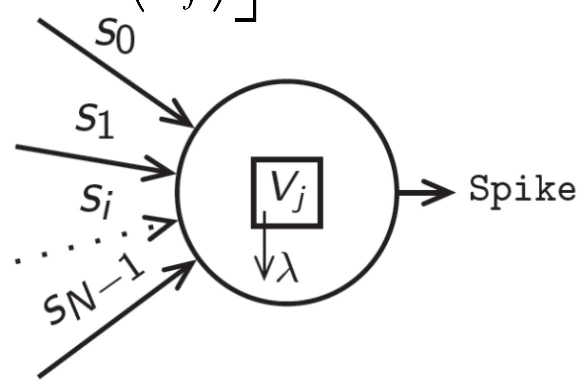
leak direction **leak reversal flag** **leak weight**

$$\Omega \leftarrow (1-\varepsilon_j) + \varepsilon_j \cdot \operatorname{sgn}\left(V_j(t)\right)$$

random sample

$$V_j(t) \leftarrow V_j(t) + \Omega\left[(1-c_j)\cdot\lambda_j + c_j \cdot F(\lambda_j,\rho)\cdot \operatorname{sgn}\left(\lambda_j\right)\right]$$

probabilistic mode flag



2nd step of Neuron Computation

(Cassidy et. al. 2013)

TrueNorth: Extension of the LIF Model

mask (if 0 computation is deterministic)

$$\eta_j \leftarrow \rho \,\&\, M$$

Stochastic component

$$\boxed{if \quad V_j(t) \geq \alpha_j + \eta_j}$$

positive threshold

$$SPIKE$$

rest value

$$V_j(t) \leftarrow \delta(\gamma_j)R_j(t) + \delta(\gamma_j - 1)\cdot\left(V_j(t) - (\alpha_j + \eta_j)\right) + \delta(\gamma_j - 2)\cdot V_j(t)$$

negative threshold

$$\boxed{elseif \quad V_j(t) < -\left[\beta_j\kappa_j + \left(\beta_j + \eta_j\right)(1 - \kappa_j)\right]}$$

$$V_j(t) \leftarrow \boxed{-\beta_j\kappa_j} + [-\delta(\gamma_j)R_j(t) + \delta(\gamma_j - 1)\cdot\left(V_j(t) - (\beta_j + \eta_j)\right) + \delta(\gamma_j - 2)\cdot V_j(t)](1 - \kappa_j)$$

Threshold, Fire, Reset:

- Positive / negative thresholds
- Reset modes:
  - Normal $\gamma = 0$
  - Linear $\gamma = 1$
  - Non-reset $\gamma = 2$
- Reset / saturate behavior

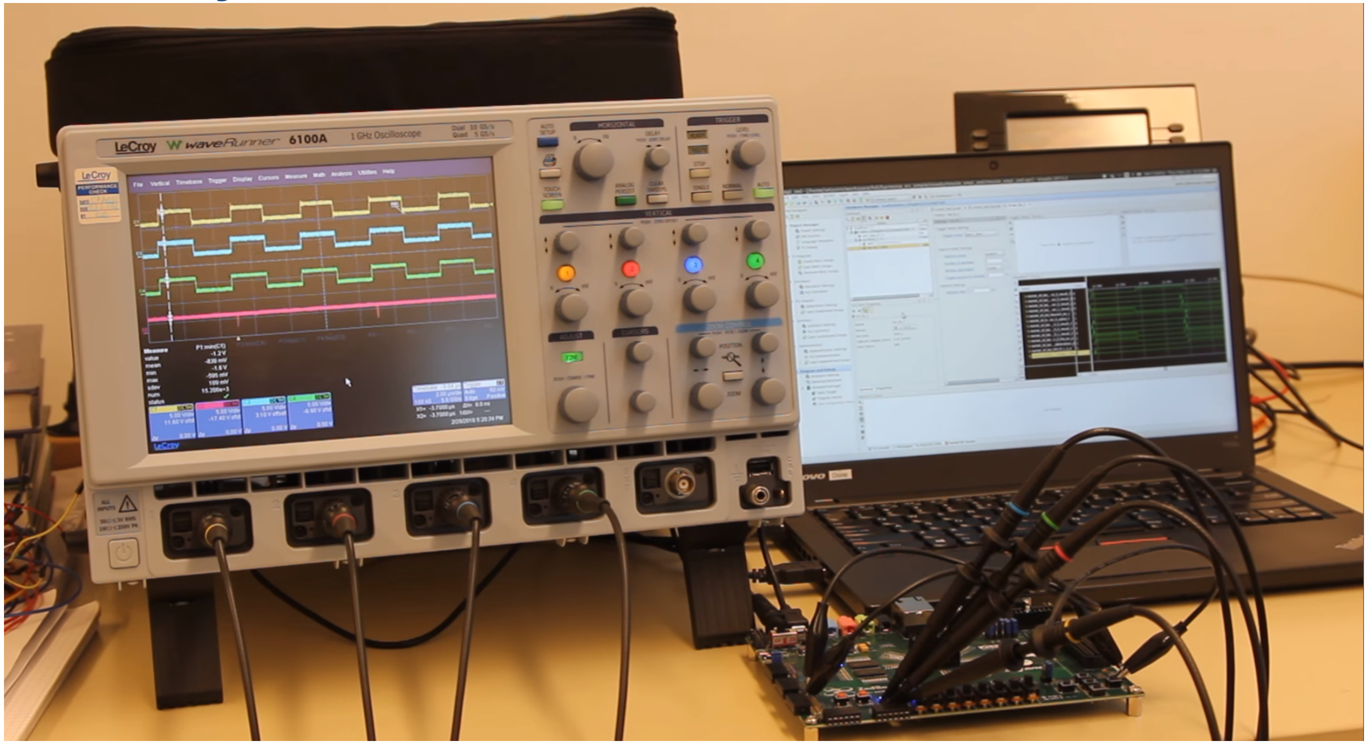
3rd step of Neuron Computation

(Cassidy et. al. 2013)

TrueNorth: Extension of LIF Model

Given

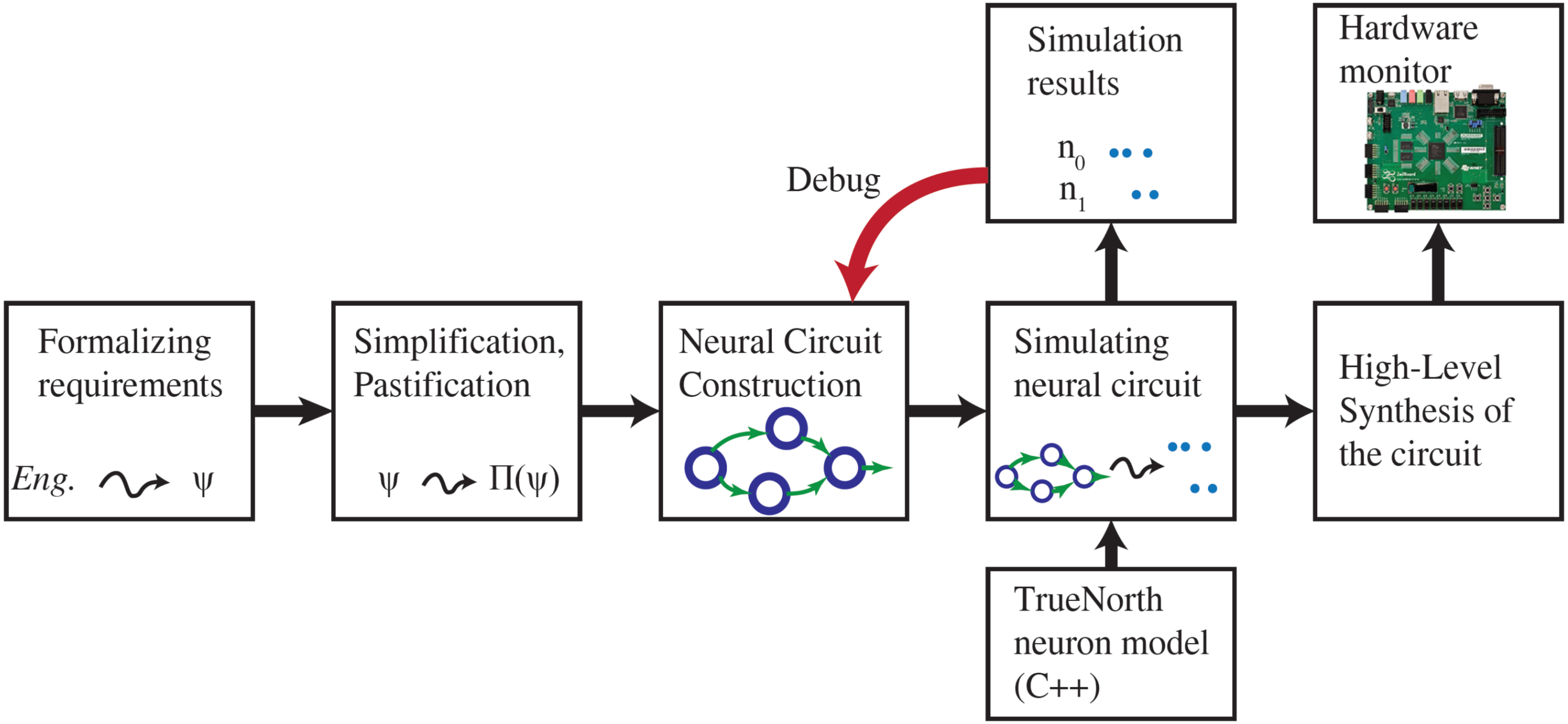
- TrueNorth Neuron Model
- MTL specification φ
- A run of a system



Build

Runtime monitor that checks φ using the TrueNorth model

Building Hardware Monitors with Neurons

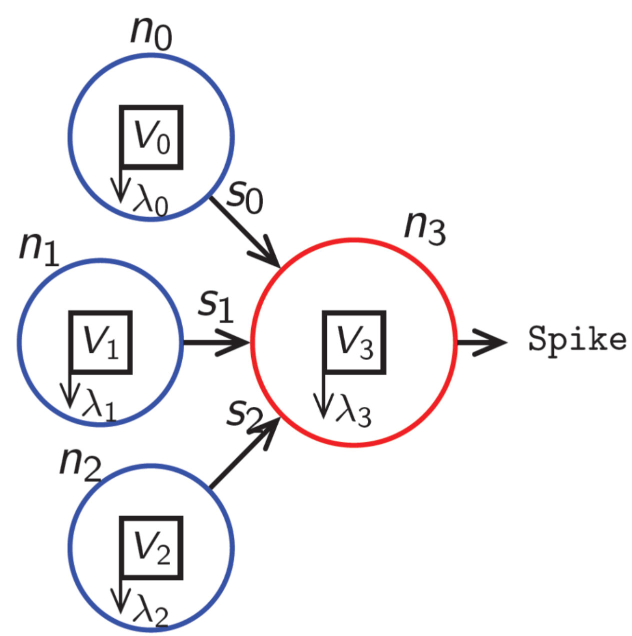


Logical Operations with TrueNorth

Combinatorial behavior with TrueNorth Model:

- Impose memoryless behavior
- Find parameter values of neurons
- AND, OR, NOT, NAND, NOR require one neuron
- Example: find parameters for 3-AND

- n3 must fire only when n0, n1, and n2 fire
- Always reset n3 after computation (memoryless)
- Finding parameters can be stated as ILP



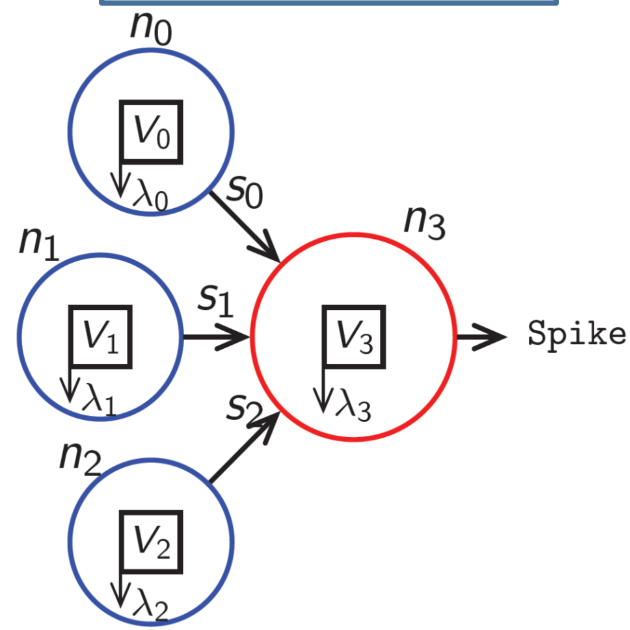
Logical Operations with TrueNorth

ILP: x - integers
min Cx
s.t.
Ax<= B

min 0 (i.e. find feasible solution)
s.t.

No spike & Reset

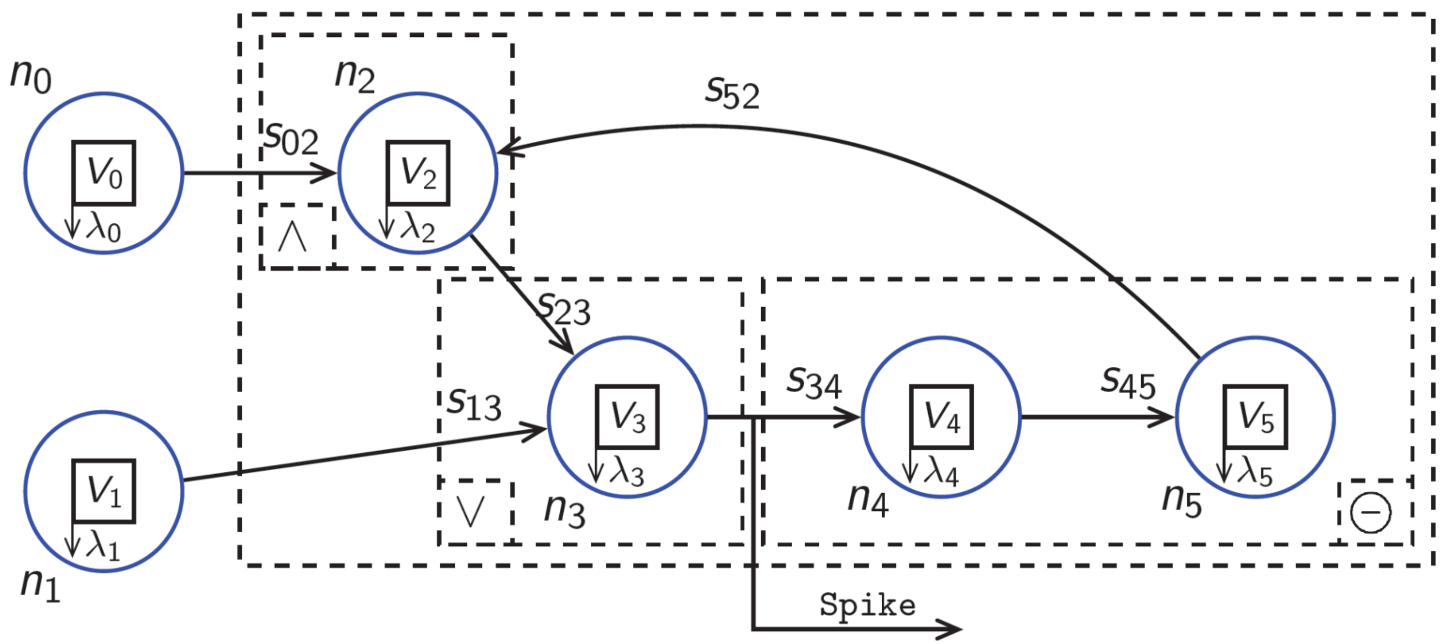
$$
\begin{array}{ccccccc}
 & & & +\lambda_3 & < & \beta \\
 & & +s_2 & +\lambda_3 & < & \beta \\
 & +s_1 & & +\lambda_3 & < & \beta \\
 & +s_1 & +s_2 & +\lambda_3 & < & \beta \\
+s_0 & & & +\lambda_3 & < & \beta \\
+s_0 & & +s_2 & +\lambda_3 & < & \beta \\
+s_0 & +s_1 & & +\lambda_3 & < & \beta
\end{array}
$$



Temporal Operations with TrueNorth

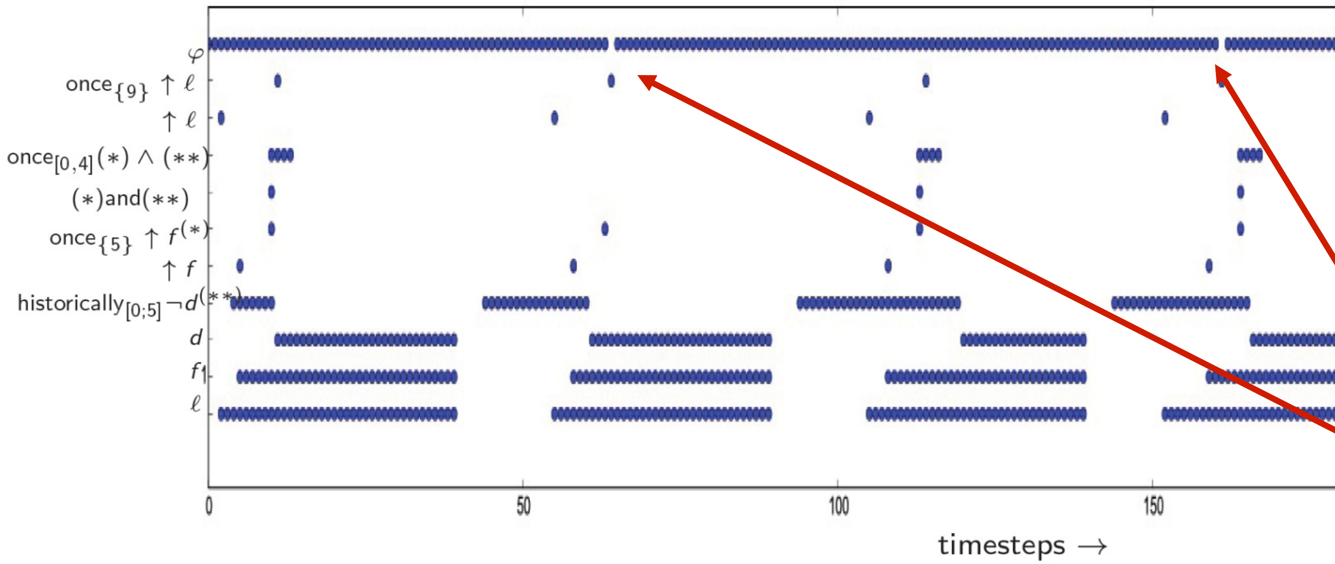
Past STL with TrueNorth Model:

- Constraints analogous to previous (details in the paper)
- Composition of combinational & temporal operators
- Example: "n_1 since n_0" circuit with TrueNorth



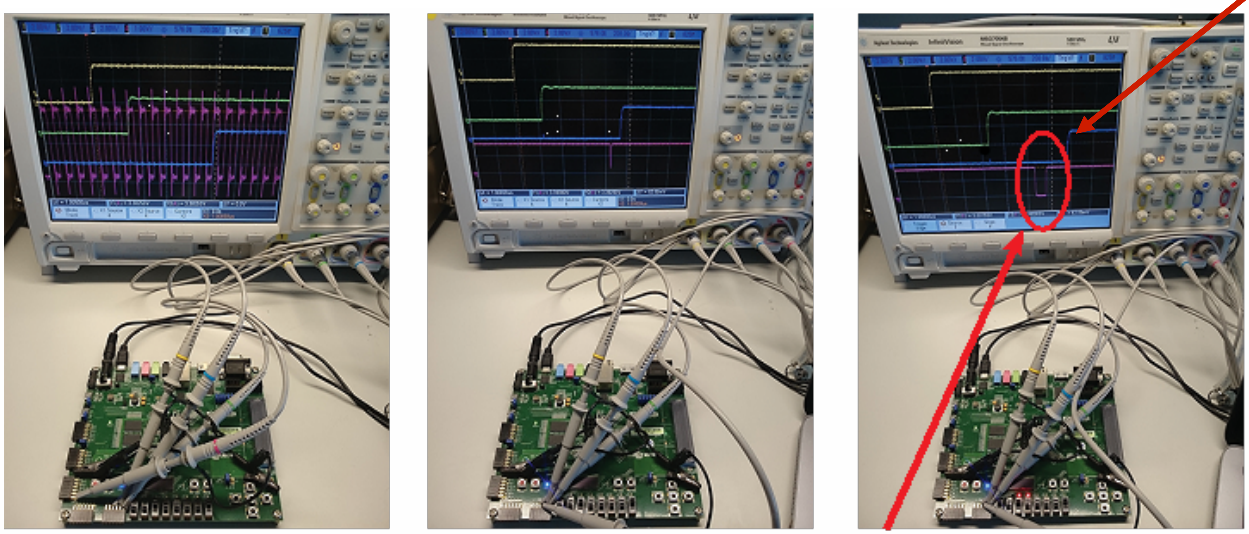
"Monitoring of MTL Specifications with IBM's Spiking Neural Model", DATE 2016

MTL Monitoring with TrueNorth in Action



Simulation

Violation detection



Hardware

Deep Learning Solutions for Integrated Circuits

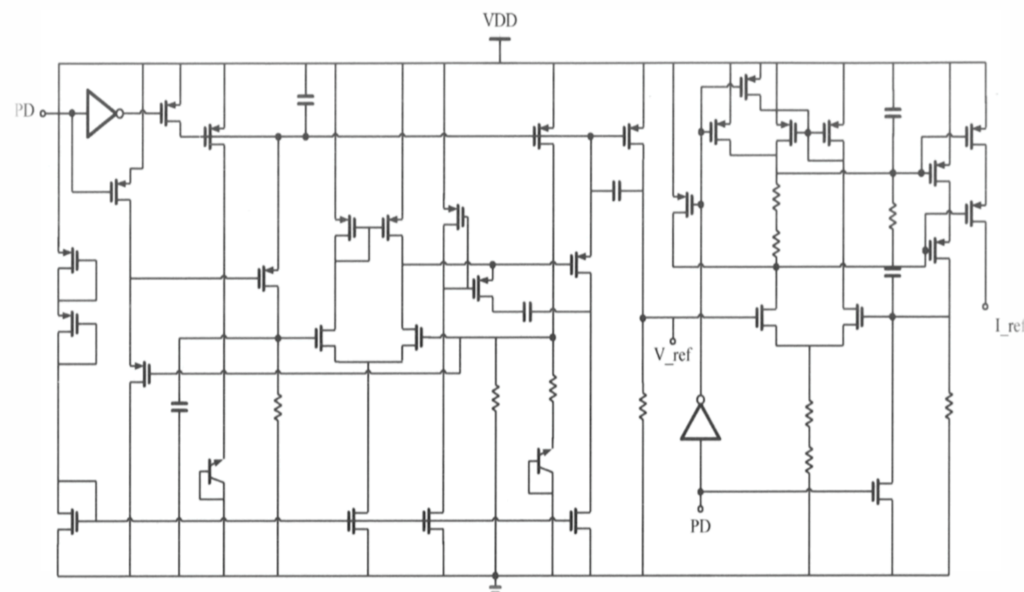
Efficient Modeling of a CMOS Band-Gap Reference Circuit Using Neural Networks

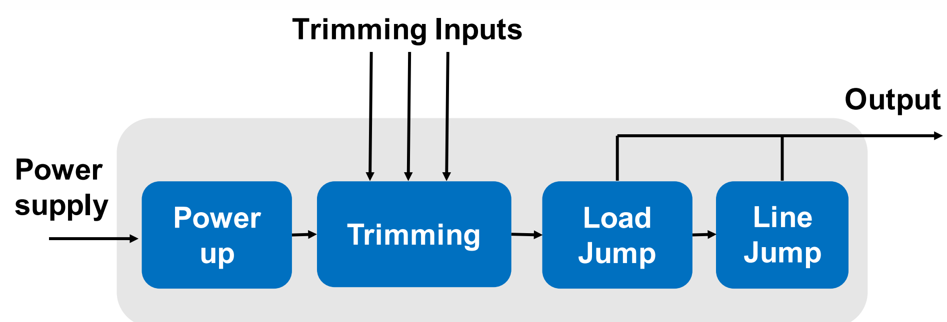
Band-Gap:
- Provides a constant 1V at its output

Trimming:
- 3 digital inputs, 8 possible output values ranging from 0.8-1.2

Load Jump:
- variation of output in case of load

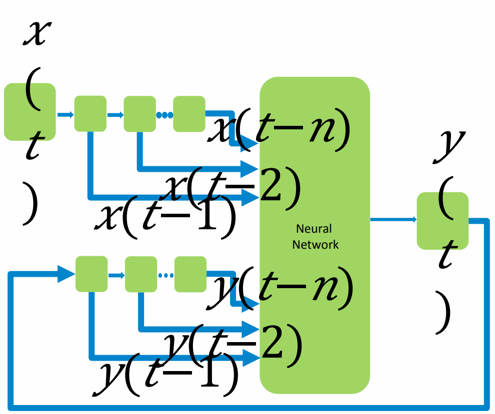
Line Jump:
- change of the output as a result of variations on the power supply

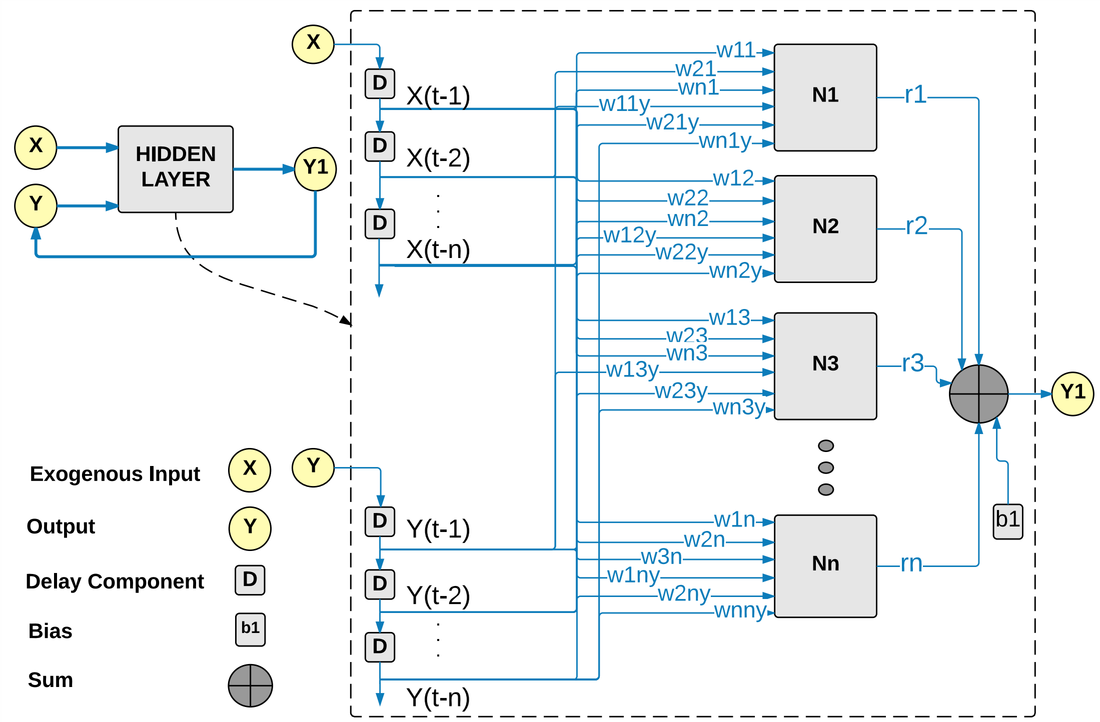




Deep Learning Solutions for Integrated Circuits

Non-Linear Auto-Regressive Neural Network with exogenous Input (NARX)

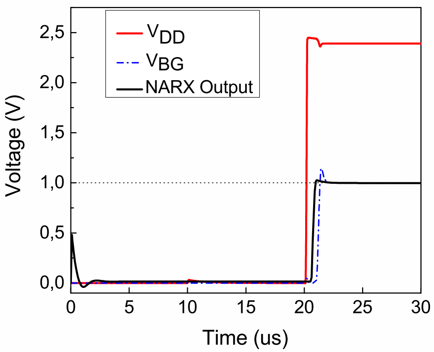




Deep Learning Solutions for Integrated Circuits

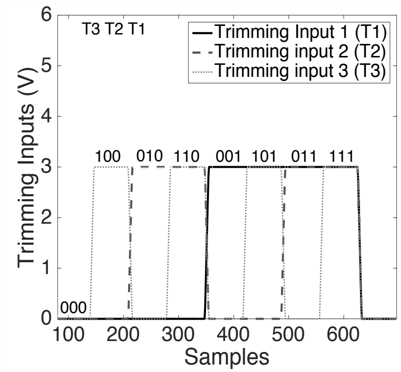
Response of the designed NARX NN behavioral Models

Power-Up

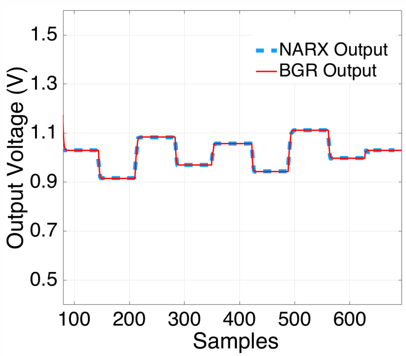


Input & Output

Trimming

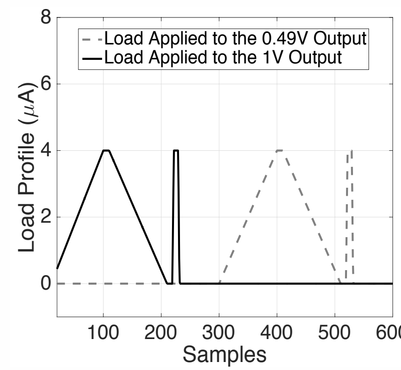


Input

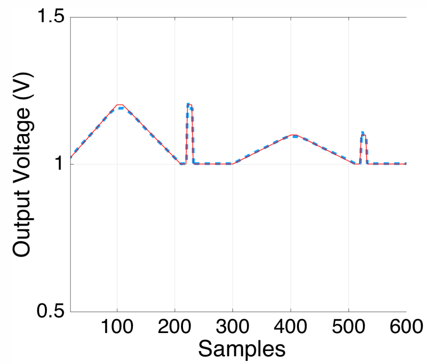


Output

Load Jump

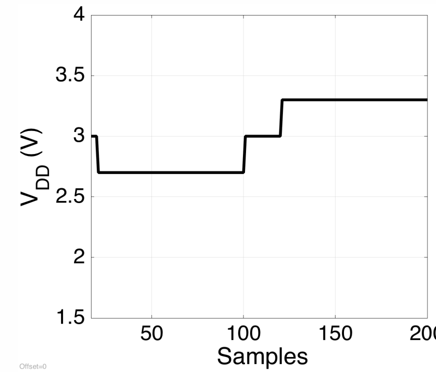


Input

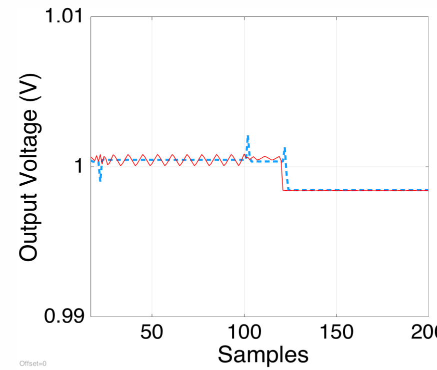


Output

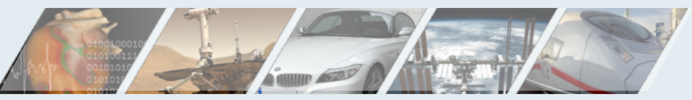
Line Jump



Input



Output



Deep Learning Solutions for Integrated Circuits

2-Layer Stack Neural Network

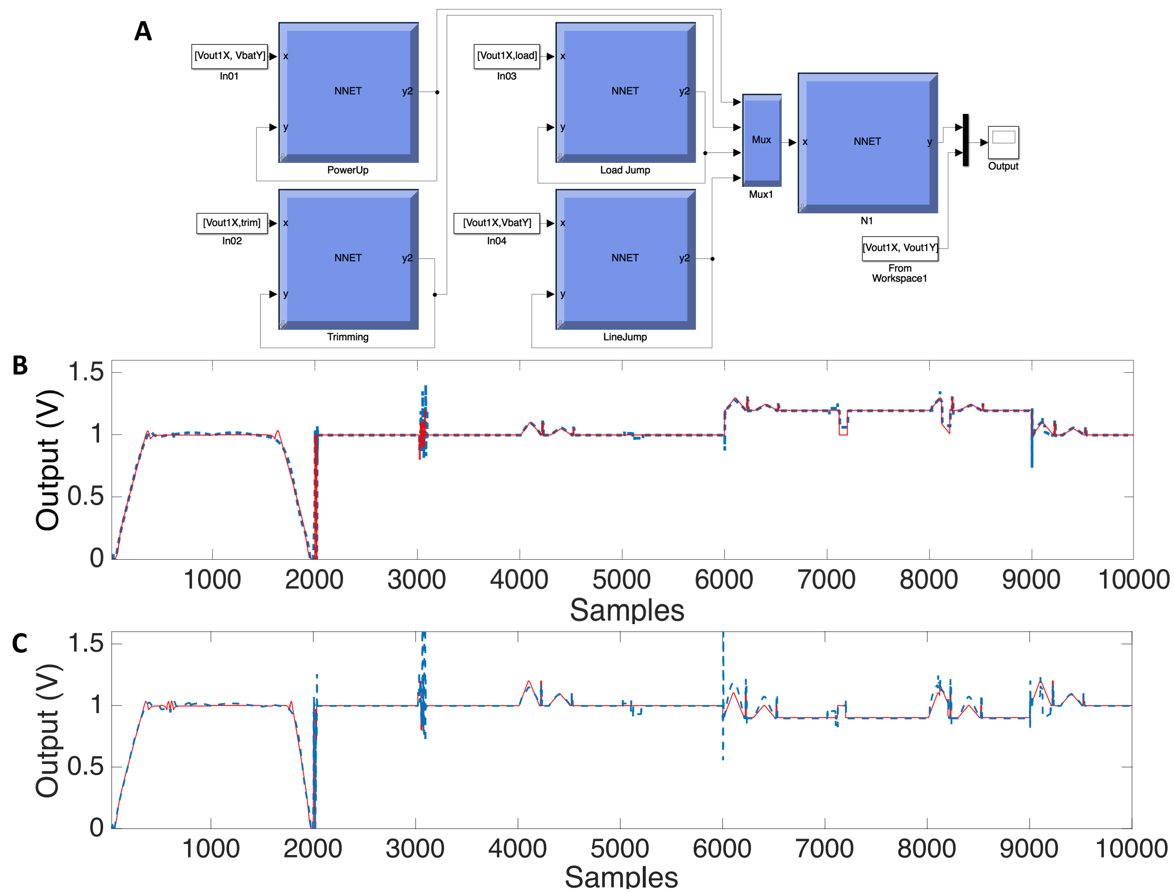
Unpublished

Combining the developed behavioral models

Network Architecture

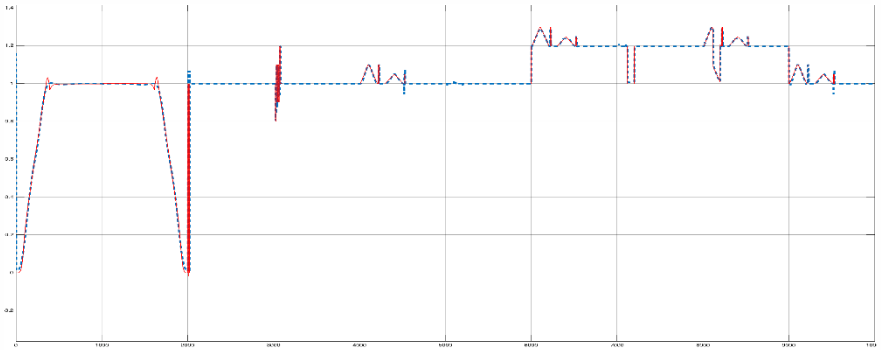
Training dataset response

Test dataset response

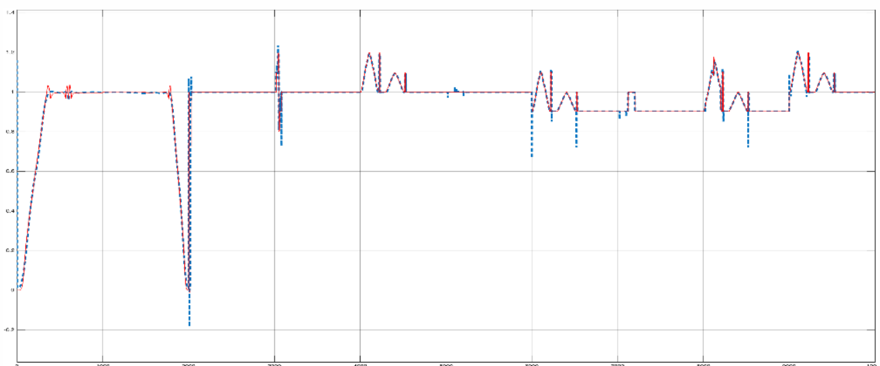


Deep Learning Solutions for Integrated Circuits

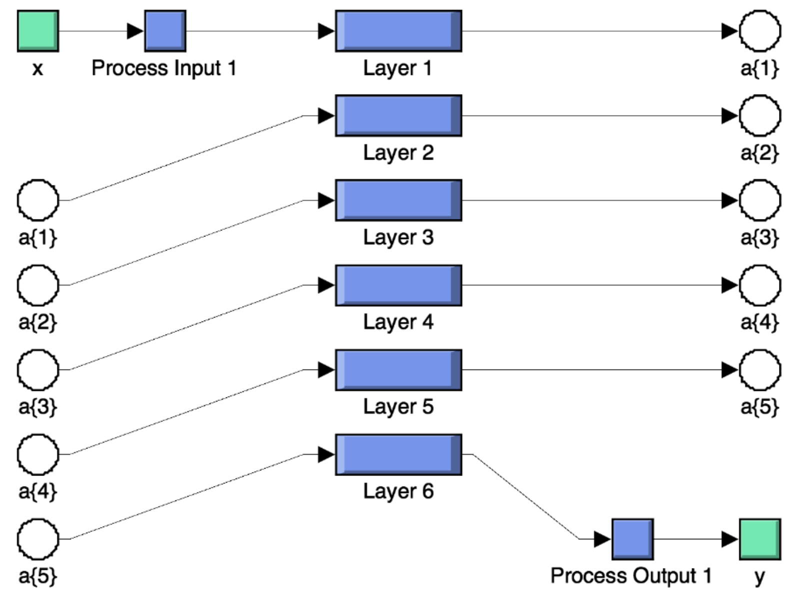
Deep Structure:
6-Layer Time-Delayed Neural Network with 3 delay components



Training dataset response



Test dataset response



- 3- input delay components
- layer 1: 50 neurons
- Layer 2-6: 10 neurons each

Unpublished