2020 PDC Project - Hines

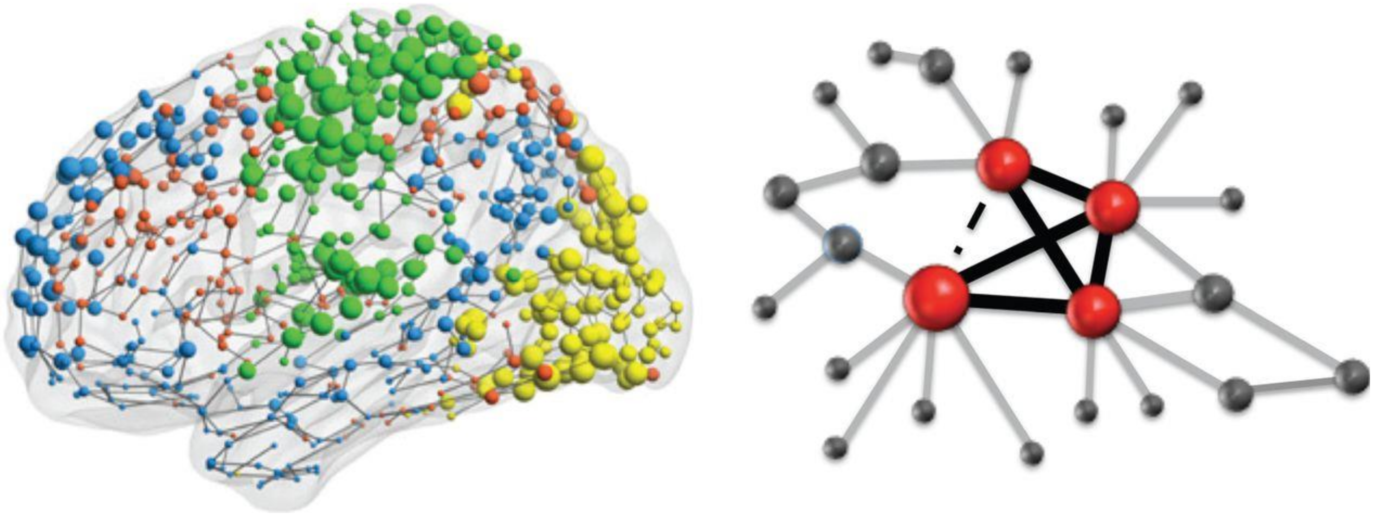
Guojie Luo

gluo@pku.edu.cn

Contents

- **Brief Introduction**
  - **Overview of brain simulation**
  - **Neuroscience Basics**
- Neuron Simulator and Related Works
  - Mathematic Basics
  - Neuron Simulator

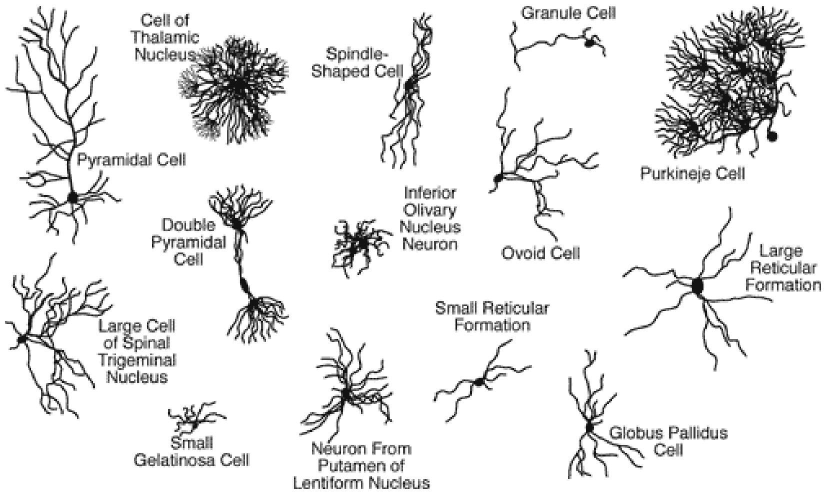
How the brain computes information?

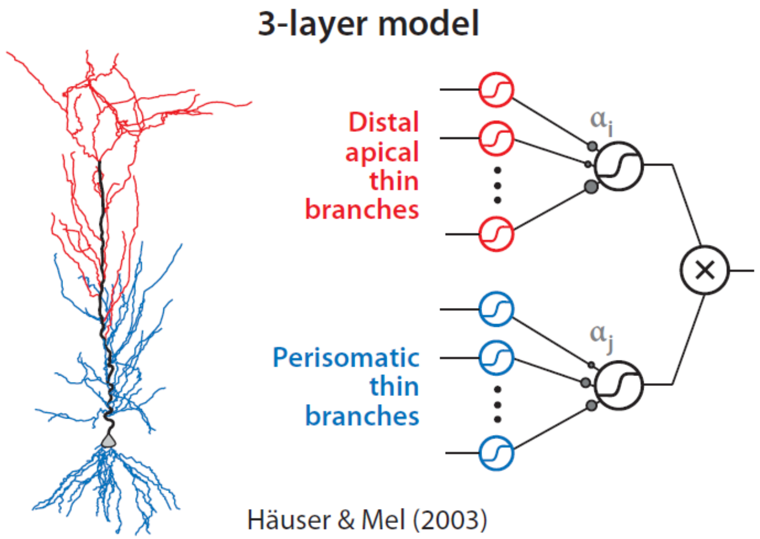


Simulating the whole brain with “point” neurons. e.g. IBM simulated a cat brain with 1.6 billions point neurons and 9 trillion synapses.

Petra E. Vértes, et.al. 2014

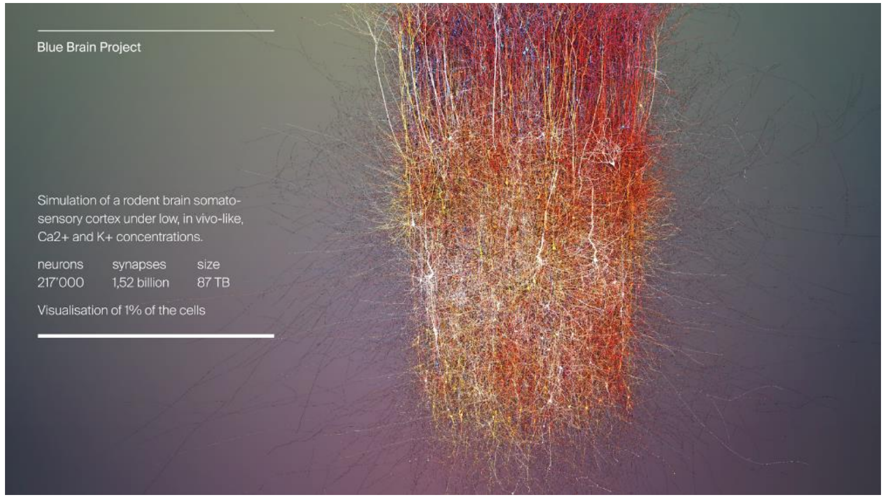
Towards better understanding the brain





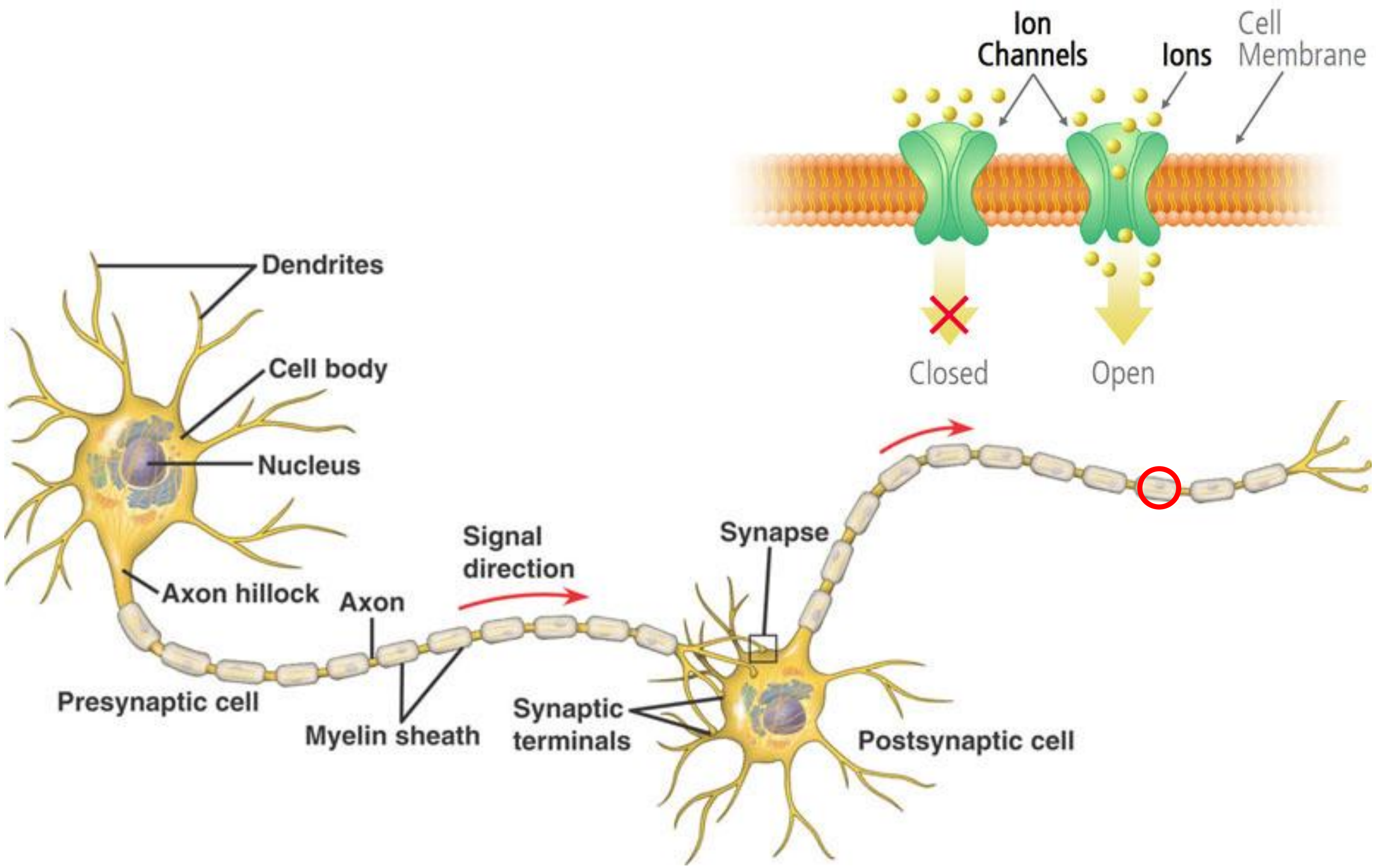
Häuser & Mel (2003)

Towards better understanding the brain



EU Human Brain Project

Neuron

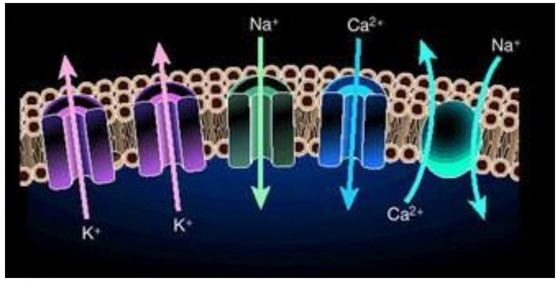


Ion Channels
Ions
Cell Membrane
Closed
Open
Dendrites
Cell body
Nucleus
Signal direction
Synapse
Axon hillock
Axon
Presynaptic cell
Myelin sheath
Synaptic terminals
Postsynaptic cell

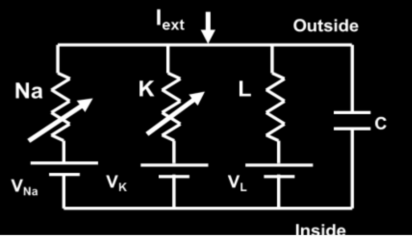
Contents

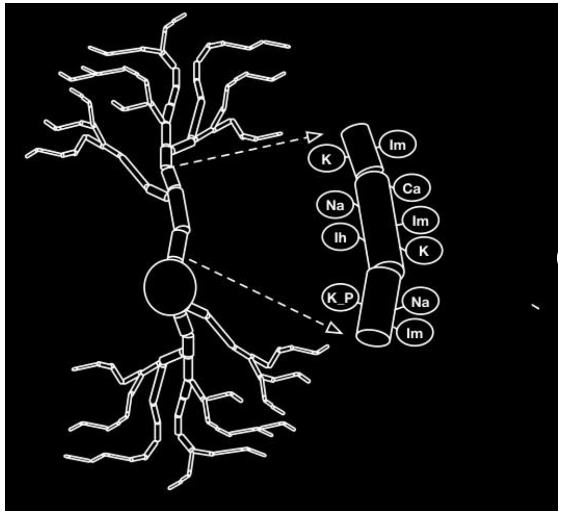
- Brief Introduction
  - Overview of brain simulation
  - Neuroscience Basics
- **Neuron Simulator and Related Works**
  - **Mathematic Basics**
  - **Neuron Simulator**

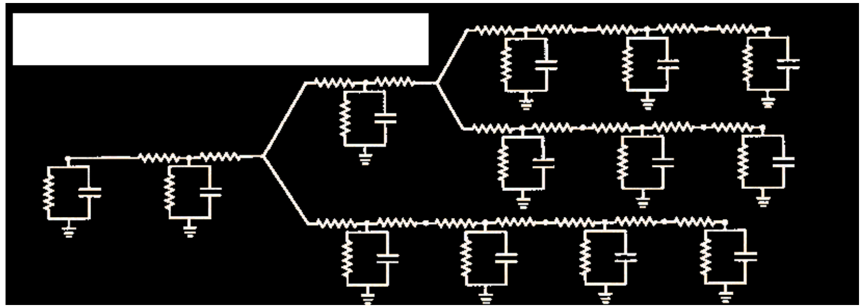
How to model a biologically detailed neuron?



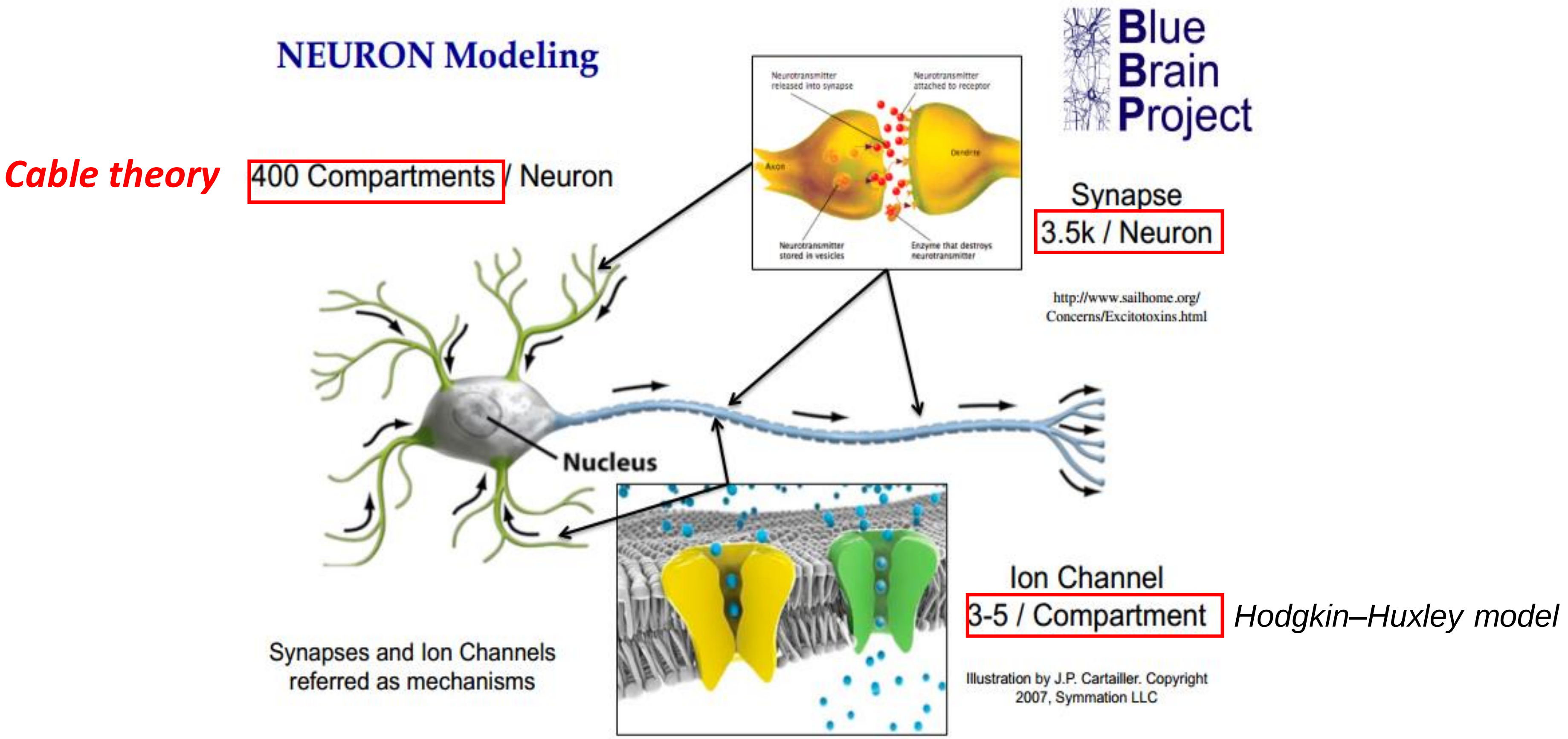
$$
\begin{aligned}
C\frac{dV}{dt} &= [I_{inj} - \bar{g}_{Na}m^3h(V - V_{Na}) - \bar{g}_K n^4(V - V_K) - g_L(V - V_L)] \\
\frac{dn}{dt} &= \alpha_n(V)(1-n) - \beta_n(V)n \\
\frac{dm}{dt} &= \alpha_m(V)(1-m) - \beta_m(V)m \\
\frac{dh}{dt} &= \alpha_h(V)(1-h) - \beta_h(V)h
\end{aligned}
$$







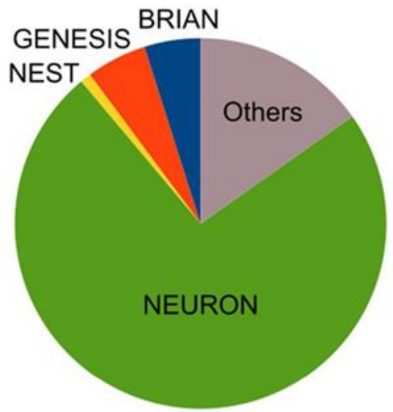
Computational Complexity of Detailed Model



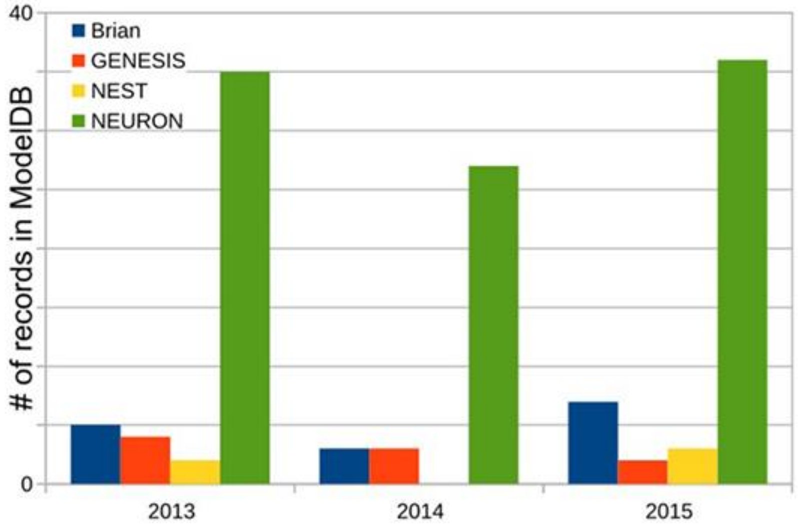
“Leveraging heterogeneous systems and deep memory hierarchies for brain tissue modeling”, Blue Brain Project

Most Popular Simulators in Neuroscience

- NEURON
  - More than 1900 papers build models on NEURON
- GENESIS
  - Classical simulator for detailed model
- CoreNEURON
  - Optimized compute engine of NEURON
  - Support single GPU simulation

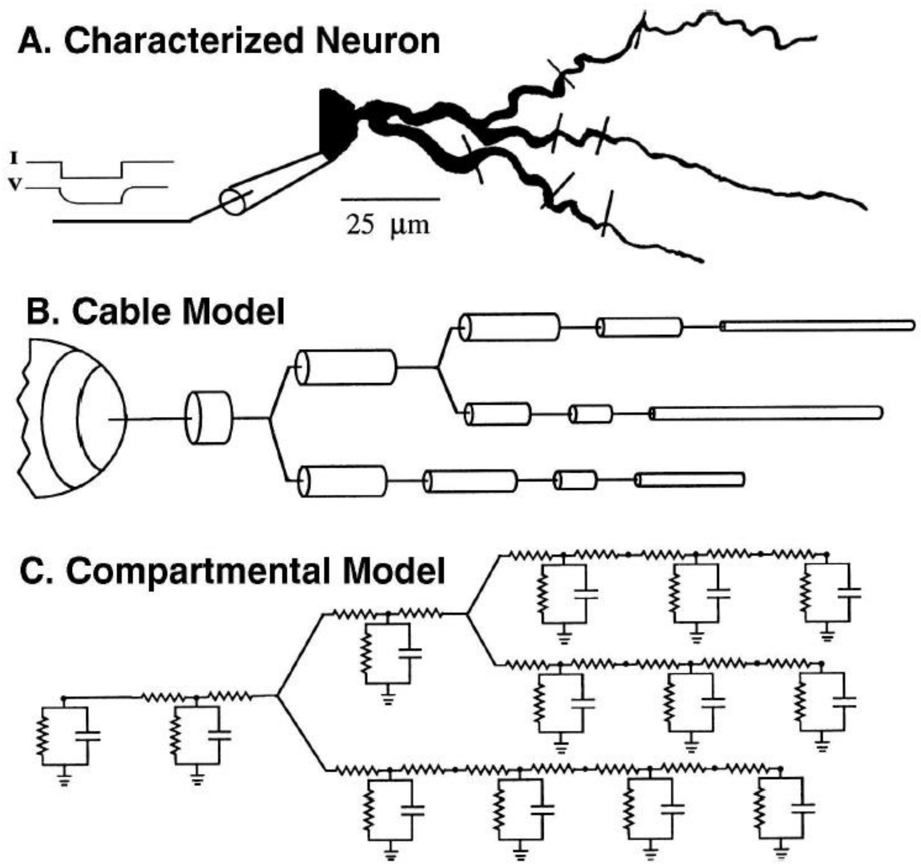


ModelDB index



Tikidji-Hamburyan R A, Narayana V, Bozkus Z, et al. Software for brain network simulations: a comparative study[J]. Frontiers in neuroinformatics, 2017, 11: 46.

Cable Equation



$$\boxed{\frac{\partial V}{\partial T}} + F(V) = \boxed{\frac{\partial^2 V}{\partial X^2}}$$

Membrane voltage changes with time

Membrane voltages at different positions influence each other

Cable Theory, Rall,1962

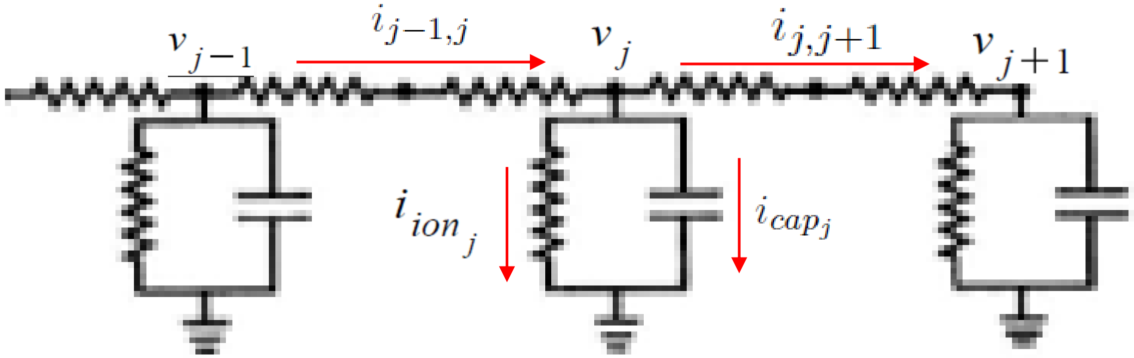
Detailed Model with HH Channels

$$\frac{\partial V}{\partial T} + F(V) = \frac{\partial^2 V}{\partial X^2}$$

$$F(V) = -\bar{g}_{Na} m^3 h (V - V_{Na}) - \bar{g}_K n^4 (V - V_K) - g_L (V - V_L)$$

$$\frac{dn}{dt} = \alpha_n(V)(1-n) - \beta_n(V) n$$

Cable Equation



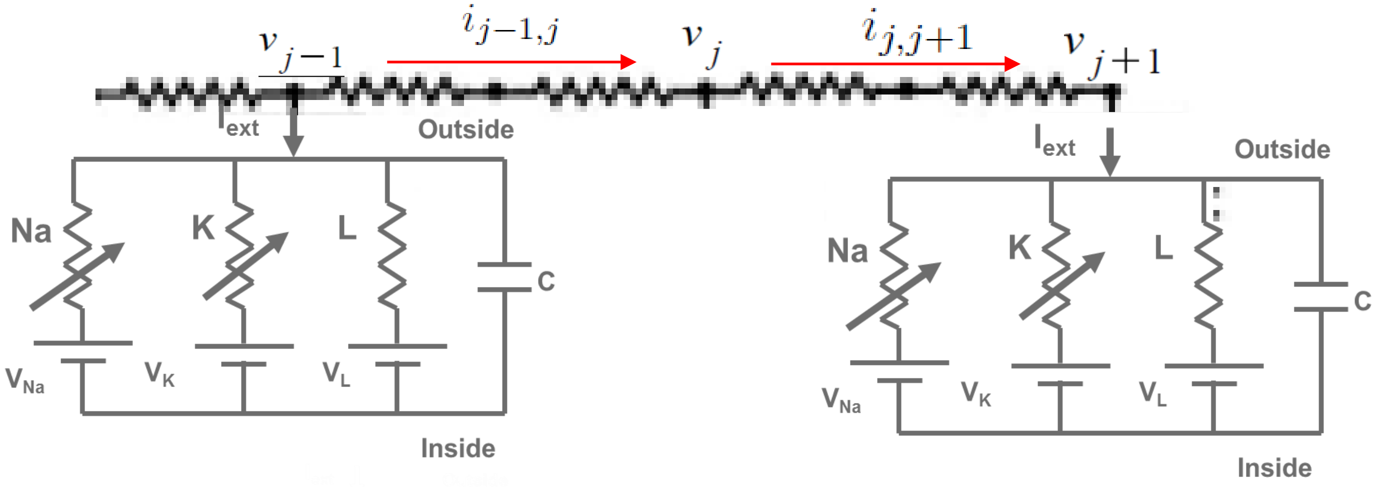
$$c_j \frac{dv_j}{dt} + i_{ion_j}(v_j, t) = \frac{v_{j-1} - v_j}{r_{j-1,k}} + \frac{v_{j+1} - v_j}{r_{j+1,k}}$$

If each compartment has length Δx and diameter d.
its capacitance is $C_m \pi d \Delta x$
axial resistance is $R_a \Delta x / \pi (d/2)^2$

$$C_m \frac{dv_j}{dt} + i_j(v_j, t) = \frac{d}{4 R_a} \frac{v_{j+1} - 2 v_j + v_{j-1}}{\Delta x^2}$$

Finite difference method for PDE

Cable Equation

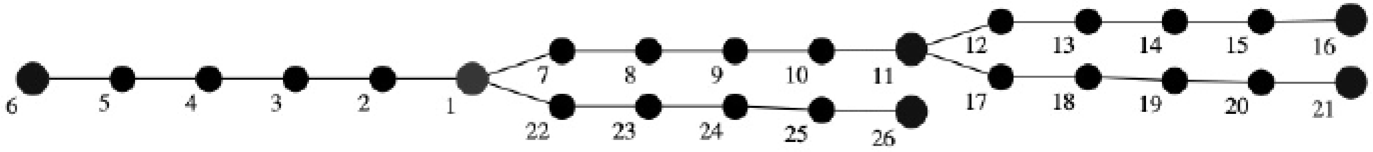


$$c_j \frac{dv_j}{dt} + i_{ion_j}(v_j, t) = \frac{v_{j-1} - v_j}{r_{j-1,\,k}} + \frac{v_{j+1} - v_j}{r_{j+1,\,k}}$$

$$i_{ion_j} = -\bar{g}_{Na} m^3 h (v_j - V_{Na}) - \bar{g}_K n^4 (v_j - V_K) - g_L (v_j - V_L)$$

$$\frac{dn}{dt} = \alpha_n(v_j)(1-n) - \beta_n(v_j) n$$

Branched Neuron Model



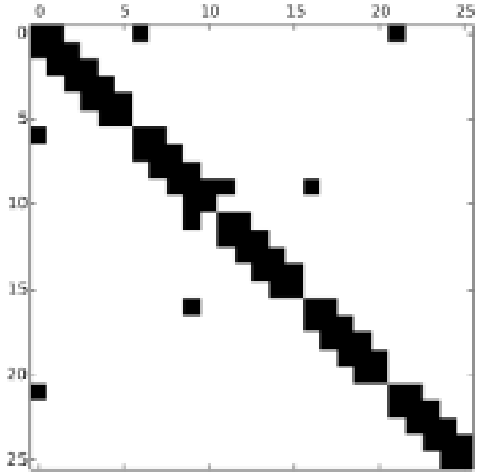
$$c_j \frac{dv_j}{dt} + i_{ion_j}(v_j, t) = \sum_k (v_k - v_j) / r_{jk}$$

Backward Euler Method

$$(\mathbf{I} - \psi\mathbf{B}' - \mathbf{G})\mathbf{V}_\mathbf{j}^{\mathbf{t+1}} = \mathbf{V}_\mathbf{j}^\mathbf{t}$$

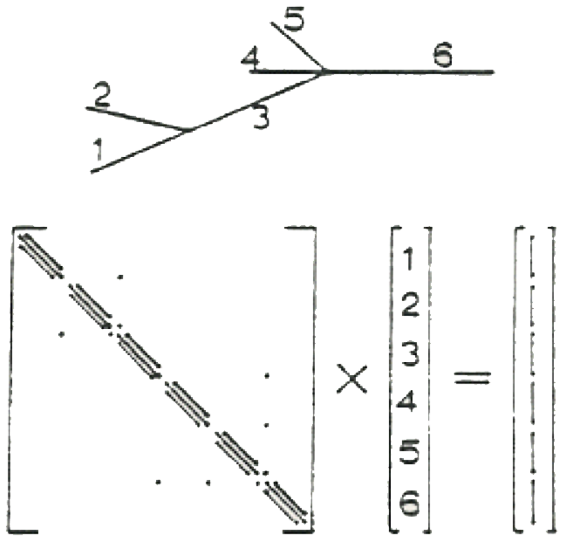
$$\mathbf{V}_\mathbf{j}^{\mathbf{t+1}} = (\mathbf{I} - \psi\mathbf{B}' - \mathbf{G})^{-1}\mathbf{V}_\mathbf{j}^\mathbf{t}$$

B' =



Operations in single step

- Deliver events
- Setup matrix
- Solve linear equations
- Update values
- Update states



Setup matrix

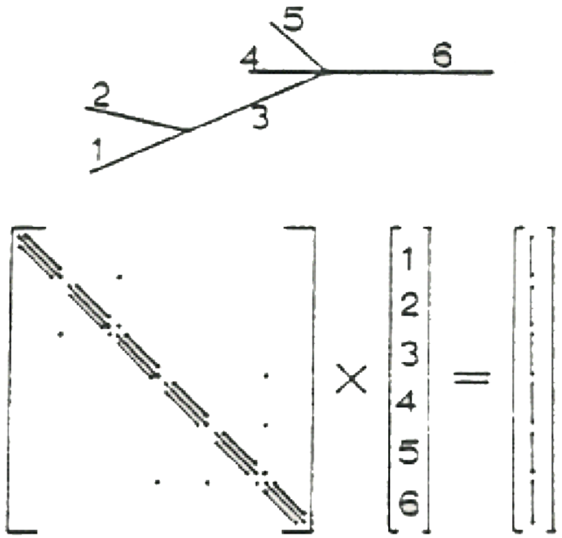
```
for (_iml = 0; _iml < _cntml_actual; ++_iml) {
#else /* LAYOUT > 1 */ /*AoSoA*/
#error AoSoA not implemented.
for (;;) { /* help clang-format properly indent */
#endif
    int _nd_idx = _ni[_iml];
    _v = _vec_v[_nd_idx];
    _PRCELLSTATE_V
  ena = _ion_ena;
  ek = _ion_ek;
 _g = _nrn_current(_threadargs_, _v + .001);
    { double _dik;
 double _dina;
  _dina = ina;
  _dik = ik;
 _rhs = _nrn_current(_threadargs_, _v);
  _ion_dinadv += (_dina - ina)/.001 ;
  _ion_dikdv += (_dik - ik)/.001 ;
    }
 _g = (_g - _rhs)/.001;
  _ion_ina += ina ;
  _ion_ik += ik ;
 _PRCELLSTATE_G
    _vec_rhs[_nd_idx] -= _rhs;
    _vec_d[_nd_idx] += _g;

}

}
```

```
static double _nrn_current(_threadargsproto_, double _v){double _current=0.;v=_v;{ {
   gna = gnabar * m * m * m * h ;
   ina = gna * ( v - ena ) ;
   gk = gkbar * n * n * n * n ;
   ik = gk * ( v - ek ) ;
   il = gl * ( v - el ) ;
   }
 _current += ina;
 _current += ik;
 _current += il;

} return _current;
}
```



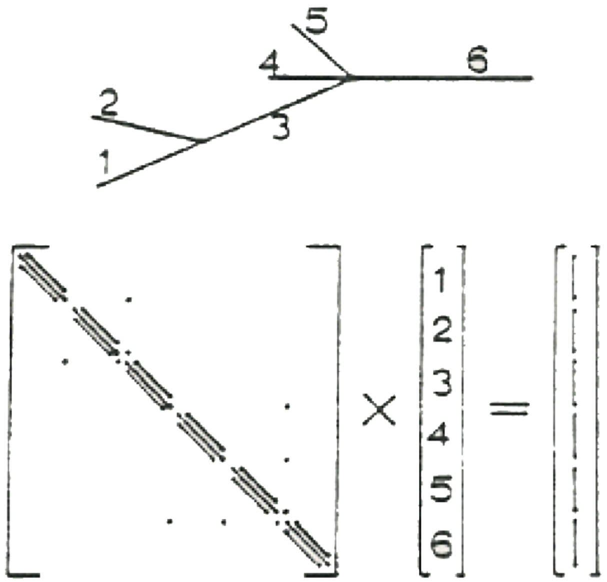
Solve equations

Algorithm 1 Hines algorithm.

```
void solveHines(double *u, double *l,   double *d,
                    double *rhs,     int *p, int cellSize)
// u → upper vector, l → lower vector
int i;
double factor;
// Backward Sweep
for i = cellSize − 1 → 0 do
    factor    = u[i] / d[i];
    d[p[i]]   -= factor × l[i];
    rhs[p[i]] -= factor × rhs[i];
end for
rhs[0] /= d[0];
// Forward Sweep
for i = 1 → cellSize − 1 do
    rhs[i] -= l[i] × rhs[p[i]];
    rhs[i] /= d[i];
end for
```

Kernel of NEURON simulator

Similar to Thomas Method



Update values

```
static void update(NrnThread* _nt) {
    int i, i1, i2;
    i1 = 0;
    i2 = _nt->end;
#if defined(_OPENACC)
    int stream_id = _nt->stream_id;
#endif
    double* vec_v = &(VEC_V(0));
    double* vec_rhs = &(VEC_RHS(0));

    /* do not need to worry about linmod or extracellular*/
    if (secondorder) {
        #pragma acc parallel loop present(vec_v[0 : i2], \
                                    vec_rhs[0 : i2]) if (_nt->compute_gpu) async(stream_id)
        for (i = i1; i < i2; ++i) {
            vec_v[i] += 2. * vec_rhs[i];
        }
    } else {
        #pragma acc parallel loop present(vec_v[0 : i2], \
                                    vec_rhs[0 : i2]) if (_nt->compute_gpu) async(stream_id)
        for (i = i1; i < i2; ++i) {
            vec_v[i] += vec_rhs[i];
        }
    }

    // update_matrix_to_gpu(_nt);

    if (_nt->tml) {
        assert(_nt->tml->index == CAP);
        nrn_cur_capacitance(_nt, _nt->tml->ml, _nt->tml->index);
    }
}
```

Update states

```c
static int states (_threadargsproto_) { {
   rates ( _threadargscomma_ v ) ;
    m = m + (1. - exp(dt*(( ( ( - 1.0 ) ) ) / mtau)))*(- ( ( ( minf ) ) / mtau ) / ( ( ( ( - 1.0) ) ) / mtau ) - m) ;
    h = h + (1. - exp(dt*(( ( ( - 1.0 ) ) ) / htau)))*(- ( ( ( hinf ) ) / htau ) / ( ( ( ( - 1.0) ) ) / htau ) - h) ;
    n = n + (1. - exp(dt*(( ( ( - 1.0 ) ) ) / ntau)))*(- ( ( ( ninf ) ) / ntau ) / ( ( ( ( - 1.0) ) ) / ntau ) - n) ;
   }
  return 0;
}
```

Solution: Backward Euler Method

$$C_m \frac{v_j^{t+1} - v_j^t}{\Delta t} = \frac{d}{4R_j} \frac{v_{j+1}^{t+1} - 2v_j^t + v_{j-1}^{t+1}}{2\Delta x^2} - g_m v_j^{t+1} \qquad C_m \frac{\partial v}{\partial t} + i(v, t) = \frac{d}{4R_a} \frac{\partial^2 v}{\partial x^2}$$

$$(\mathbf{I} - \psi \mathbf{B}' - \mathbf{G})\mathbf{V_j^{t+1}} = \mathbf{V_j^t}$$

$$\mathbf{V_j^{t+1}} = (\mathbf{I} - \psi \mathbf{B}' - \mathbf{G})^{-1}\mathbf{V_j^t}$$

$$\mathbf{B'} = \begin{bmatrix} -2 & 1 & & & \\ 1 & -2 & 1 & & \\ & \ddots & \ddots & \ddots & \\ & & 1 & -2 & 1 \\ & & & 1 & -2 \end{bmatrix}$$

Thomas Algorithm: More Efficiency

- Matrix inverse: time-consuming

$(\mathbf{I} - \psi\mathbf{B}' - \mathbf{G})\mathbf{V}_{\mathbf{j}}^{\mathbf{t+1}} = \mathbf{V}_{\mathbf{j}}^{\mathbf{t}}$: tridiagonal matrix equations

- Thomas Algorithm: solve tridiagonal matrix equations with O(n) operations

Thomas Algorithm: More Efficiency

Triangularize

$$\begin{pmatrix} b_1 & c_1 & 0 & 0 & 0 & 0 \\ a_2 & b_2 & c_2 & 0 & 0 & 0 \\ 0 & a_3 & b_3 & c_3 & 0 & 0 \\ 0 & 0 & a_4 & b_4 & c_4 & 0 \\ 0 & 0 & 0 & a_5 & b_5 & c_5 \\ 0 & 0 & 0 & 0 & a_6 & b_6 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \\ x_6 \end{pmatrix} = \begin{pmatrix} r_1 \\ r_2 \\ r_3 \\ r_4 \\ r_5 \\ r_6 \end{pmatrix}$$

row2 – a2/b1 * row1

$$\begin{pmatrix} 1 & \gamma_1 & 0 & 0 & 0 & 0 \\ 0 & 1 & \gamma_2 & 0 & 0 & 0 \\ 0 & a_3 & b_3 & c_3 & 0 & 0 \\ 0 & 0 & a_4 & b_4 & c_4 & 0 \\ 0 & 0 & 0 & a_5 & b_5 & c_5 \\ 0 & 0 & 0 & 0 & a_6 & b_6 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \\ x_6 \end{pmatrix} = \begin{pmatrix} \rho_1 \\ \rho_2 \\ r_3 \\ r_4 \\ r_5 \\ r_6 \end{pmatrix}$$

rowi – ai/b(i-1) * row(i-1)

$$\begin{pmatrix} 1 & \gamma_1 & 0 & 0 & 0 & 0 \\ 0 & 1 & \gamma_2 & 0 & 0 & 0 \\ 0 & 0 & 1 & \gamma_3 & 0 & 0 \\ 0 & 0 & 0 & 1 & \gamma_4 & 0 \\ 0 & 0 & 0 & 0 & 1 & \gamma_5 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \\ x_6 \end{pmatrix} = \begin{pmatrix} \rho_1 \\ \rho_2 \\ \rho_3 \\ \rho_4 \\ \rho_5 \\ \rho_6 \end{pmatrix}$$

Back-Substitute

$$x_6 = \rho_6.$$

$$x_5 = \rho_5 - \gamma_5 x_6.$$

$$x_4 = \rho_4 - \gamma_4 x_5.$$

$$x_3 = \rho_3 - \gamma_3 x_4.$$

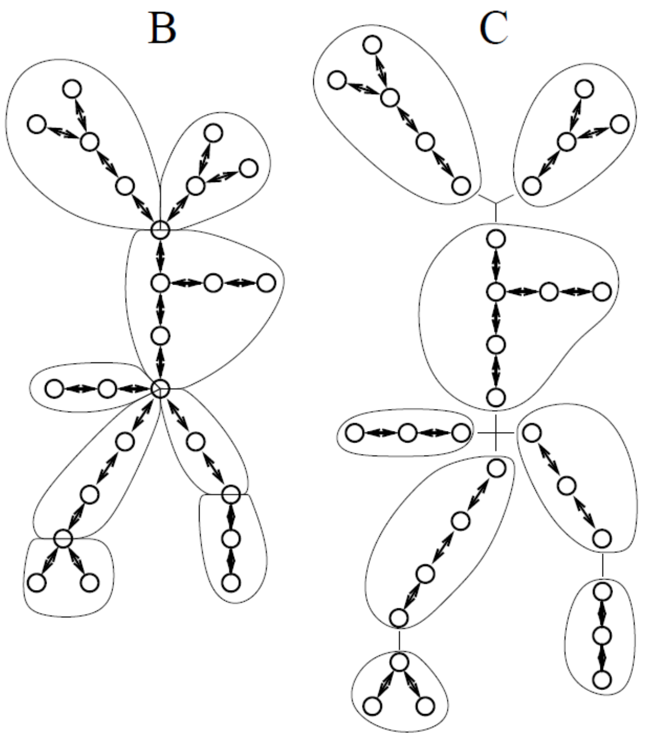
$$x_2 = \rho_2 - \gamma_2 x_3.$$

$$x_1 = \rho_1 - \gamma_1 x_2.$$

Current Progress to Speedup Simulation

- Network level:
  - Use multiple processes / threads to parallelize the computation of different neurons

- **Cell level:**
  - **Parallelize single cell computation**
  - **Main idea: parallelize computation on different branches**



Multisplit -- Hines M L et al. 2005

Related Works of Cell Parallelism

- “A parallelizing algorithm for computing solutions to arbitrarily branched cable neuron models” Michael Mascagni, Journal of Neuroscience, 1990
- solve two tridiagonal systems per branch.
  - One with the original right hand side but with the "branch point" voltage assumed zero,
  - The other with the a zero right hand side and with the "branch point" voltage assumed one.
  - Combine the results

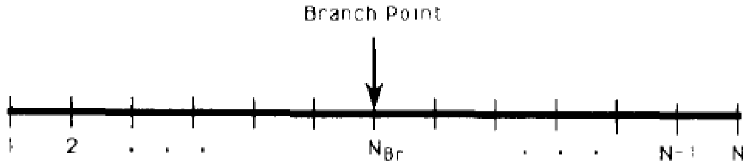


Fig. 2. The simplest branching structure with the ‘branch point’ identified.

Related Works of Cell Parallelism

- Multi-split: used in NEURON
  - Divide a tree into subtrees
  - Triangularize each subtree as much as possible
  - Transform tridiagonal submatrix into a submatrix in which only two end compartments affects
  - Each subtree sends equations that still contain interaction terms
  - Back substitute