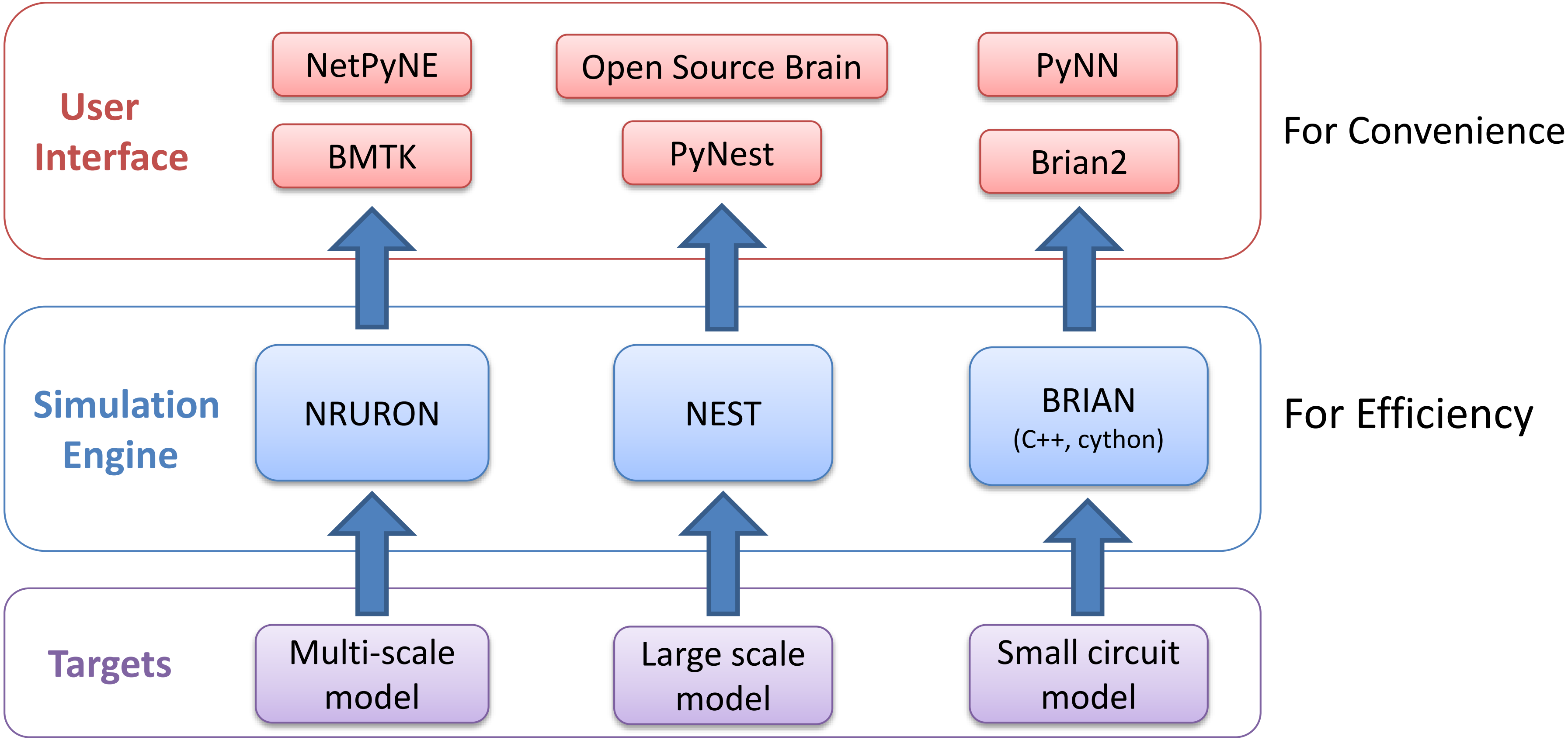
BrainPy (灵机): A *flexible* and *extensible* framework for brain modeling

https://github.com/PKU-NIP-Lab/BrainPy

王超名 chao.brain@qq.com

The programming paradigm of current neural simulators



Model definition, building and simulation are separated from each other.

Example: NEST

底层定义和构建模型，Python调用模型

```python
import nest

nest.ResetKernel()
nest.SetKernelStatus(
    {"resolution": 0.05})
neuron_param = {"tau_m": Tau,
                "t_ref": TauR,
                "tau_syn_ex": Tau_psc,
                "tau_syn_in": Tau_psc,
                "C_m": C,
                "V_reset": E_L,
                "E_L": E_L,
                "V_m": E_L,
                "V_th": Theta}
neurons = nest.Create("iaf_psc_exp", 2)
neurons.set(neuron_param)
```

```cpp
#include "iaf_psc_exp.h"

// C++ includes:
#include <limits>

// Includes from libnestutil:
#include "dict_util.h"
#include "numerics.h"
#include "propagator_stability.h"

// Includes from nestkernel:
#include "event_delivery_manager_impl.h"
#include "exceptions.h"
#include "kernel_manager.h"
#include "universal_data_logger_impl.h"
```

Flexibility Problem

	NEURON	NEST
Neuron model without dynamics	M	M
Neuron model with simplified and discontinuous dynamics *Examples: Leaky Integrate-and-Fire (LIF), Izhikevich or Quadratic LIF; Exponent Leaky Integrate-and-Fire (eLIF)*	M	M
Neuron model with simplified and continues dynamics *Examples: FitzHugh–Nagumo, Morris–Lecar*	M	M
Single compartment, conductance-based model—temporal integration (point neuron) *Examples: Single-Compartment Hodgkin–Huxley model*	YG	M
Can conductance-based descriptions of ion channels be added to the neuron model? *Example: h-channel*	YG/M	m
Neuron model with simplify morphology (2-compartment model) *Example: Pinsky–Rinzel model*	YG	M
New model of chemical synapse	M	m
New model of electrical synapse	M	m
New model of learning rule	m	M

Tikidji-Hamburyan, Ruben A., et al. "Software for brain network simulations: a comparative study." Frontiers in Neuroinformatics 11 (2017): 46.

Descriptive Language： Python描述(定义)模型，底层构建和运行模型

Example: BRIAN2

```
G = NeuronGroup(10, '''dv/dt = I_leak / Cm : volt
                      I_leak = g_L*(E_L - v) : amp''')
```

Code Generation

```
for(int _idx=0; _idx<_num_idx; _idx++)
{
      double v = _array_neurongroup_v[_idx];
      double w = _array_neurongroup_w[_idx];
      const double _w = -(dt) * w / tau_w + w;
      const double _v = dt * (-(v) + w) / tau_v + v;
      w = _w;
      v = _v;
      _array_neurongroup_v[_idx] = v;
      _array_neurongroup_w[_idx] = w;
}
```

(frontiers, 2014; eLife, 2019)

Example: NetPyNE

i)

```
popParams['EXC_L2'] = {
'cellType':   'PYR',
'cellModel':  'simple',
'yRange':     [100, 400],
'numCells':   50}
```

ii)

```
popParams['EXC_L5'] = {
'cellType':   'PYR',
'cellModel':  'complex',
'yRange':     [700, 1000],
'density':    80e3}
```

iii)

```
cellParams['PYR_simple'] = {
'conds':  {'cellType': 'PYR',
           'cellModel': 'simple'},
'secs':   {'soma':
            'geom': {'diam': 18, 'L': 18},
            'mechs': {'hh':
                       {'gnabar': 0.12,
                        'gkbar': 0.036,
                        'gl':    0.003,
                        'el':    -70}}}}
```

iv)

```
importCellParams(
label =     'PYR_complex',
conds =     {'cellType':   'PYR',
             'cellModel':  'complex'},
fileName = 'L5_pyr_full.hoc',
cellName = 'PYR_L5')
```

(eLife, 2019)

Example: BMTK

```python
from bmtk.builder import NetworkBuilder
from bmtk.builder.auxi.node_params import positions_columinar

net = NetworkBuilder("ei")
net.add_nodes(N=8000, pop='exc', model_name='Scnn1a',
              morphology='Scnn1a_473845048_m',
              model_type = 'biophysical',
              model_template='nml:Cell_472363762.cell.nml',
              positions=positions_columinar(8000, radius=1000, height=500))
net.add_nodes(N=2500, pop='inh', model_name='PV',
              morphology='Pvalb_470522102_m',
              model_type = 'biophysical',
              model_template='nml:Cell_472912177.cell.nml',
              positions=positions_columinar(2500, radius=1000, height=500))

net.add_edges(source={'pop': 'e'}, target={'pop': 'i'},
          connection_rule=gaussian_distance,
          dynamics_params='AMPA_ExcToInh.json',
          model_template='Exp2Syn',
          syn_weight=0.1,
          delay=1.5,
          target_sections=['somatic', 'basal'])
net.add_edges(source={'pop': 'i'}, target={'pop': 'e'},
          connection_rule=random_connections,
          dynamics_params='GABA_InhToExc.json',
          model_template='Exp2Syn',
          syn_weight=0.5,
          delay=1.5,
          target_sections=['apical'],
          distance_range=[50.0, 100.0])

net.build()
net.save(output_dir='ei_network')
```

(PLOS CB, 2020)

Transparence and Extensibility Problems

1. String description is pseudo programming, greatly reducing the program expressive power

```
dm/dt = alpha_m*(1-m)-beta_m*m : 1
dn/dt = alpha_n*(1-n)-beta_n*n : 1
dh/dt = alpha_h*(1-h)-beta_h*h : 1
```

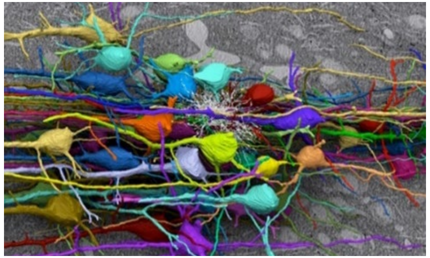
```python
m = np.clip(int_m(ST['m'], _t_, ST['V']), 0., 1.)
h = np.clip(int_h(ST['h'], _t_, ST['V']), 0., 1.)
n = np.clip(int_n(ST['n'], _t_, ST['V']), 0., 1.)
```

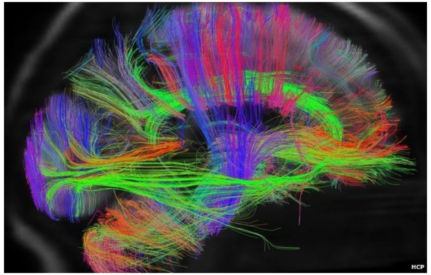
2. 代码对用户隐藏，不支持debug，不知道是否生成用户想到的逻辑。一旦发现错误用户无法纠正代码。这往往导致用户定义新模型时捉襟见肘。

3. 对不满足假设与规定的模型不支持

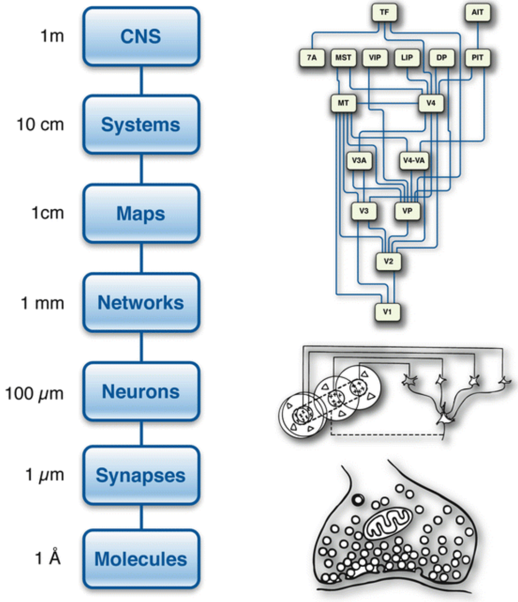
Challenges of Programming in Computational Neuroscience

多尺度建模





大尺度建模



1. Efficiency Problem

2. Flexibility & Transparence Problem

新模型

- Pre-defined models is not sufficient
- New models at every scale

3. Extensibility Problem

新方法

- New differential equations
- New numerical integrators
- Numerical continuation

Easy to learn and use

Flexible

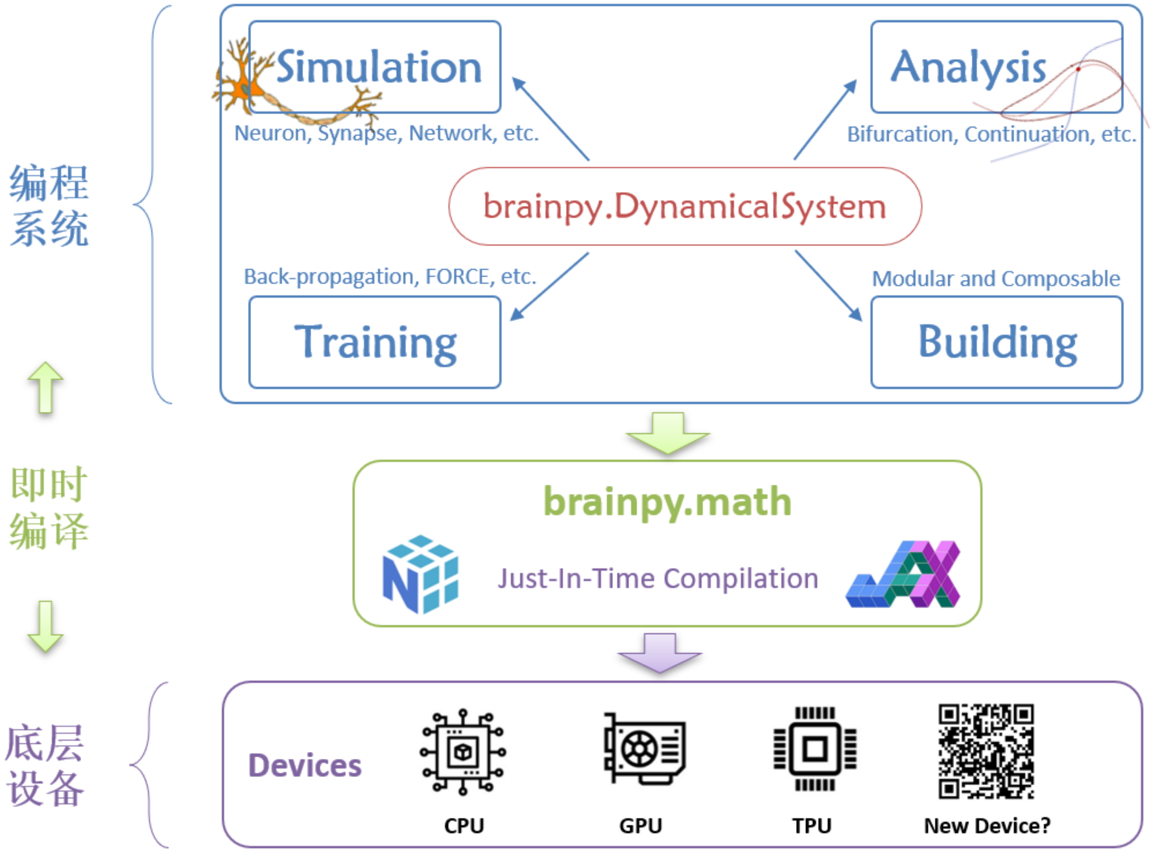
Transparent

Extensible

Efficient

- ✓ 到目前没有任何一个计算神经科学内的框架能像PyTorch、TensorFlow一样允许用户自主灵活编程
- ✓ 是时候改变计算神经科学的编程范式了！

A Just-In-Time compilation approach for brain modeling



Model definition, building and simulation are all done in Python.

- Easy to learn and use
- Efficient
- Flexible
- Transparent
- Extensible

HH model as an example for flexible customization

```python
class HH(bp.NeuGroup):
  def __init__(self, size, ENa=50., gNa=120., EK=-77., gK=36., EL=-54.387,
               gL=0.03, V_th=20., C=1.0, method='exponential_euler', **kwargs):
    # 初始化父类
    super(HH, self).__init__(size=size, **kwargs)

    # 定义神经元参数
    self.ENa = ENa
    self.EK = EK
    self.EL = EL
    self.gNa = gNa
    self.gK = gK
    self.gL = gL
    self.C = C
    self.V_th = V_th

    # 定义神经元变量
    self.V = bm.Variable(-65. * bm.ones(self.num))  # 膜电位
    self.m = bm.Variable(0.5 * bm.ones(self.num))  # 离子通道m
    self.h = bm.Variable(0.6 * bm.ones(self.num))  # 离子通道h
    self.n = bm.Variable(0.32 * bm.ones(self.num))  # 离子通道n
    self.input = bm.Variable(bm.zeros(self.num))  # 神经元接收到的输入电流
    self.spike = bm.Variable(bm.zeros(self.num, dtype=bool))  # 神经元的发放状态
    self.t_last_spike = bm.Variable(bm.ones(self.num) * -1e7)  # 神经元上次发放的时刻
```

Initialize Parameters

Initialize Variables

```python
    @bp.odeint(method='exponential_euler')
    def integral(self, V, m, h, n, t, Iext):
        alpha = 0.1 * (V + 40) / (1 - bm.exp(-(V + 40) / 10))
        beta = 4.0 * bm.exp(-(V + 65) / 18)
        dmdt = alpha * (1 - m) - beta * m

        alpha = 0.07 * bm.exp(-(V + 65) / 20.)
        beta = 1 / (1 + bm.exp(-(V + 35) / 10))
        dhdt = alpha * (1 - h) - beta * h

        alpha = 0.01 * (V + 55) / (1 - bm.exp(-(V + 55) / 10))
        beta = 0.125 * bm.exp(-(V + 65) / 80)
        dndt = alpha * (1 - n) - beta * n

        I_Na = (self.gNa * m ** 3.0 * h) * (V - self.ENa)
        I_K = (self.gK * n ** 4.0) * (V - self.EK)
        I_leak = self.gL * (V - self.EL)
        dVdt = (- I_Na - I_K - I_leak + Iext) / self.C
        return dVdt, dmdt, dhdt, dndt

    def update(self, _t, _dt):
        V, m, h, n = self.integral(self.V, self.m, self.h, self.n, _t, self.input, dt=_dt)  # 更新变量值
        self.spike[:] = bm.logical_and(self.V < self.V_th, V >= self.V_th)  # 判断神经元是否产生膜电位
        self.t_last_spike[:] = bm.where(self.spike, _t, self.t_last_spike)  # 更新神经元发放的时间
        self.V[:] = V
        self.m[:] = m
        self.h[:] = h
        self.n[:] = n
        self.input[:] = 0.   # 重置神经元接收到的输入
```

$$\frac{dm}{dt} = \alpha_m(1-m) - \beta_m$$

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

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

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

$$x(t+dt) = f(x(t), t, dt)$$

BrainModels: https://github.com/PKU-NIP-Lab/BrainModels

Channels	INa	IK
Ca	ICaT	ICaN

Neuron models	Hodgkin-Huxley Model	Morris–Lecar model
LIF model	Quadratic LIF model	Exponential LIF model
Resonate-fire model	Izhikevich model	Generalized LIF model
Wilson Cowan model	Hindmarsh Rose model	And others ···

<table>
<tr><th colspan="2">Synapse models</th><th colspan="2">Exponential synapse</th><th>Voltage jump synapse</th></tr>
<tr><td colspan="2">Dual Exponential synapse</td><td colspan="2">Alpha synapse</td><td>Short-term plasticity</td></tr>
<tr><td>GABA$_A$</td><td>GABA$_B$</td><td>NMDA</td><td>AMPA</td><td>And others ···</td></tr>
</table>

Network models	CANN network	E/I balanced network
Liquid-state machine	Echo state network	And others ···

Learning rules	STDP	Oja	BCM	And others ···