Introduction to NESTML

Tom Tetzlaff

Institute of Neuroscience and Medicine (INM-6), Jülich Research Centre and JARA

t.tetzlaff@fz-juelich.de

http://www.csn.fz-juelich.de

EITN fall school, Paris, 22.09.2023

https://github.com/tomtetzlaff/2023_eitnfallschool

Outline

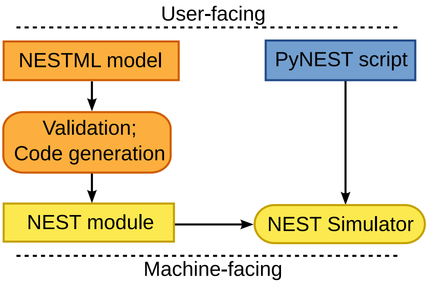
Overview

“Hello world (neuron)!”

“Hello world (synapse)!”

Overview

- domain-specific language for definition of **custom neuron and synapse models**
- specifically tailored for NEST (SpiNNaker and NEST GPU support in progress)
- NESTML toolchain includes
  - syntax validation
  - system analysis and automatized selection of appropriate solving method (using ODE-toolbox)
  - code generation (C++ for NEST)
- see NESTML language concepts



- code: `https://github.com/nest/nestml`
- docs: `https://nestml.readthedocs.io`

“Hello world (neuron)!”: NESTML model definition I

```

neuron iaf_psc_exp:

  state:
    r integer = 0      # refratory state
    V_m mV = 0 mV      # membrane potential

  equations:
    kernel I_kernel_exc = exp(-t / tau_syn_exc)
    kernel I_kernel_inh = exp(-t / tau_syn_inh)
    recordable inline I_syn pA = convolve(I_kernel_exc, ExcInput) * pA - convolve(I_kernel_inh, In

    V_m' = - (V_m - E_L) / tau_m + (I_syn + I_e + IStim) / C_m

  parameters:
    C_m pF = 250 pF            # membrane capacitance
    tau_m ms = 10 ms           # membrane time constant
    tau_syn_exc ms = 5 ms      # time constant of excitatory synapses
    tau_syn_inh ms = 5 ms      # time constant of inhibitory synapses
    t_ref ms = 2 ms            # refractory period
    E_L  mV = 0.0 mV           # resting potential
    V_reset mV = 0.0 mV        # reset potential
    V_th   mV = 15.0 mV        # spike threshold
    I_e pA = 0 pA              # constant external input current

  internals:
    RefractoryCounts integer = steps(t_ref) # refractory time in steps

  input:

```

“Hello world (neuron)!”: NESTML model definition II

```
    ExcInput <- excitatory spike
    InhInput <- inhibitory spike
    IStim     pA <- continuous

  output:
    spike

  update:

    integrate_odes()

    if r == 0:          # neuron is not refractory
      if V_m >= V_th:   # threshold crossing
        emit_spike()
        r = RefractoryCounts
        V_m = V_reset

    else:               # neuron is refractory
      V_m = V_reset
      r -= 1
```

(see `iaf_psc_exp.nestml`)

“Hello world (neuron)!”: PyNEST code

```python
import nest                                                         # import NEST module
import matplotlib.pyplot as plt                                     # for plotting
from pynestml.frontend.pynestml_frontend import generate_nest_target # NESTML

# compile nestml model
generate_nest_target(input_path="../nestml/iaf_psc_exp.nestml",
                     target_path="./nestml_target",
                     suffix="_nestml",
                     logging_level='ERROR')

# install resulting NESTML module to make models available in NEST
nest.Install('nestmlmodule')

nest.ResetKernel() # reset simulation kernel

neuron=nest.Create('iaf_psc_exp_nestml') # create LIF neuron with exponential synaptic currents

...

plt.ylabel('membrane potential (mV)')
plt.savefig('./figures/hello_world_nestml.pdf')
```

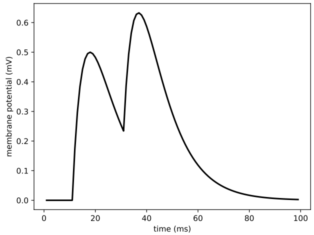
(see `hello_world_nestml.py`)

“Hello world (neuron)!”: PyNEST code

```python
import nest                                                      # import NEST module
import matplotlib.pyplot as plt                                  # for plotting
from pynestml.frontend.pynestml_frontend import generate_nest_target # NESTML

# compile nestml model
generate_nest_target(input_path="../nestml/iaf_psc_exp.nestml",
                     target_path="./nestml_target",
                     suffix="_nestml",
                     logging_level='ERROR')

# install resulting NESTML module to make models available in NEST
nest.Install('nestmlmodule')

nest.ResetKernel() # reset simulation kernel

neuron=nest.Create('iaf_psc_exp_nestml') # create LIF neuron with exponential synaptic currents

...

plt.ylabel('membrane potential (mV)')
plt.savefig('./figures/hello_world_nestml.pdf')
```

(see `hello_world_nestml.py`)

`hello_world_nestml.pdf`:



“Hello world (synapse)!”

Thanks

References I