Introduction to NESTML

Tom Tetzlaff

t.tetzlaff@fz-juelich.de

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

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

EITN fall school, Paris, 22.09.2023

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

Outline

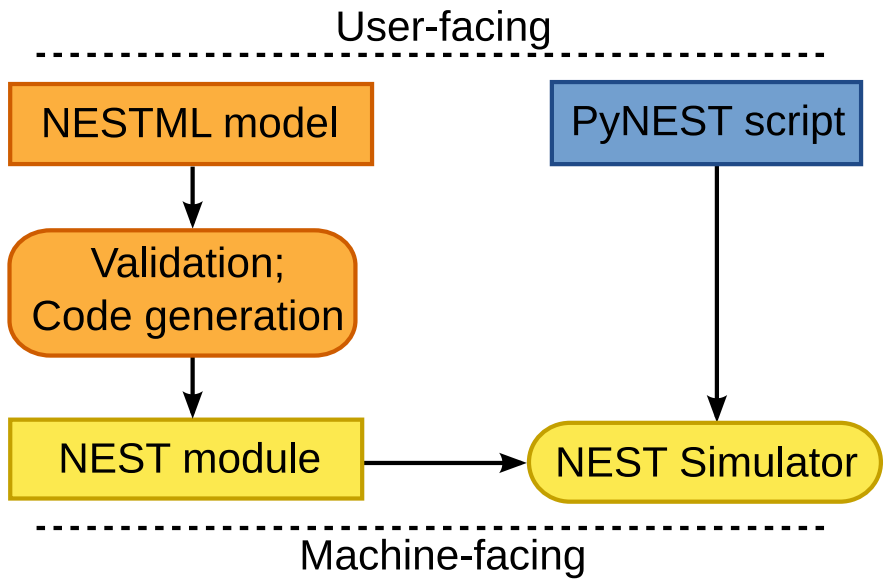
Overview

“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`

nest::ml

“Hello world (neuron)!”: PyNEST code

```
1  import nest                                                          # import NEST module
2  import matplotlib.pyplot as plt                                      # for plotting
3  from pynestml.frontend.pynestml_frontend import generate_nest_target # NESTML
4
5  # compile nestml model
6  generate_nest_target(input_path="../nestml/iaf_psc_exp_nestml.nestml",
7                       target_path="./nestml_target",
8                       logging_level='ERROR')
9
10 # install resulting NESTML module to make models available in NEST
11 nest.Install('nestmlmodule')
12
13 nest.ResetKernel() # reset simulation kernel
14
15 neuron=nest.Create('iaf_psc_exp_nestml') # create LIF neuron with exponential synaptic currents


   ...


40 plt.savefig('./figures/hello_world_nestml.pdf')
```

(see `hello_world_nestml.py`)

“Hello world (neuron)!”: PyNEST code

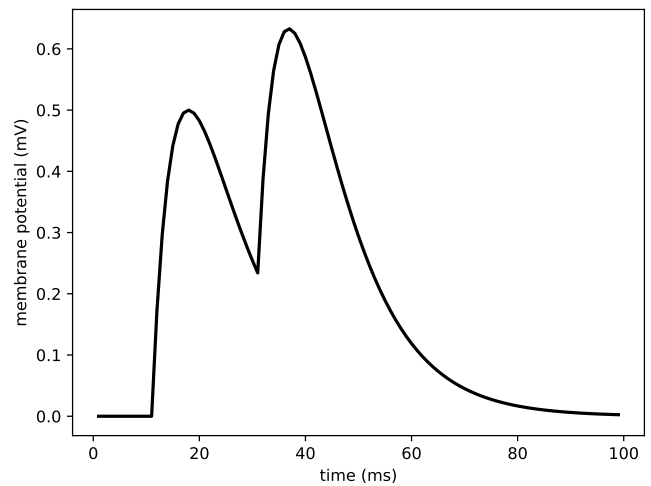
```
1  import nest                                                          # import NEST module
2  import matplotlib.pyplot as plt                                      # for plotting
3  from pynestml.frontend.pynestml_frontend import generate_nest_target # NESTML
4
5  # compile nestml model
6  generate_nest_target(input_path="../nestml/iaf_psc_exp_nestml.nestml",
7                       target_path="./nestml_target",
8                       logging_level='ERROR')
9
10 # install resulting NESTML module to make models available in NEST
11 nest.Install('nestmlmodule')
12
13 nest.ResetKernel() # reset simulation kernel
14
15 neuron=nest.Create('iaf_psc_exp_nestml') # create LIF neuron with exponential synaptic currents


   ...


40 plt.savefig('./figures/hello_world_nestml.pdf')
```

(see `hello_world_nestml.py`)

`hello_world_nestml.pdf`:



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

```
1
2  neuron iaf_psc_exp_nestml:
3
4    state:
5      r integer = 0       # refratory state
6      V_m mV = 0 mV       # membrane potential
7
8    equations:
9      kernel I_kernel_exc = exp(-t / tau_syn_exc)
10     kernel I_kernel_inh = exp(-t / tau_syn_inh)
11     recordable inline I_syn pA = convolve(I_kernel_exc, ExcInput) * pA - convolve(I_kernel_inh, In
12
13     V_m' = -(V_m - E_L) / tau_m + (I_syn + I_e + IStim) / C_m
14
15   parameters:
16     C_m pF = 250 pF             # membrane capacitance
17     tau_m ms = 10 ms            # membrane time constant
18     tau_syn_exc ms = 5 ms       # time constant of excitatory synapses
19     tau_syn_inh ms = 5 ms       # time constant of inhibitory synapses
20     t_ref ms = 2 ms             # refractory period
21     E_L  mV = 0.0 mV            # resting potential
22     V_reset mV = 0.0 mV         # reset potential
23     V_th   mV = 15.0 mV         # spike threshold
24     I_e pA = 0 pA               # constant external input current
25
26   internals:
27     RefractoryCounts integer = steps(t_ref) # refractory time in steps
28
29   input:
30
```

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

```
31      ExcInput <- excitatory spike
32      InhInput <- inhibitory spike
33      IStim      pA <- continuous
34
35    output:
36      spike
37
38    update:
39
40      integrate_odes()
41
42      if r == 0:          # neuron is not refractory
43        if V_m >= V_th:   # threshold crossing
44          emit_spike()
45          r = RefractoryCounts
46          V_m = V_reset
47
48      else:               # neuron is refractory
49        V_m = V_reset
50        r -= 1
```

(see `iaf_psc_exp_nestml.nestml`)

“Hello world (synapse)!”

Thanks

References I