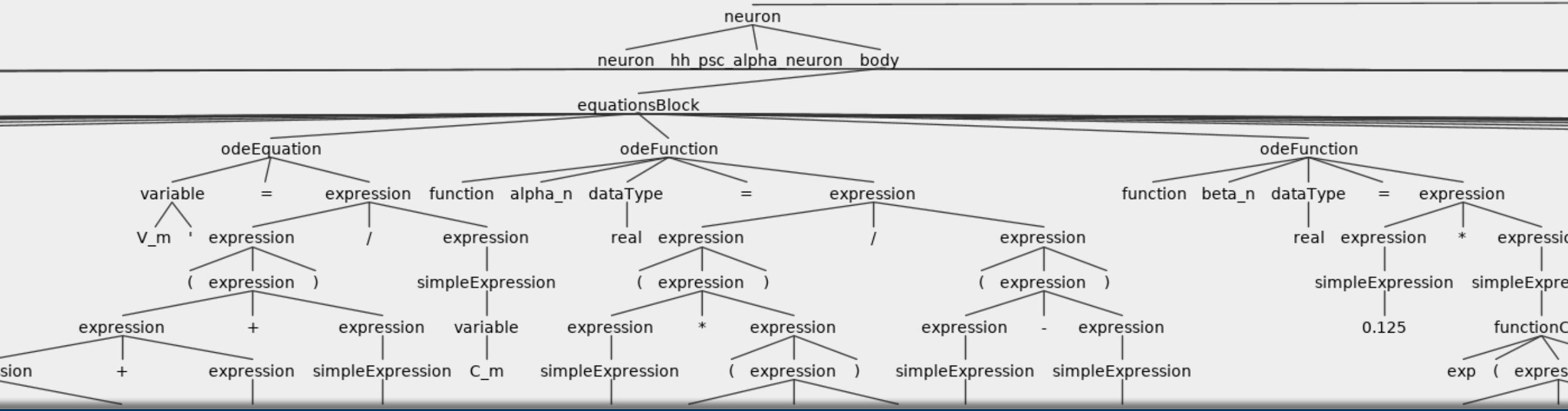


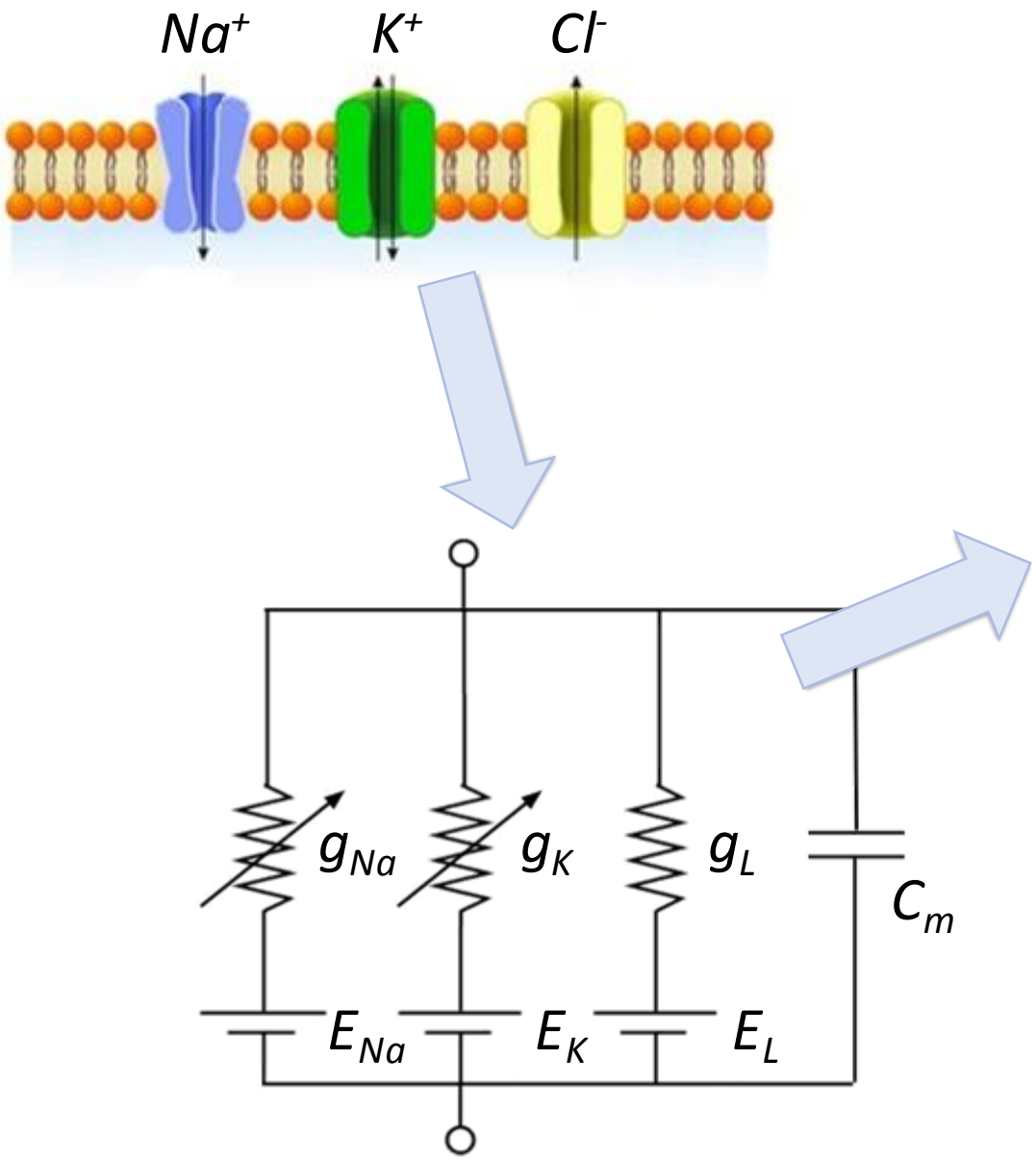
NESTML: CODE GENERATION FOR NEURON AND SYNAPSE MODELS

18 July 2020 I CNS2020 | Charl Linssen <c.linssen@fz-juelich.de>

Co-funded by the European Union

Human Brain Project

JÜLICH Forschungszentrum
Member of the Helmholtz Association



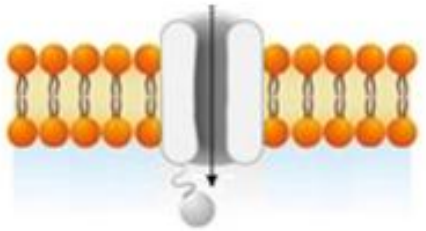

```
neuron hodgkin_huxley:
  state:
    V_m mV = -65 mV
    Act_m, Act_n, Inact_h ...
  end

  equations:
    shape syn_psc_kernel = exp(-t / tau_syn)
    function I_Na pA = g_Na * Act_m**3 * Inact_h * (V_m - E_Na)
    function I_K pA = ...
    function I_L pA = g_L * (V_m - E_L)
    V_m' = -(I_Na + I_K + I_L) / C_m
           + convolve(syn_psc_kernel, spikes)
    Act_n' = (alpha_n * (1 - Act_n) - beta_n * Act_n) / ms
    Act_m' = ...
    Inact_h' = ...
  end

  parameters:
    C_m pF = 250 pF
    V_threshold mV = 40 mV
    ...
  end

  update:
    integrate_odes()
    if V_m >= V_threshold:
      emit_spike()
    end
  end
end
```

nest::ml

JÜLICH Forschungszentrum

Image credit: Designua / Shutterstock

NMDA (Ca^{2+})



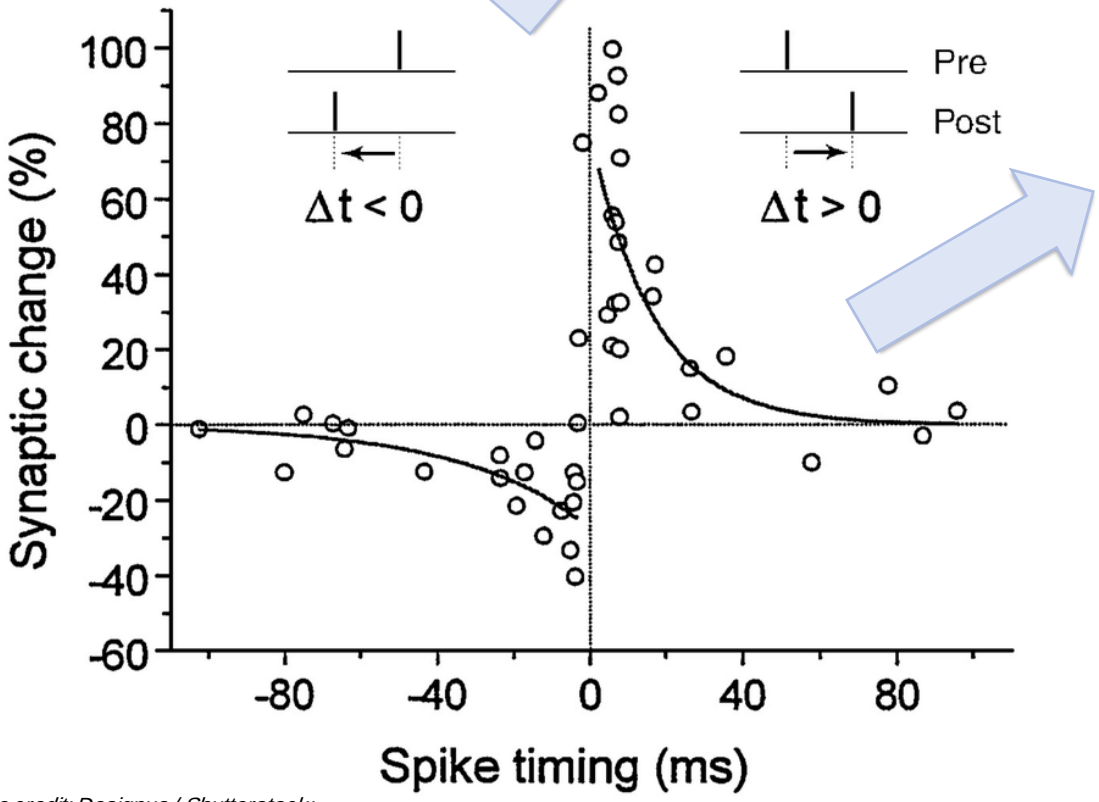


Image credit: Designua / Shutterstock; Bi & Poo 2001

```
synapse stdp:
  state:
    w nS = 1 nS
    tr_post real = 0
    tr_pre real = 0
  end

  equations:
    tr_pre' = tr_pre / tau_tr
    tr_post' = tr_post / tau_tr
  end

  input:
    pre_spikes nS <- spike
    post_spikes nS <- post spike
  end

  preReceive:
    w -= alpha * tr_post        # depress synapse
    tr_pre += 1                 # update presynaptic trace
    deliver_spike(w, delay)     # to postsynaptic partner
  end

  postReceive:
    w += alpha * tr_pre         # potentiate synapse
    tr_post += 1                # update postsynaptic trace
  end

  # parameters: tau_tr, alpha, delay
end
```

nest::ml

JÜLICH Forschungszentrum

What is NESTML?

SBML

NestML

NeuroML (v. 1)

PyNN

NineML

NeuroML (v. 2)

JÜLICH
Forschungszentrum

What is NESTML?

NESTML is a **domain-specific language** for **neuron and synapse models**.

- Low on boilerplate, redundancy
- Speak in the vernacular of the neuroscientist (keywords such as "neuron", "synapse")
- Programming-language-like (`if V_m ≥ threshold: ...`)

NESTML comes with a **code generation toolbox**.

- Code generation (model definition but *not* instantiation)
- Platform-awareness (loose platform coupling)

```
spiking_neuron iaf_psc_delta:
  state:
    V_m mV = 0
  end

  equations:
    shape G = exp(-t / tau_syn)
    V_m' = -(V_m - V_rest)/tau
           + (I_ext + convolve(G, spikes))/C_m
  end

  parameters:
    C_m      pF = 250 pF
    tau      ms = 10 ms
    ...
  end

  input:
    spikes pA <- spike
    I_ext <- current
  end

  update:
    integrate_odes()
    if V_m > threshold:
      V_m = V_reset
      emit_spike()
    end
  end
end
```

Comparison between NESTML and generated code

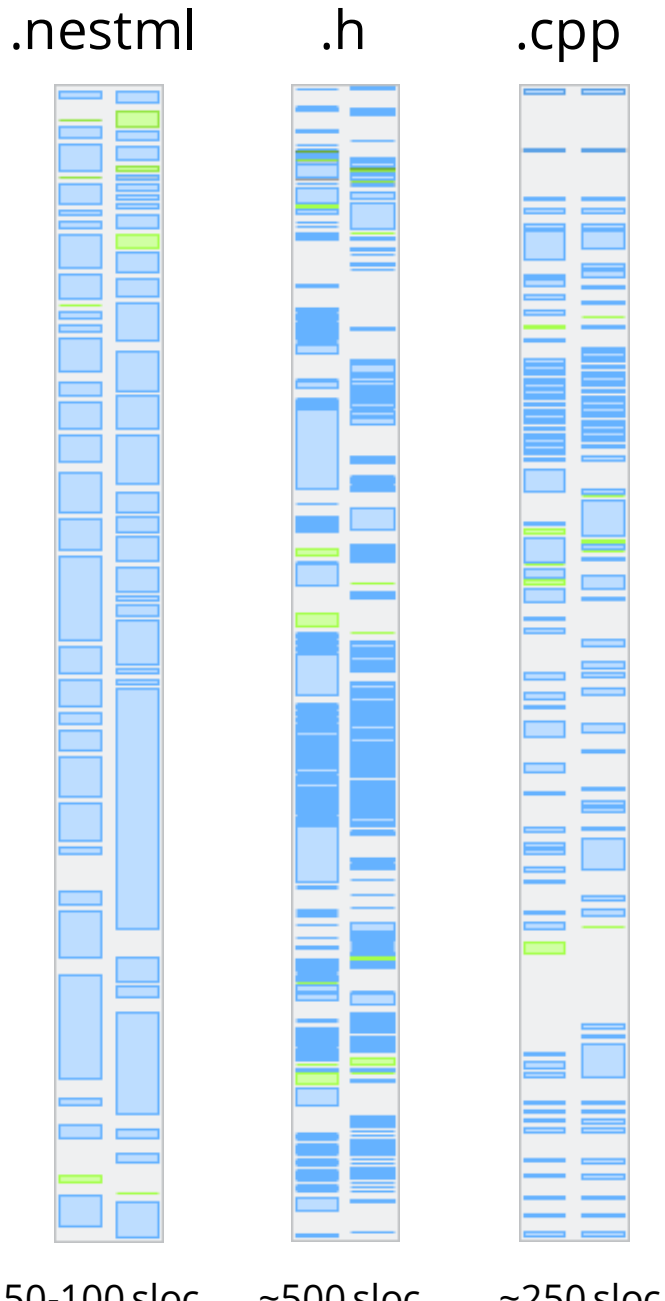
`terub_neuron_gpe` (left)

vs.

`aeif_cond_alpha` (right)

Legend:

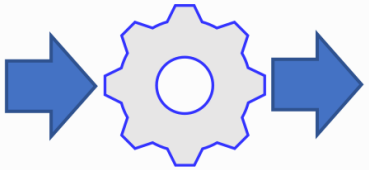
- lines can be matched, but content differs
- lines exist on one side but not the other
- complete match (no difference)

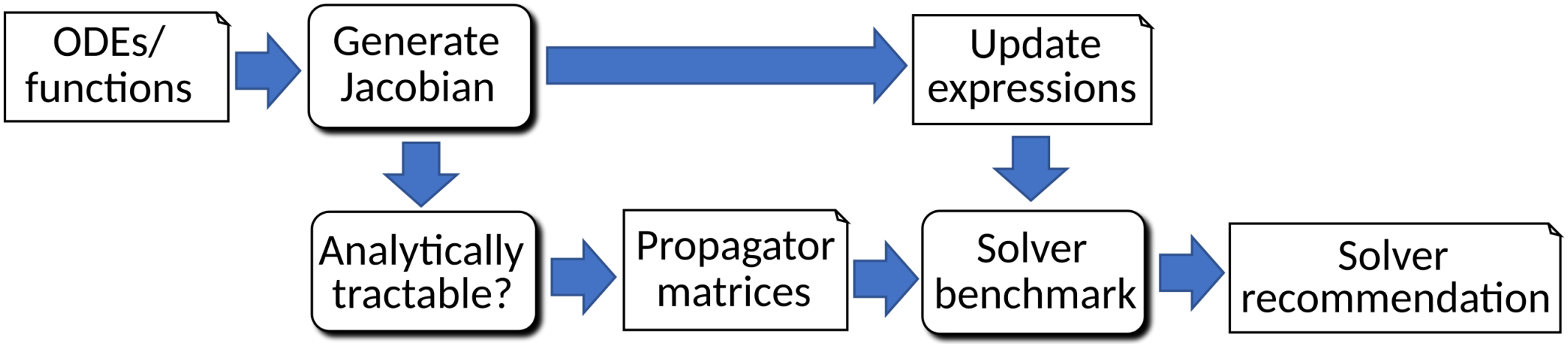


Automated processing of differential equations

```
equations:
    h_Na'   = (h_infNa  - h_Na) / t_hNa
    [...]
    shape g'' = (-2/tau_syn)  * g' - (1/tau_syn**2)  * g
    [...]
    V_m' = (-I_leak  - I_syn_exc  - I_syn_inh  + I_e) / C_m
    [...]
end
```

ODE-toolbox

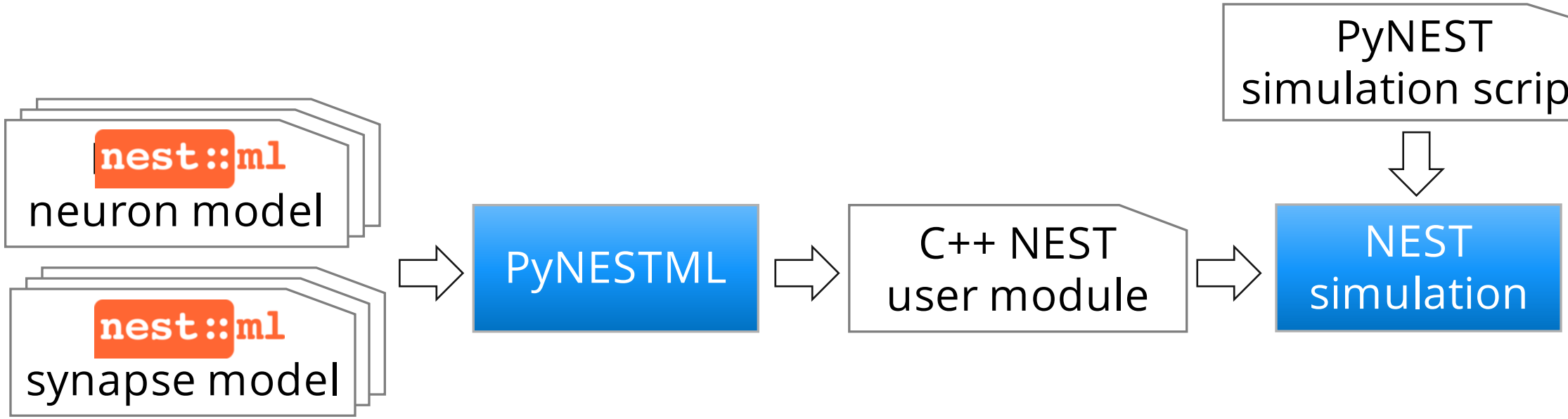




JÜLICH Forschungszentrum

Typical NESTML workflow

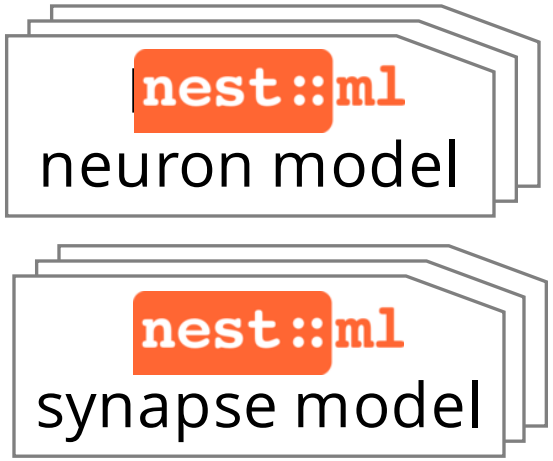
```
>>> nest.Install('nestmlmodule')
>>> pre, post = nest.Create('iaf_psc_exp', 2)
>>> nest.Connect(pre, post,
     'all_to_all', syn_spec={'synapse_model': 'stdp'})
>>> nest.Simulate(1000)
```




```
$ ls -l models
iaf_psc_exp.nestml
stdp.nestml

$ nestml --input_path=models
```

JÜLICH
Forschungszentrum

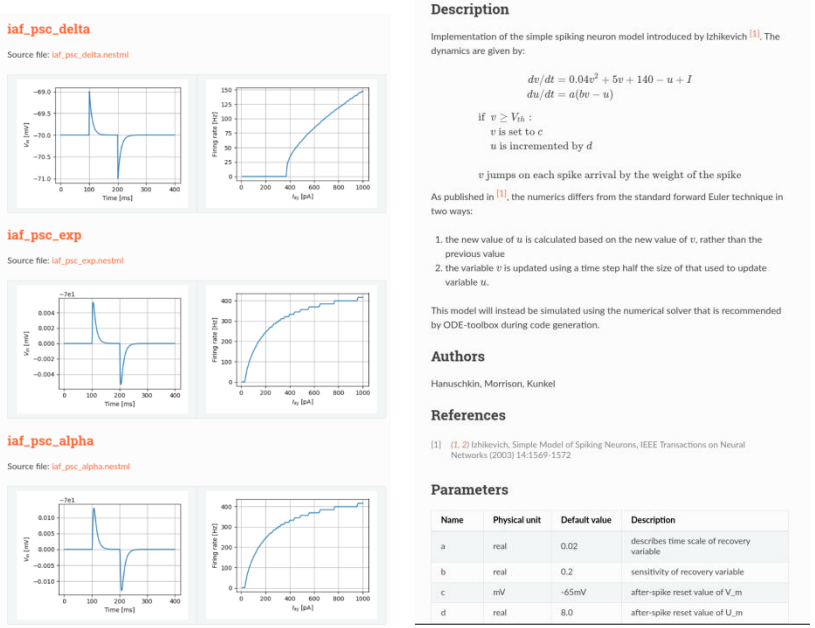
NESTML documentation workflow



PyNESTML

MarkDown

Read the Docs

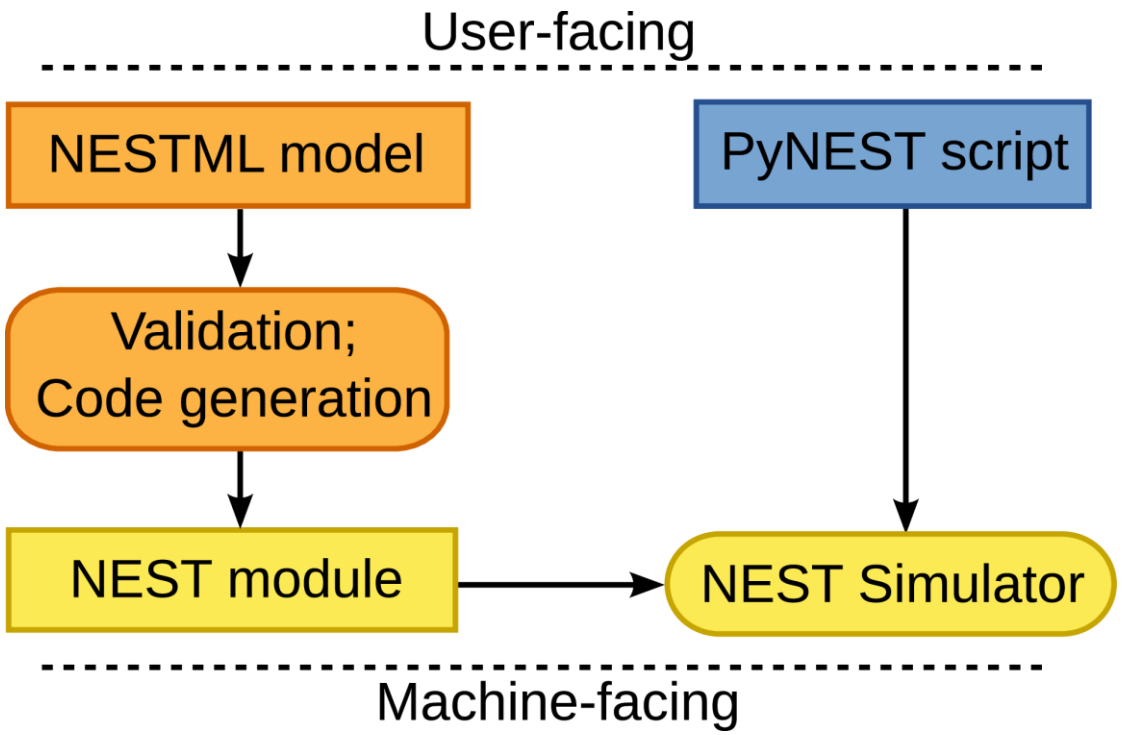


View the NESTML models library at
https://nestml.readthedocs.io/en/latest/models_library/

```
$ ls -l models
iaf_psc_exp.nestml
stdp.nestml

$ nestml --input_path=models
```

JÜLICH
Forschungszentrum

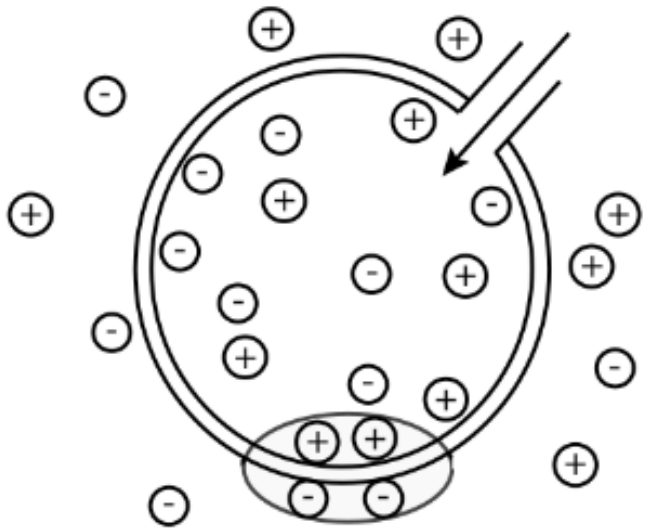
Integration with NEST Simulator



Mapping biological neurons to NESTML

Model

neuron-model based on an RC-circuit



NESTML

<<rc_neuron.nestml>>

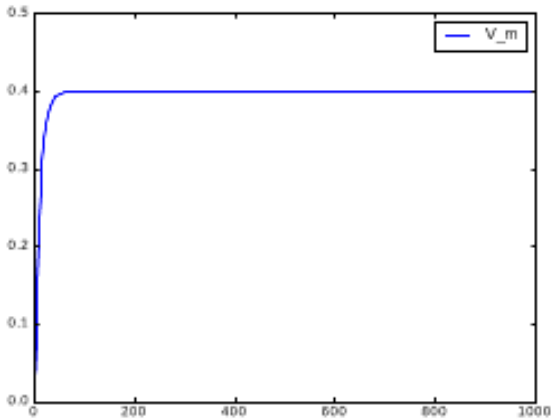
```
neuron rc_neuron:


end
```

JÜLICH
Forschungszentrum

Mapping biological neurons to NESTML

Model

$$\frac{d\,V_\mathrm{m}}{dt} = -\frac{V_\mathrm{m}}{\tau_\mathrm{m}} + \frac{I_\mathrm{syn}}{C_\mathrm{m}}$$



NESTML

<<rc_neuron.nestml>>

```
neuron rc_neuron:

  initial_values:
    V_m mV = 0 mV
  end

  equations:
    V_m' = -V_m/tau_m + I_syn/C_m
  end

  parameters:
    # values taken from experiments
    C_m pF    = 250 pF
    tau_m ms  = 10 ms
    I_syn pA  = 10 pA
  end

  update:
    integrate_odes()
  end

end
```

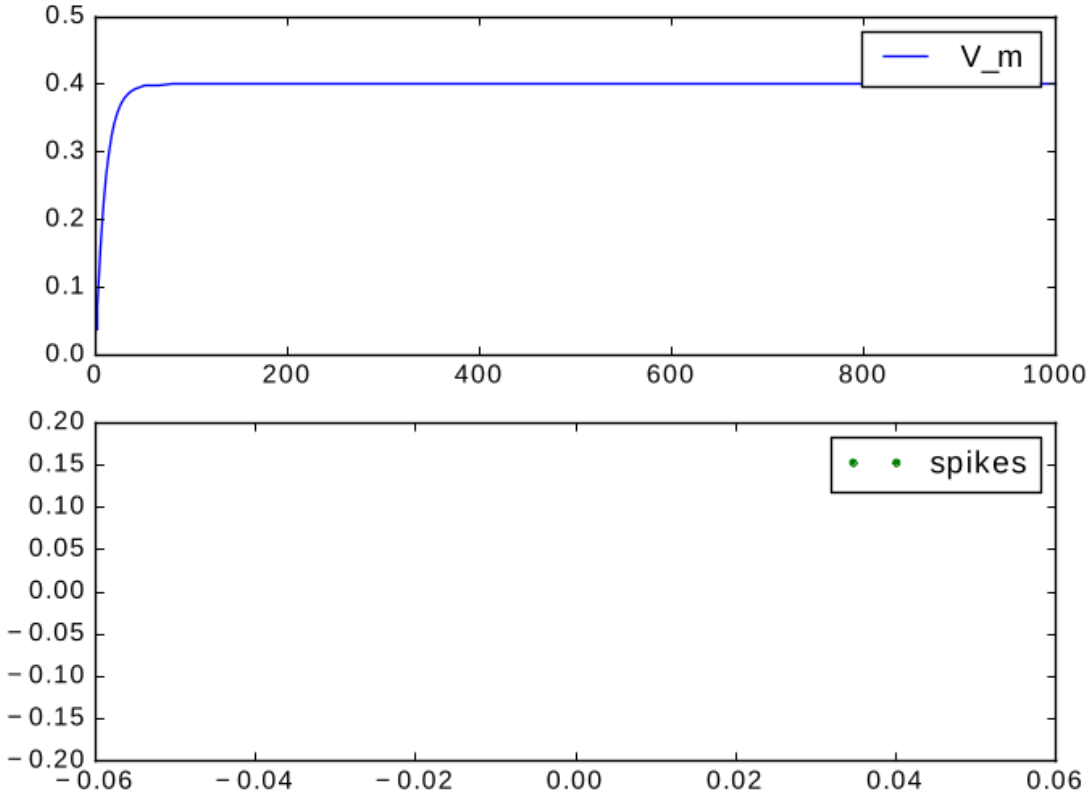
JÜLICH
Forschungszentrum

Mapping biological neurons to NESTML

- Simulating `rc_neuron` for 1000 ms with constant input current of 10 pA

→ Strictly positive membrane potential

→ No spikes



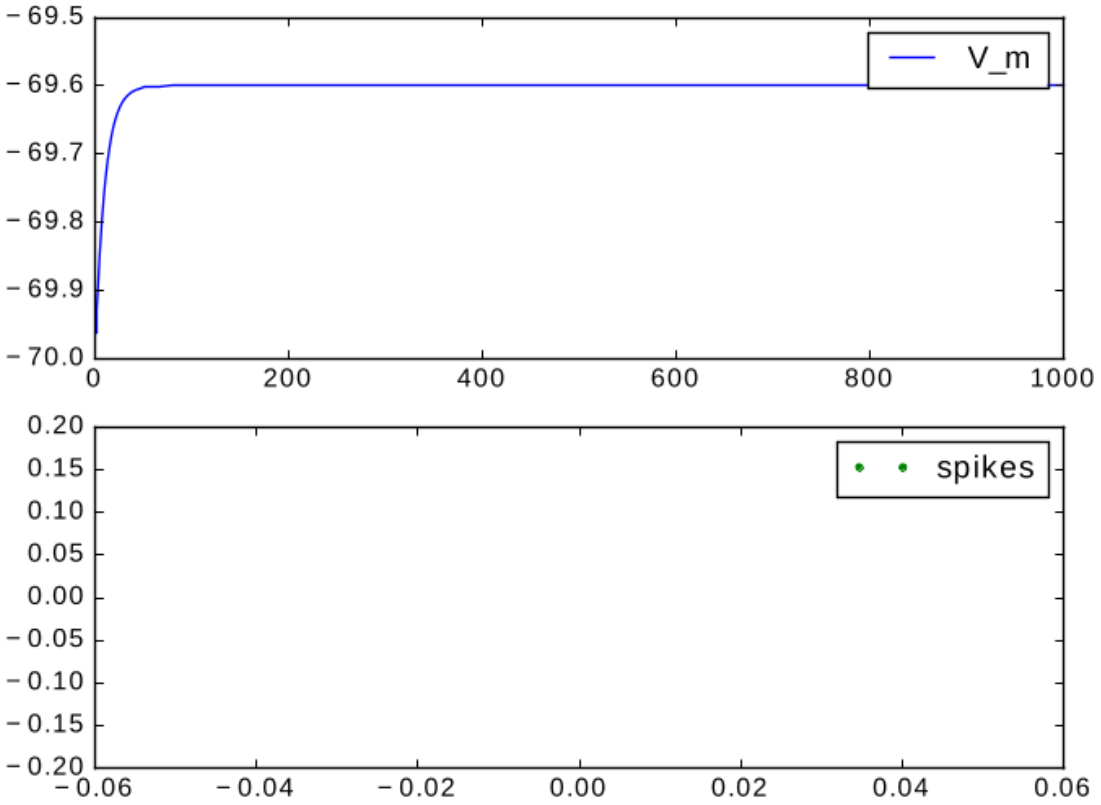
JÜLICH
Forschungszentrum

Adding the resting potential E_L

- Shift V_m by E_L:

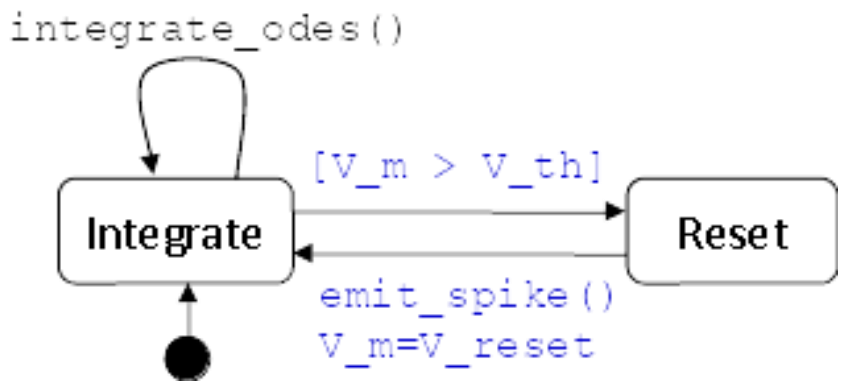
```
neuron rc_neuron_rest:
  initial_values:
    V_m mV = E_L
  end

  equations:
    V_m' = -(V_m-E_L)/tau_m + I_syn/C_m
  end

  parameters:
    E_L mV = -70 mV
  end

  ...
end
```

→ Still no spikes



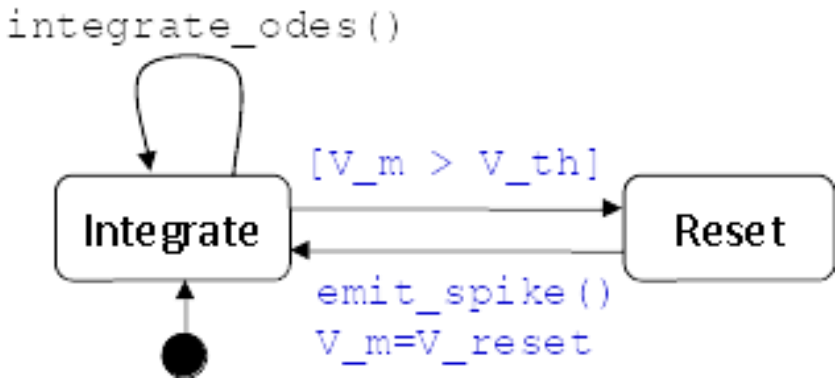
JÜLICH
Forschungszentrum

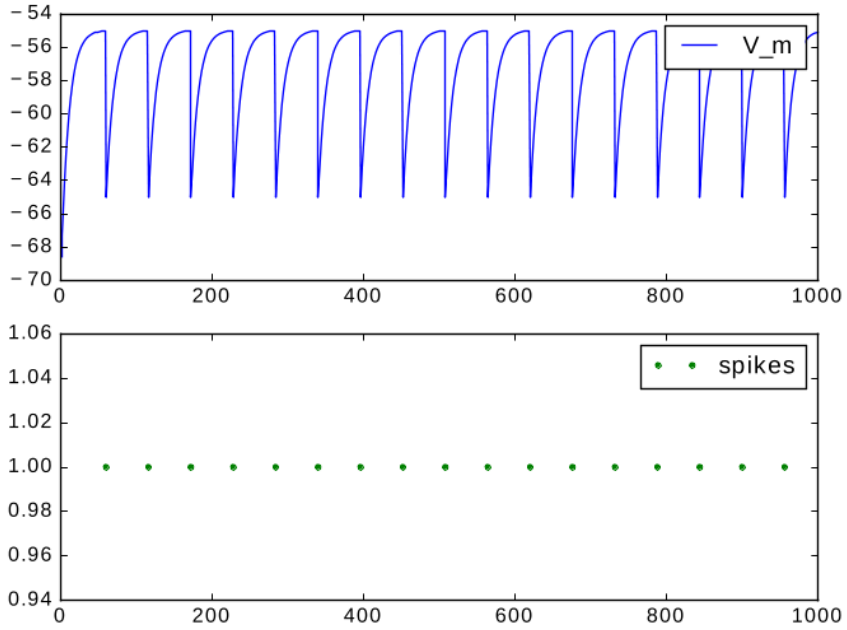
Spiking and reset




```
neuron rc_fire:
  parameters:
    V_th mV = -55 mV - E_L
  end
  update:
    integrate_odes()
    if V_m >= V_th:
      V_m = V_reset
      emit_spike()
    end
  end
  ...
end
```

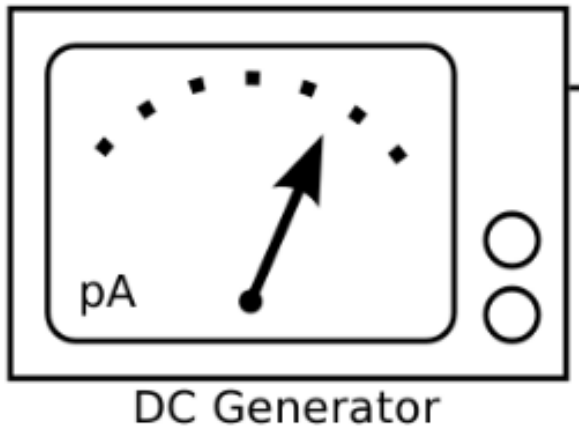
Spiking and reset





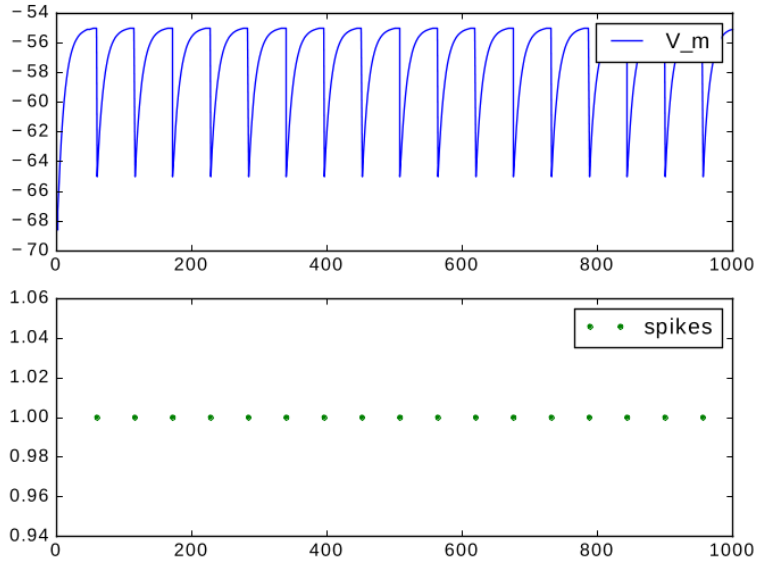
JÜLICH
Forschungszentrum

Injecting currents

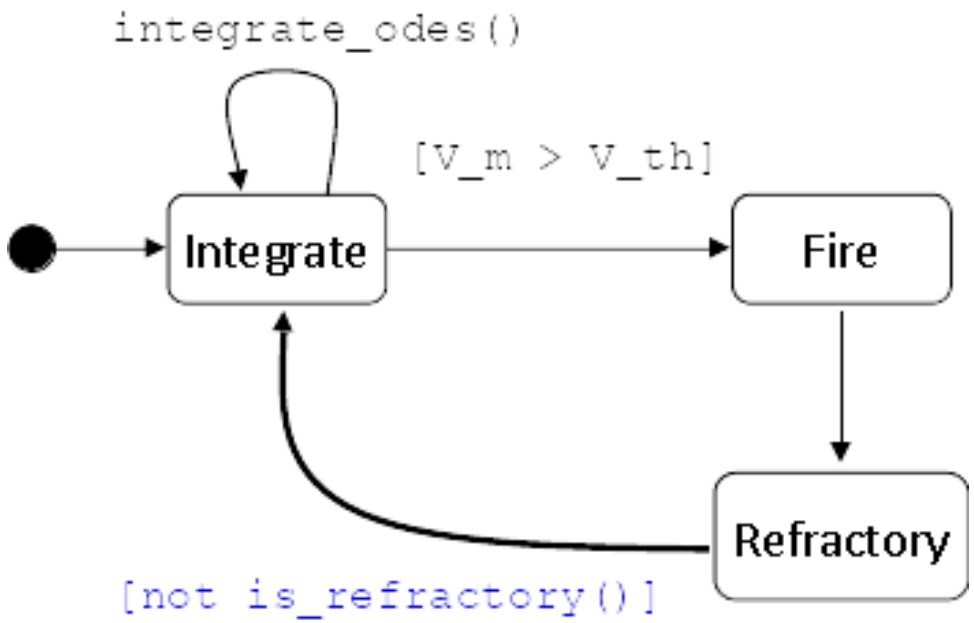



```
currents = nest.Create('dc_generator', 1,
                       {'amplitude':
100.0})
nest.Connect(currents, rc_neuron)
```

```
neuron rc_neuron:
...
  equations:
     function I_syn pA = I_e + convolve(I_a, spikes) + currents
     V_m' = -V_m/tau_m + I_syn/C_m
  end

  input:
     currents pA <- current
     spikes pA <- spike
  end

  output: spike

...

end
```



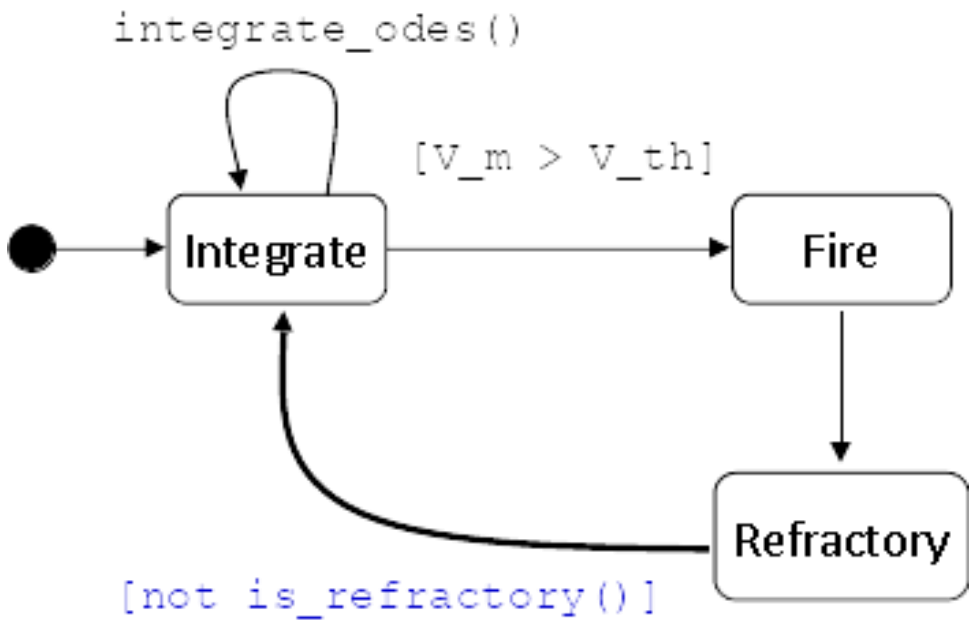
Refractoriness

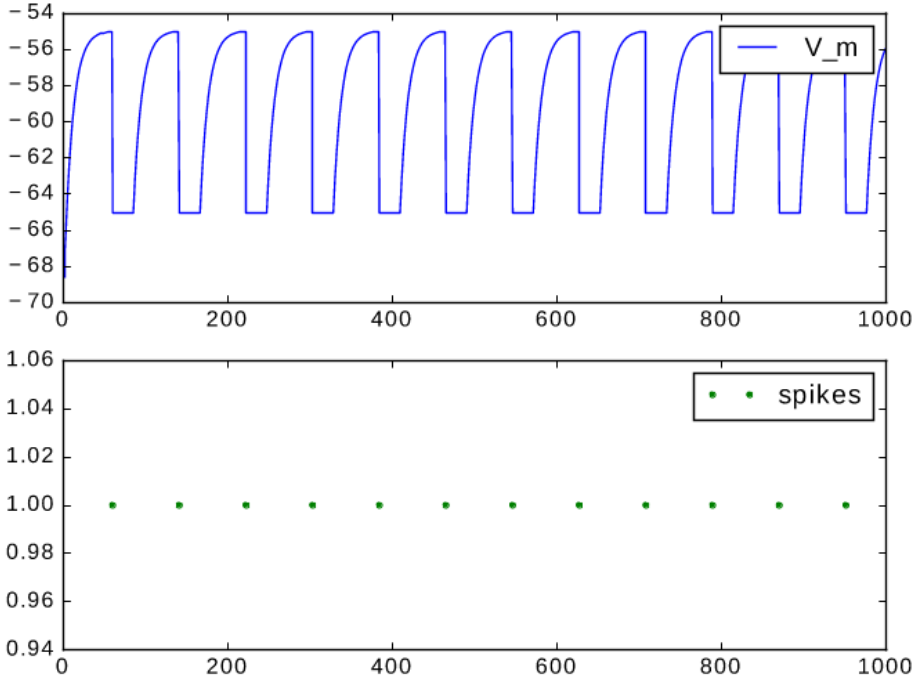



```
neuron rc_refractory:
  parameters:
    ref_counts integer = 0
    ref_timeout ms = 2 ms
  end
  internals:
    timeout_ticks integer = steps(ref_timeout)
  end
  update:
    if ref_counts == 0:
      integrate_odes()
      if V_m >= V_th:
        emit_spike()
        ref_counts = timeout_ticks
        V_m = V_reset
      end
    else:
      ref_counts -= 1
    end
  end
end
```

JÜLICH
Forschungszentrum

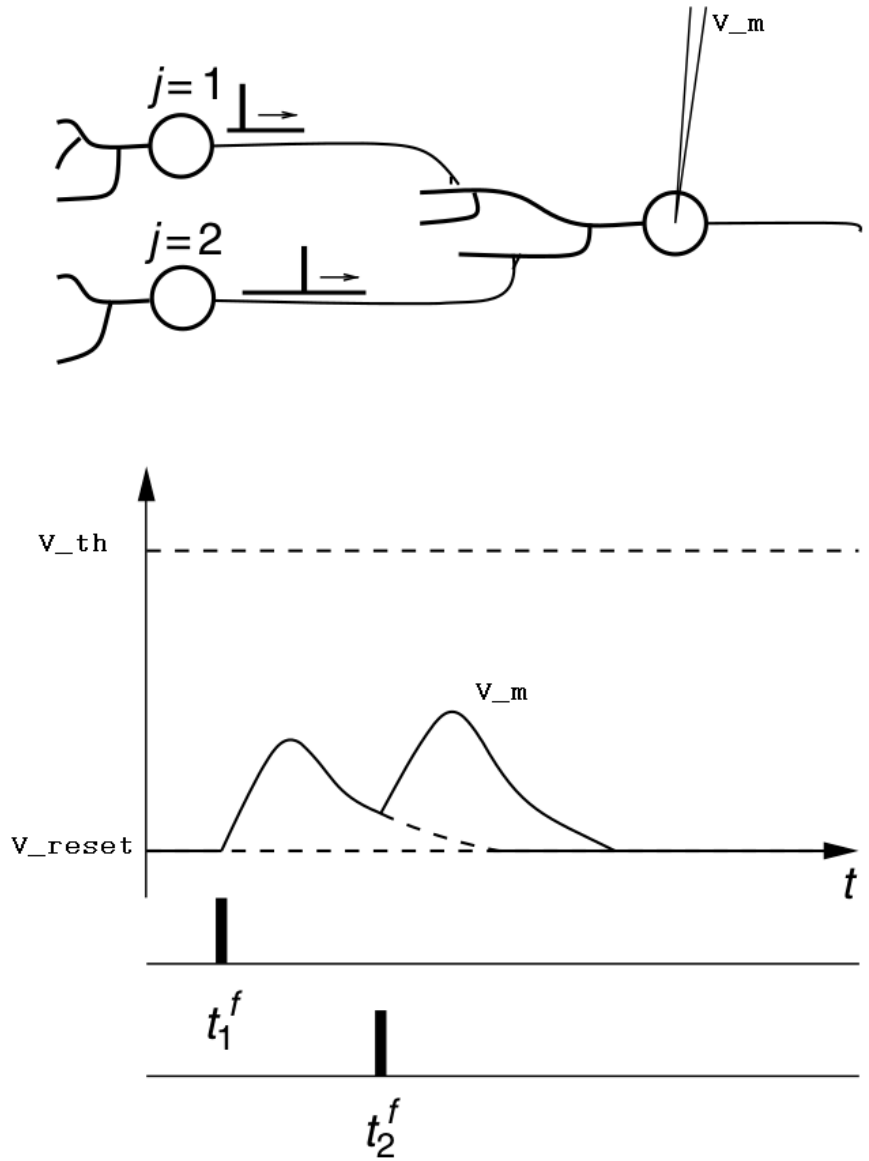
Refractoriness





JÜLICH
Forschungszentrum

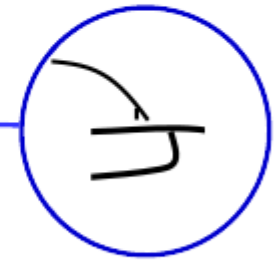
Input handling



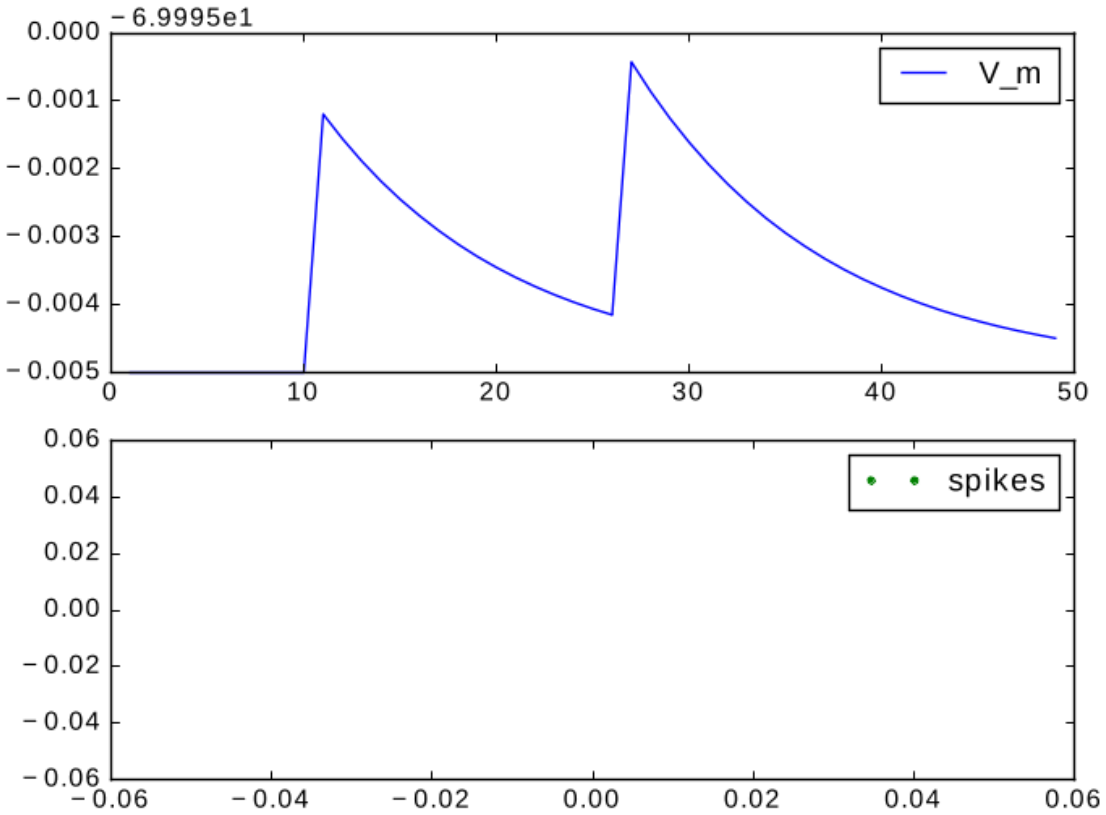
(Source: Wulfram Gerstner, Werner M. Kistler, Richard Naud, Liam Paninski-Neuronal Dynamics From Single Neurons to Networks and Models of Cognition)

Input handling

```
neuron rc_input:
  initial_values:
    V_m mV = E_L
  end

  equations:
    V_m' = -(V_m-E_L)/tau_m + I_syn/C_m
  end

  parameters:
    E_L mV = -70 mV
    ...
  end

  input:
    I_syn pA <- spike
  end

  output: spike

end
```



buffer can be `inhibitory`, `excitatory` or both (if nothing else stated)

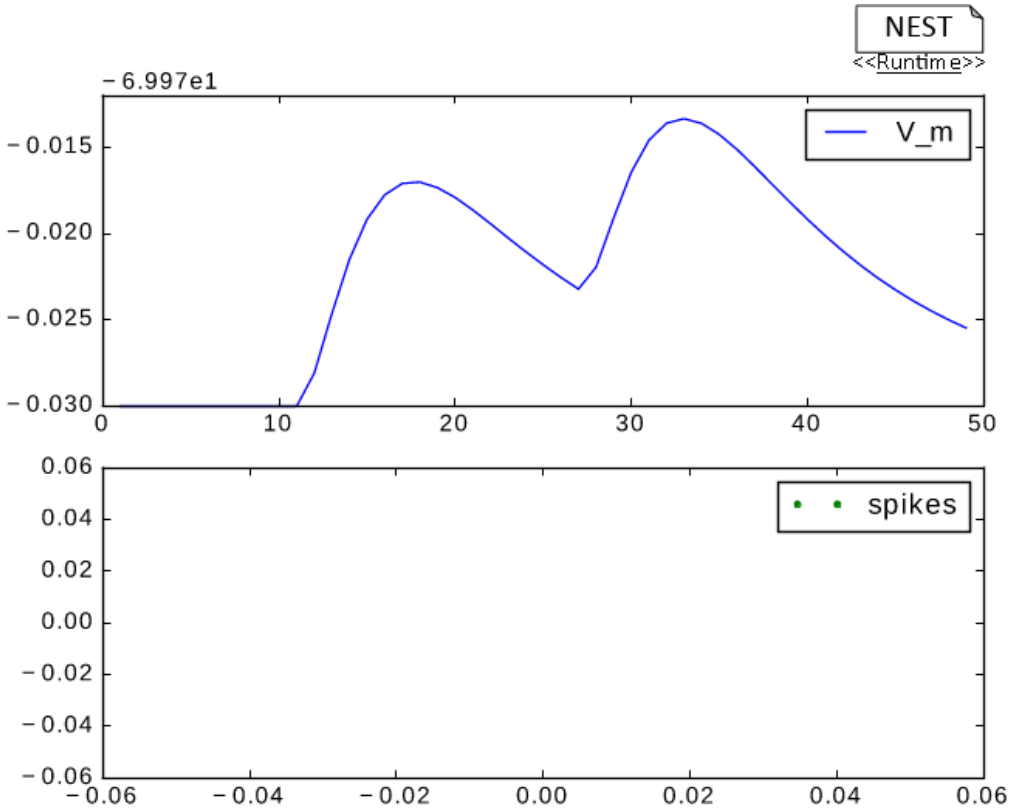


JÜLICH
Forschungszentrum

Input handling

```
neuron rc_alpha_response:
  initial_values:
    V_m mV = E_L
    I_a pA = 0 pA
    I_a' pA/ms = e/tau_syn * pA
  end


  equations:
    shape I_a'' = -2 * I_a' / tau_syn - I_a / tau_syn**2
    V_m' = -(V_m-E_L)/tau_m + convolve(I_a, spikes)/C_m
  end

  input:
    spikes pA <- spike
  end

  output: spike

  update:
    integrate_odes()
    ...
  end

end
```

$$\sum_{t_i \le t, i \in \mathbb{N}} \sum_{w \in W} w \cdot I_a(t_i - t) = \sum_{t_i \le t, i \in \mathbb{N}} I_a(t_i - t) \sum_{w \in W} w$$

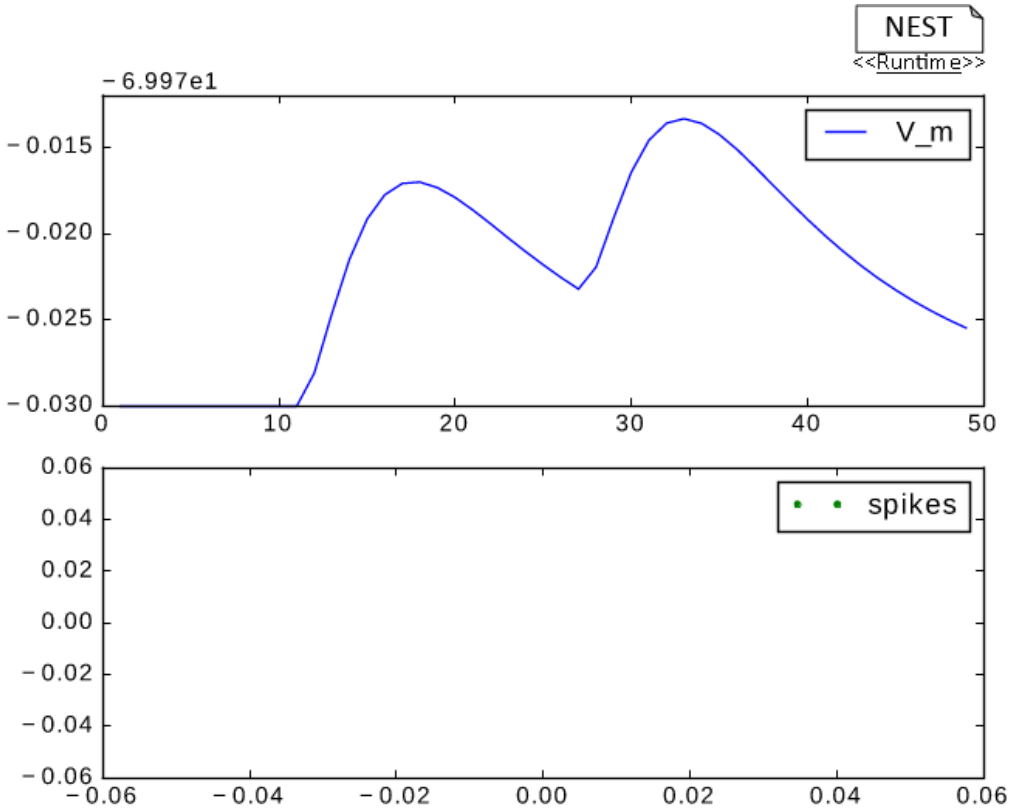


JÜLICH
Forschungszentrum

Shape notation

```
neuron rc_alpha_response_shape:
  state:
    V_m mV = E_L
  end


  equations:
    shape I_a = (e/tau_syn) * t * exp(-t/tau_syn)
    V_m' = -(V_m-E_L)/tau_m + convolve(I_a, spikes)/C_m
  end

  input:
    spikes pA <- spike
  end

  output: spike

  update:
    integrate_odes()
    ...
  end


end
```

initial values
computed automatically



JÜLICH
Forschungszentrum

PyNEST API of generated NEST module

PyNEST

```
import nest

nest.Install("nestmlmodels")

neuron = Create("rc_neuron")
nest.SetStatus(neuron, {"V_m":-72.0,
                        "C_m":300.0})

mm = nest.Create('multimeter')
nest.SetStatus(mm, {"record_from":["V_m"]})
Connect(mm, neuron)
```

NESTML

<<rc_neuron.nestml>>

```
neuron rc_neuron:

  initial_values:
    V_m mV = -70mV
  end

  equations:
    V_m' = -(V_m-70mV)/tau_m + I_syn/C_m
  end

  parameters:
    C_m pF    = 250pF
    tau_m ms  = 10ms
    I_syn     = 10pA
  end

end
```

JÜLICH Forschungszentrum

Thank you!

nest::ml

https://github.com/nest/nestml

Jochen M. Eppler
Abigail Morrison
Markus Diesmann
Konstantin Perun
Dimitri Plotnikov
Inga Blundell
Tanguy Fardet
Jessica Mitchell
Sara Konradi

This project has received funding from the Helmholtz Association through the Helmholtz Portfolio Theme "Supercomputing and Modeling for the Human Brain" and the European Union's Horizon 2020 research and innovation programme under grant agreement No 720270 (HBP SGA1). and No 785907 (HBP SGA2).

JÜLICH
Forschungszentrum
Member of the Helmholtz Association