NESTML
Tutorial

I.Blundell, M.J.Eppler, D.Plotnikov

Software Engineering
RWTH Aachen

http://www.se-rwth.de/

Usage of the NESTML Infrastructure

- Starting eclipse:

```
cd /home/nest/eclipse_nestml
./eclipse
```

- Working folder for the code generation:

```
/home/nest/nestml_workshop/nestml_workshop_project
```

- Console-tool for the codegeneration

  `java -jar nestml-core-0.0.3-SNAPSHOT-jar-with-dependencies.jar` *`pathToFile.nestml`*

  - Optional parameters:
    - **--target** generationPath (current directory if omitted)

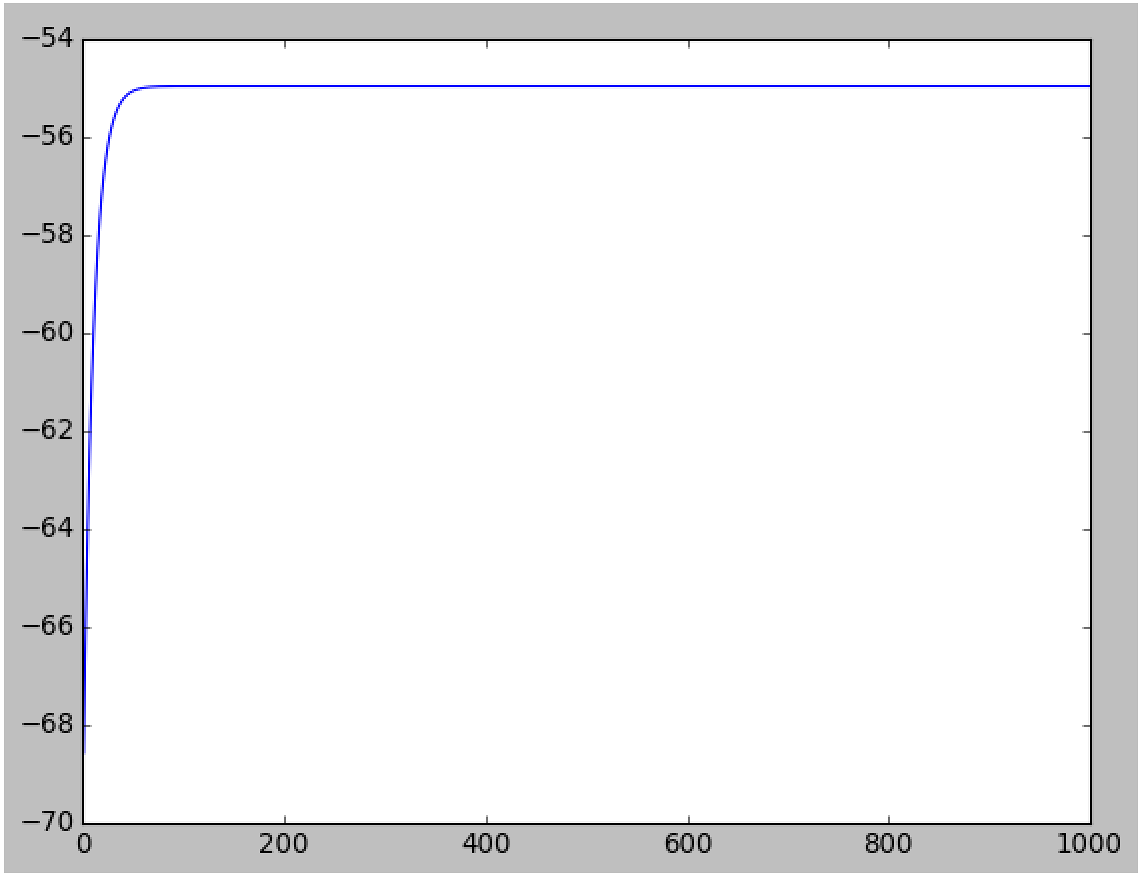
- Change to the generated folder
  - `cd codegeneration\neuron_level_1` (or _2, _3 for particular task)
- Execute the following 3 commands (enter them individually)

```
sh bootstrap.sh
./configure --with-nest=${NEST_INSTALL_DIR}/bin/nest-config
make && make install
```

Blundell, Plotnikov, Eppler
Lehrstuhl für Software Engineering
RWTH Aachen

Task 1: Simple Case Integrate neuron 1/2

- Implement a simple integrate neuron
  - The neuron doesn’t spike, but integrates over the time



Task 1: Simple Case
Integrate neuron 2/2

- Use the template neuron_level_1.nestml
  - Fix errors showed by the editor
  - Fill/extend TODO

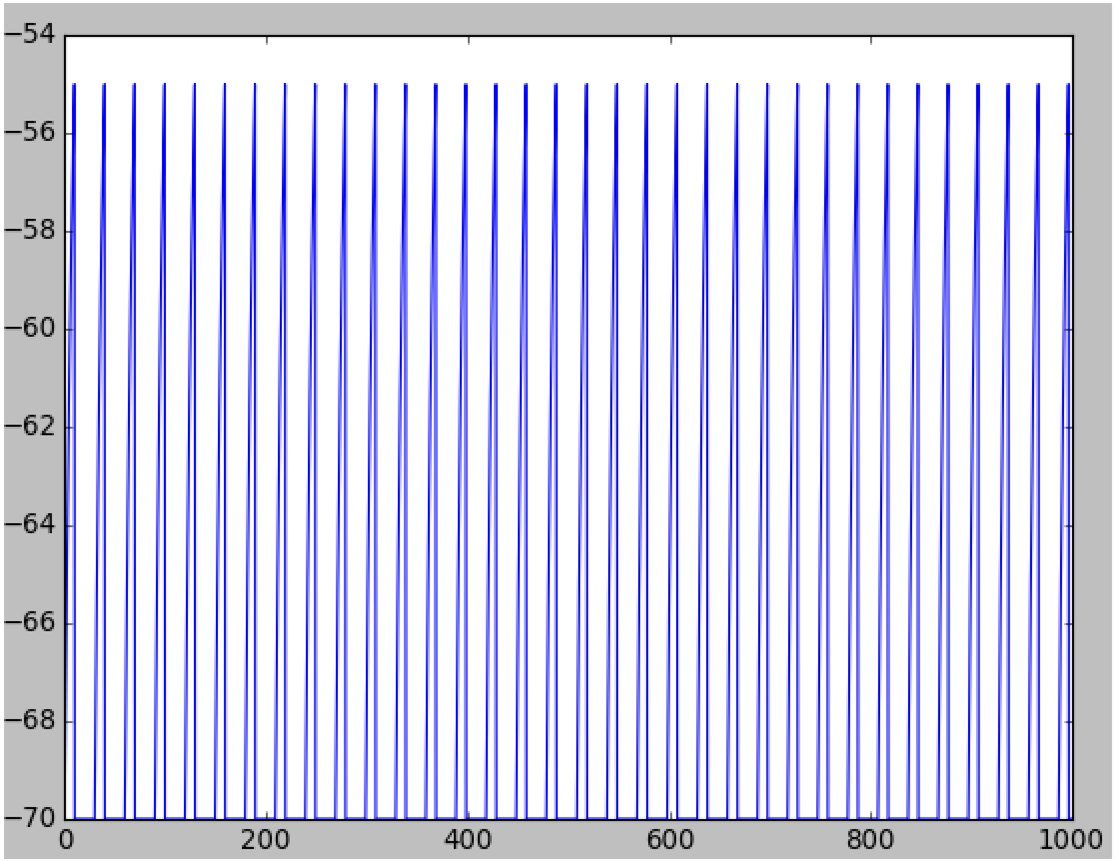
- The dynamics is described as:

$$G := \frac{E}{tau\_syn} * t * exp(\frac{-1}{tau\_syn} * t)$$
$$\frac{d}{dt}V := \frac{-1}{Tau} * V + \frac{1}{C\_m} * G + I_e + cur$$

- Use tester_workshop_neuron_level_1.py to test

Task 2: Threshold
Integrate and fire neuron 1/2

- Add the threshold test in the dynamics
- Increase the refractory time to 20 ms



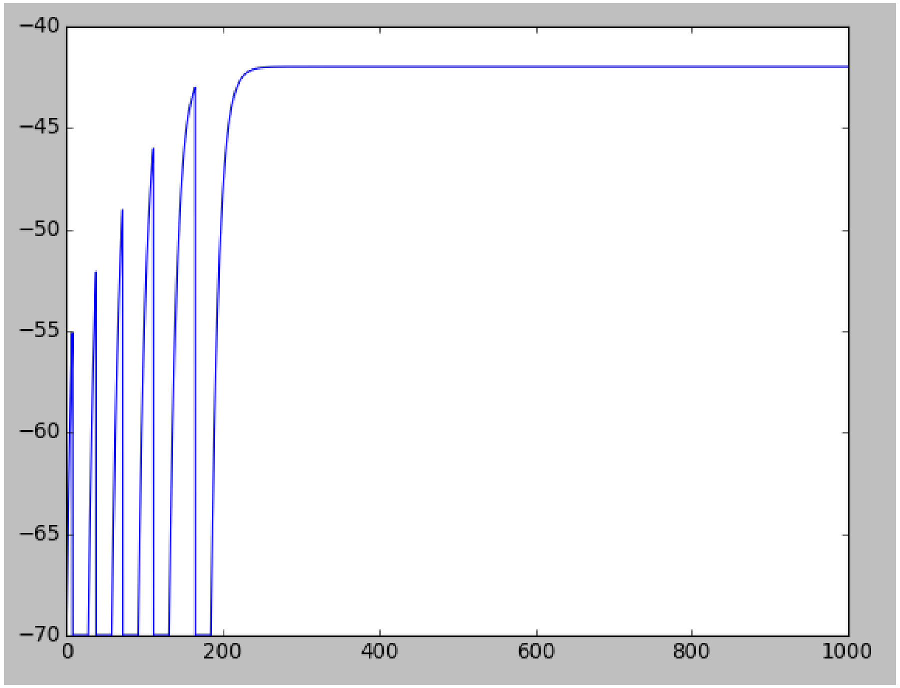
Blundell, Plotnikov, Eppler
Lehrstuhl für Software Engineering
RWTH Aachen

Task 2: Threshold
Integrate and fire neuron 2/2

- Use the template neuron_level_2.nestml
  - Fill/extend TODOs
- Implement threshold crossing using the variable `thresholdTheta`
- Use python tester_workshop_neuron_level_2.py to test

Task 3: Adaptive Threshold 1/2

- Make an adaption of the threshold after each spiking



Task 3: Adaptive Threshold 2/2

- Use the template neuron_level_3.nestml
    - Fill/extend TODO

- Use a threshold adaption, e.g. Theta = Theta + 3 after spiking

- Use tester_workshop_neuron_level_3.py to test

Language Concepts

Procedural Language: Declarations

Multiple variables in the same declaration

Type

Optional initial value

```
a, b, c real = 0

x real = 3; y real = 4; z real

f real = -2e12
```

Possible types:
integer, real, string, ms, mV, ...

Blundell, Plotnikov, Eppler
Lehrstuhl für
Software Engineering
RWTH Aachen

Simple Programming Language
Function Calls

```
base, power real = 0

pow(base, power)
```

Function name → `pow`

Parameters → `base, power`

Important pre-defiend functions:

emitSpike(): emits spike
exp(x): Returns the base-e exponential function of x, which is e raised to the power x: e^x
pow(base, power): raises base to the power exponent.

Constants:
E: Euler's number

Simple Programming Language: Control flow 1/2

```
if 2 < 3:
  ...
end
```

```
if 2 < 3:
  ...
else:
  ...
end
```

```
if 2 < 3:
  ...
elif 4>6:
  ...
else:
  ...
end
```

```
x real
for x in 1 ... 5 :

end
```

```
x real
for x in 1 ... 5 step 2:

end
```

```
x real
for x in 1 ... -5.6 step 0.1:

end
```

```
x, y real
x = 1
y = 2
while x <= 10:
    y = x*2
    x = x+1
end
```

NESTML
Model structure

Package name. Relevant for model crossreferences.

neuron name

Declarations are possible, same for parameter, internals

Mandatory part describing inputs

Mandatory part describing outputs

Dynamics definition

Entire SPL code is possible

```
package testing:

neuron WorkingNeuron:
  state:
     i_0       mV
  end

  input:
    spikeBuffer   <- inhibitory excitatory spike
  end

  output: spike

  dynamics timestep(t ms):

  end

end
```

Buffer Blocks

"spike" and "current" are possible

```
input:
  bufferName <- inhibitory excitatory spike
end
```

"inhibitory", "excitatory", both or none are possible

"spike" and "current" are possible

```
output: spike
```

Simple Programming Language Differential Equations

```
dynamics timestep(t ms):
  ODE:
   G := E/tau_syn) * t * exp(-1/tau_syn*t)

   d/dt V := -1/Tau * V + 1/C_m * G + I_e + cur
  end
end
```

Optional current declarations as equations (zero or more)

One ore more differential equations