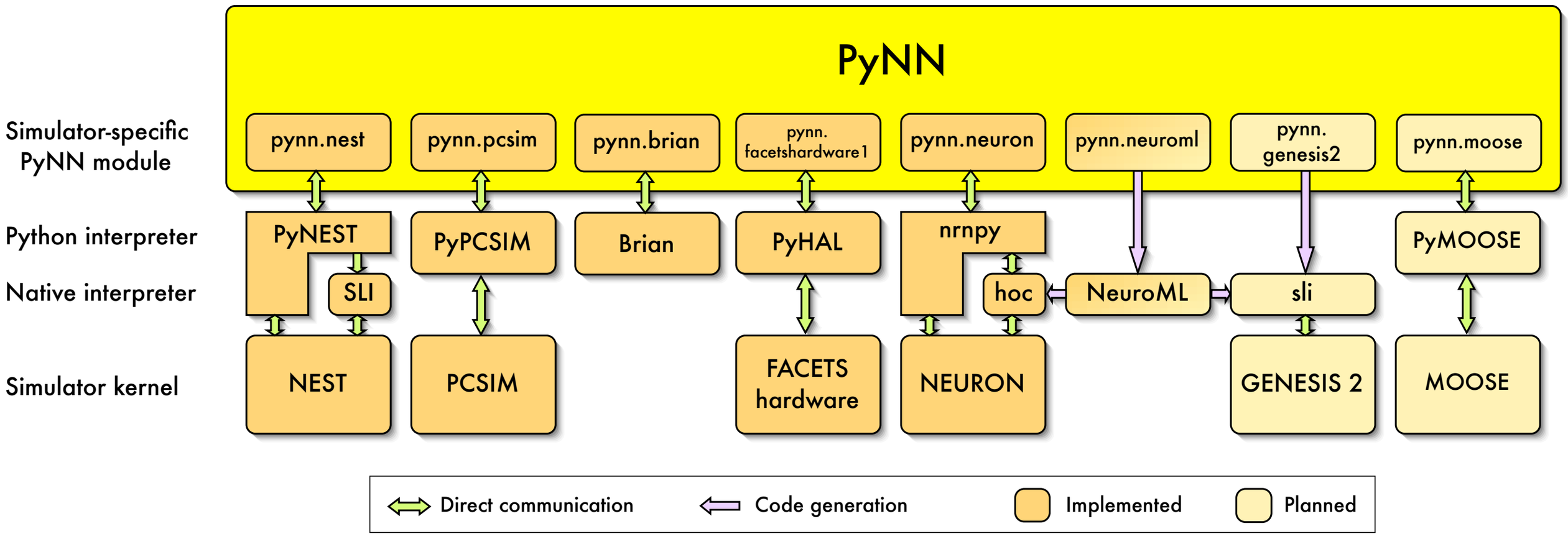
# PyNN and NineML

Andrew Davison
UNIC, CNRS
BrainScaleS CodeJam #5
Edinburgh, 16th March 2012

# PyNN: write the code for a simulation once, run it on any supported simulator or hardware device *without modification*.




http://neuralensemble.org/PyNN

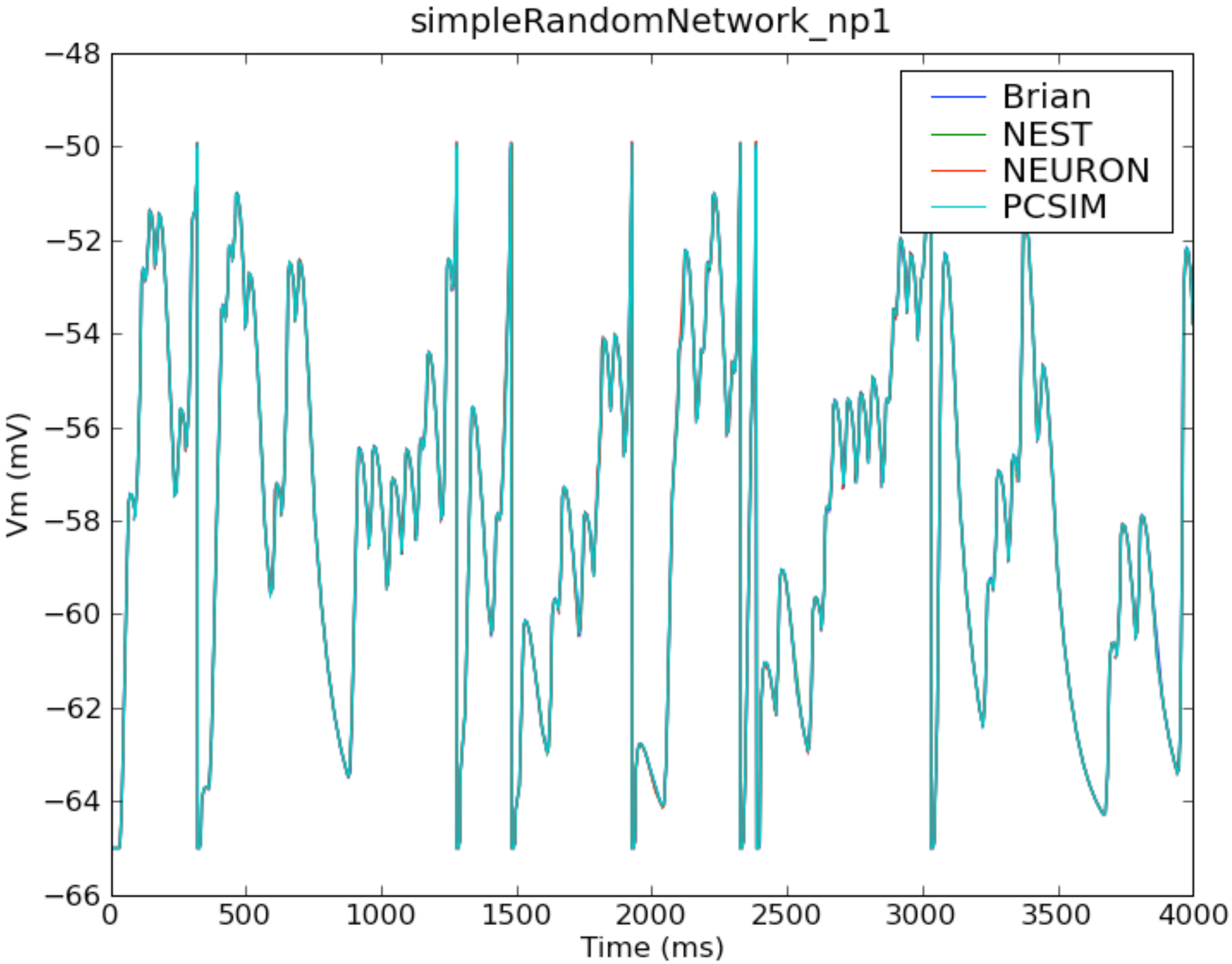
```
sim.setup(timestep=0.1)
cell_parameters = {”tau_m”: 12.0, ”cm”: 0.8, ”v_thresh”: -50.0,
                   ”v_reset”: -65.0}
pE = sim.Population((100,100), sim.IF_cond_exp, cell_parameters,
                    label=”excitatory neurons”)
pI = sim.Population((50,50), sim.IF_cond_exp, cell_parameters,
                    label=”inhibitory neurons”)
all = pE + pI
input = sim.Population(100, sim.SpikeSourcePoisson)
rate_distr = random.RandomDistribution(”normal”, (10.0, 2.0))
input.rset(”rate”, rate_distr)
background = sim.NoisyCurrentSource(mean=0.1, stdev=0.01)
all.inject(background)
weight_distr = random.RandomDistribution(”uniform”, (0.0, 0.1))
DDPC = sim.DistanceDependentProbabilityConnector
connector = DDPC(”exp(-d**2/400.0)”, weights=weight_distr,
                 delays=”0.5+0.01d”)
TMM = sim.TsodyksMarkramMechanism
depressing = sim.DynamicSynapse(fast=TMM(U=0.5,tau_rec=800.0))
exc = sim.Projection(pE, all, connector, target=”excitatory”,
                     synapse_dynamics=plasticity)
inh = sim.Projection(pI, all, connector, target=”inhibitory”)
```

```
import pyNN.neuron as sim
sim.setup(timestep=0.1)
cell_parameters = {”tau_m”: 12.0, ”cm”: 0.8, ”v_thresh”: -50.0,
                   ”v_reset”: -65.0}
pE = sim.Population((100,100), sim.IF_cond_exp, cell_parameters,
                    label=”excitatory neurons”)
pI = sim.Population((50,50), sim.IF_cond_exp, cell_parameters,
                    label=”inhibitory neurons”)
all = pE + pI
input = sim.Population(100, sim.SpikeSourcePoisson)
rate_distr = random.RandomDistribution(”normal”, (10.0, 2.0))
input.rset(”rate”, rate_distr)
background = sim.NoisyCurrentSource(mean=0.1, stdev=0.01)
all.inject(background)
weight_distr = random.RandomDistribution(”uniform”, (0.0, 0.1))
DDPC = sim.DistanceDependentProbabilityConnector
connector = DDPC(”exp(-d**2/400.0)”, weights=weight_distr,
                 delays=”0.5+0.01d”)
TMM = sim.TsodyksMarkramMechanism
depressing = sim.DynamicSynapse(fast=TMM(U=0.5,tau_rec=800.0))
exc = sim.Projection(pE, all, connector, target=”excitatory”,
                     synapse_dynamics=plasticity)
inh = sim.Projection(pI, all, connector, target=”inhibitory”)
```

```
import pyNN.nest as sim
sim.setup(timestep=0.1)
cell_parameters = {”tau_m”: 12.0, ”cm”: 0.8, ”v_thresh”: -50.0,
                   ”v_reset”: -65.0}
pE = sim.Population((100,100), sim.IF_cond_exp, cell_parameters,
                    label=”excitatory neurons”)
pI = sim.Population((50,50), sim.IF_cond_exp, cell_parameters,
                    label=”inhibitory neurons”)
all = pE + pI
input = sim.Population(100, sim.SpikeSourcePoisson)
rate_distr = random.RandomDistribution(”normal”, (10.0, 2.0))
input.rset(”rate”, rate_distr)
background = sim.NoisyCurrentSource(mean=0.1, stdev=0.01)
all.inject(background)
weight_distr = random.RandomDistribution(”uniform”, (0.0, 0.1))
DDPC = sim.DistanceDependentProbabilityConnector
connector = DDPC(”exp(-d**2/400.0)”, weights=weight_distr,
                 delays=”0.5+0.01d”)
TMM = sim.TsodyksMarkramMechanism
depressing = sim.DynamicSynapse(fast=TMM(U=0.5,tau_rec=800.0))
exc = sim.Projection(pE, all, connector, target=”excitatory”,
                     synapse_dynamics=plasticity)
inh = sim.Projection(pI, all, connector, target=”inhibitory”)
```

```
import pyNN.brian as sim
sim.setup(timestep=0.1)
cell_parameters = {”tau_m”: 12.0, ”cm”: 0.8, ”v_thresh”: -50.0,
                   ”v_reset”: -65.0}
pE = sim.Population((100,100), sim.IF_cond_exp, cell_parameters,
                    label=”excitatory neurons”)
pI = sim.Population((50,50), sim.IF_cond_exp, cell_parameters,
                    label=”inhibitory neurons”)
all = pE + pI
input = sim.Population(100, sim.SpikeSourcePoisson)
rate_distr = random.RandomDistribution(”normal”, (10.0, 2.0))
input.rset(”rate”, rate_distr)
background = sim.NoisyCurrentSource(mean=0.1, stdev=0.01)
all.inject(background)
weight_distr = random.RandomDistribution(”uniform”, (0.0, 0.1))
DDPC = sim.DistanceDependentProbabilityConnector
connector = DDPC(”exp(-d**2/400.0)”, weights=weight_distr,
                 delays=”0.5+0.01d”)
TMM = sim.TsodyksMarkramMechanism
depressing = sim.DynamicSynapse(fast=TMM(U=0.5,tau_rec=800.0))
exc = sim.Projection(pE, all, connector, target=”excitatory”,
                     synapse_dynamics=plasticity)
inh = sim.Projection(pI, all, connector, target=”inhibitory”)
```

```
import pyNN.hardware.facets.stage1 as sim
sim.setup(timestep=0.1)
cell_parameters = {”tau_m”: 12.0, ”cm”: 0.8, ”v_thresh”: -50.0,
                   ”v_reset”: -65.0}
pE = sim.Population((100,100), sim.IF_cond_exp, cell_parameters,
                    label=”excitatory neurons”)
pI = sim.Population((50,50), sim.IF_cond_exp, cell_parameters,
                    label=”inhibitory neurons”)
all = pE + pI
input = sim.Population(100, sim.SpikeSourcePoisson)
rate_distr = random.RandomDistribution(”normal”, (10.0, 2.0))
input.rset(”rate”, rate_distr)
background = sim.NoisyCurrentSource(mean=0.1, stdev=0.01)
all.inject(background)
weight_distr = random.RandomDistribution(”uniform”, (0.0, 0.1))
DDPC = sim.DistanceDependentProbabilityConnector
connector = DDPC(”exp(-d**2/400.0)”, weights=weight_distr,
                 delays=”0.5+0.01d”)
TMM = sim.TsodyksMarkramMechanism
depressing = sim.DynamicSynapse(fast=TMM(U=0.5,tau_rec=800.0))
exc = sim.Projection(pE, all, connector, target=”excitatory”,
                     synapse_dynamics=plasticity)
inh = sim.Projection(pI, all, connector, target=”inhibitory”)
```


simpleRandomNetwork_np1
Brian
NEST
NEURON
PCSIM
Vm (mV)
−48
−50
−52
−54
−56
−58
−60
−62
−64
−66
0
500
1000
1500
2000
2500
3000
3500
4000
Time (ms)

2012 March

June 0.8

API cleaning

Functions as parameters

Neo output

9ML cell and synapse models

MOOSE backend

NeMo backend

NeuroML export

Developers' guide

Sphinx docs

Split cell and synapse models

MUSIC integration

9ML synaptic plasticity models

9ML export

9ML import

NEST-SLI backend

2013 December 0.9

Multi-compartmental models

2012 2013

March

June

December

0.8

0.9

API cleaning

Functions as parameters

Neo output

9ML cell and synapse models

MOOSE backend

NeMo backend

NeuroML export

Developers' guide

Sphinx docs

Split cell and synapse models

MUSIC integration

9ML synaptic plasticity models

9ML export

9ML import

NEST-SLI backend

Multi-compartmental models

# Standardized cell, synapse and plasticity models

```
cell_type = sim.IF_cond_exp
p = sim.Population(size, cell_type, parameters)
```

- For a given cell model:
  - identify (NEST, PCSIM) or build (NEURON, Brian) a model with the desired behaviour
  - map model name and parameter names and units
  - (test that each simulator gives the same results)

http://www.flickr.com/photos/mvjantzen/1546831097/

# Standardized cell, synapse and plasticity models

```
cell_type = sim.IF_cond_exp
p = sim.Population(size, cell_type, parameters)
```

- For a given cell model:
  - identify (NEST, PCSIM) or build (NEURON, Brian) a model with the desired behaviour
  - map model name and parameter names and units
  - (test that each simulator gives the same results)


http://www.flickr.com/photos/mvjantzen/1546831097/

# Standardized cell, synapse and plasticity models

Example: Leaky integrate-and-fire model with fixed firing threshold, and current-based, alpha-function synapses.

| PyNN | | NEURON | | NEST | | PCSIM | |
|---|---|---|---|---|---|---|---|
| `IF_curr_alpha` | | `StandardIF` (type="current", shape="alpha") | | `iaf_psc_alpha` | | `LIFCurrAlphaNeuron` | |
| v_rest | mV | v_rest | mV | E_L | mV | Vresting | V |
| v_reset | mV | v_reset | mV | V_reset | mV | Vreset | V |
| cm | nF | CM | nF | C_m | pF | Cm | F |
| tau_m | ms | tau_m | ms | tau_m | ms | taum | s |
| tau_refrac | ms | t_refrac | ms | t_ref | ms | Trefract | s |
| tau_syn_E | ms | tau_syn_E | ms | tau_syn_ex | ms | TauSynExc | s |
| tau_syn_I | ms | tau_syn_IU | ms | tau_syn_in | ms | TauSynInh | s |
| v_thresh | mV | v_thresh | mV | V_th | mV | Vthresh | V |
| i_offset | nA | i_offset | nA | I_e | pA | Iinject | A |

# More flexible cell, synapse and plasticity models


...describe cell/synapse model in simulator-independent way then generate code for each simulator

# More flexible cell, synapse and plasticity models


...describe cell/synapse model in simulator-independent way then generate code for each simulator

Model description languages (XML-based):

# More flexible cell, synapse and plasticity models


...describe cell/synapse model in simulator-independent way then generate code for each simulator

Model description languages (XML-based):

NeuroML v1 - limited to compartmental models with Hodgkin-Huxley-type ion channels.
Implicit mathematics.

NeuroML v2 - removes limitations of NeuroML v1.
Explicit mathematics. In development by NeuroML community.

NineML - removes limitations of NeuroML v1.
Explicit mathematics. In development by INCF Taskforce for Multiscale Modeling.

# NineML: scope and motivation

http://software.incf.org/software/nineml

**Motivation**

- as for PyNN and for NeuroML
  - develop a standard way to describe multi-scale neuronal models
  - enhance model sharing and reproducibility

**Scope**

- describe networks of spiking point neurons (I&F, Izhikevich, etc.) with activity-dependent plasticity
  - not well served by NeuroML v1 or other declarative modelling languages available at the time

Program on Multi-Scale Modeling

# NineML abstraction layer: Diagram module

- For describing “network element” models: neurons, synapses (including synaptic plasticity rules)
- Concept of a block diagram consisting of **Regimes** joined by **Transitions**.

## Regime

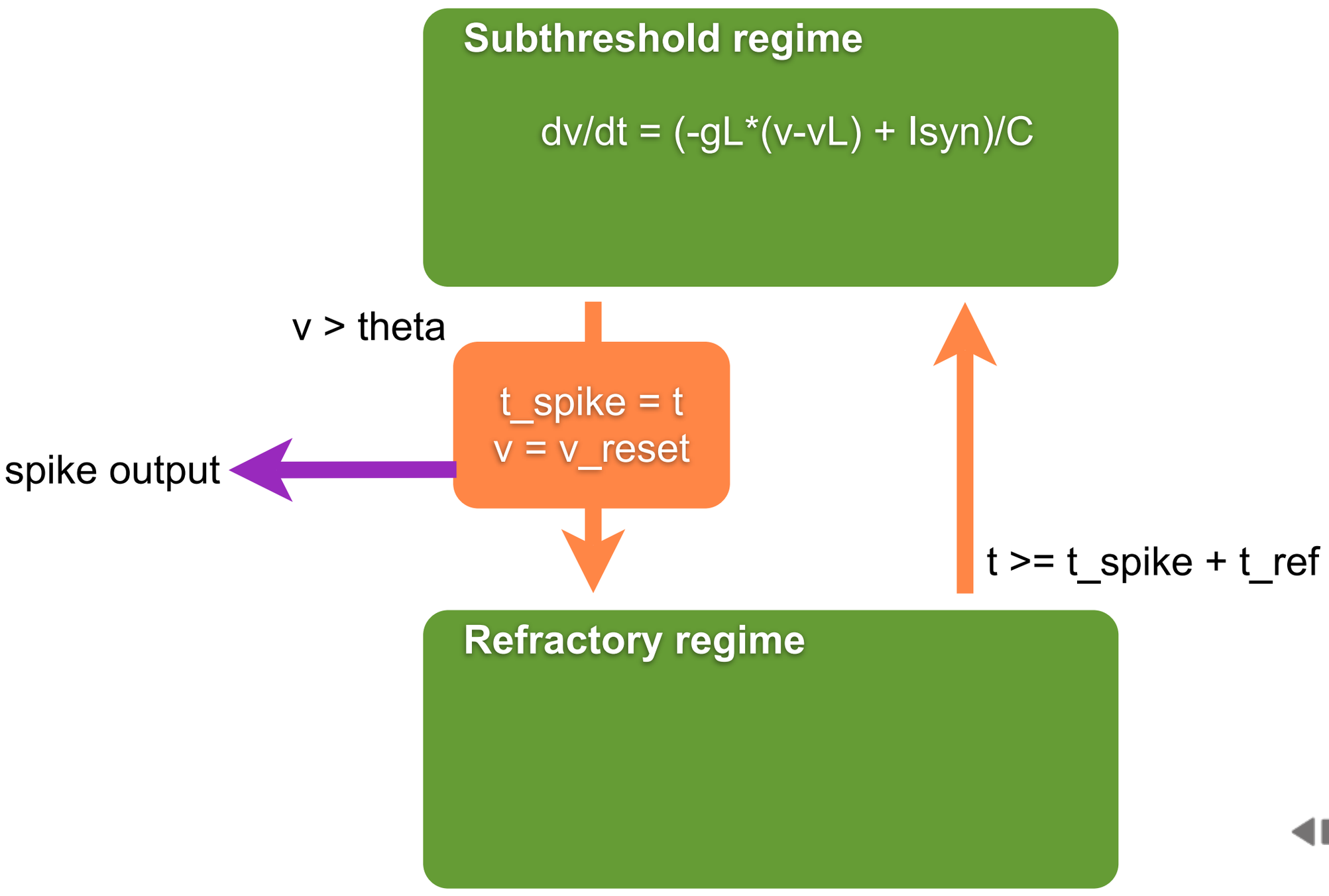
- finite temporal extent
- defines how state variables evolve in time between transitions

## Transition

- vanishing temporal extent
- joins two regimes
- may contain discontinuous changes to state variables
- may be triggered by a condition or by external events

incf
Program on Multi-Scale Modeling

# NineML: Integrate-and-fire example



# NineML: Integrate-and-fire example - Python

```
from nineml.abstraction_layer import *

subthreshold_regime = Regime(
    "dV/dt = (-gL*(V-vL) + Isyn)/C",
    transitions = On("V> theta",
                     do=["t_spike = t", "V = V_reset", SpikeOutputEvent],
                     to="refractory_regime"),
    name="subthreshold_regime")

refractory_regime = Regime(
    transitions = On("t >= t_spike + t_ref",
                     to=subthreshold_regime),
    name="refractory_regime")

ports = [SendPort("V"), ReducePort("Isyn", op="+")]

c = Component("LeakyIAF",
              regimes=[subthreshold_regime, refractory_regime])
```

incf Program on Multi-Scale Modeling

# NineML: Integrate-and-fire example - XML

```
<?xml version='1.0' encoding='UTF-8'?>
<nineml xmlns="http://nineml.org/9ML/0.1">
  <component name="LeakyIAF">
    <parameter name="C"/>
    <parameter name="Isyn"/>
    <parameter name="vL"/>
    <parameter name="V_reset"/>
    <parameter name="theta"/>
    <parameter name="gL"/>
    <parameter name="t_ref"/>
    <analog-port symbol="t_spike" mode="send"/>
    <analog-port symbol="t" mode="send"/>
    <analog-port symbol="V" mode="send"/>
    <regime name="subthreshold_regime">
      <ode independent_variable="t" name="ODE0" dependent_variable="V">
        <math-inline>(-gL*(V-vL) + Isyn)/C</math-inline>
      </ode>
    </regime>
    <regime name="refractory_regime"/>
    <transition to="refractory_regime" from="subthreshold_regime" name="Transition0" condition="V&gt; theta">
      <assignment to="t_spike" name="Assignment0">
        <math-inline>t</math-inline>
      </assignment>
      <assignment to="V" name="Assignment1">
        <math-inline>V_reset</math-inline>
      </assignment>
      <event-port symbol="spike_output" mode="send"/>
    </transition>
    <transition to="subthreshold_regime" from="refractory_regime" name="Transition1" condition="t &gt;= t_spike + t_ref"/>
  </component>
</nineml>
```

incf
Program on Multi-Scale Modeling

# NineML cell, synapse and plasticity models

```
cell_type = nineml_cell_type("iaf_3coba",
                             "iaf.xml",
                             AMPA="coba_syn.xml",
                             NMDA="nmda_syn.xml",
                             GABAA="coba_syn.xml")

parameters = {
    'iaf.cm': 1.0, 'iaf.gl': 50.0, 'iaf.taurefrac': 5.0,
    'iaf.vrest': -65.0, 'iaf.vreset': -65.0, 'iaf.vthresh': -50.0,
    'AMPA.tau': 2.0, 'GABAa.tau': 5.0, 'AMPA.vrev': 0.0, 'GABAa.vrev': -70.0,
    'nmda.taur': 3.0, 'nmda.taud': 40.0, 'nmda.gmax': 1.2, 'nmda.E': 0.0,
    'nmda.gamma': 0.062, 'nmda.mgconc': 1.2, 'nmda.beta': 3.57
}

p = sim.Population(size, cell_type, parameters)
```

- allows multi-simulator use of arbitrary cell models, no more fixed menu
- based on Python lib9ML

# NineML cell, synapse and plasticity models

```
cell_type = nineml_cell_type("iaf_3coba",
                             "iaf.xml",
                             AMPA="coba_syn.xml",
                             NMDA="nmda_syn.xml",
                             GABAA="coba_syn.xml")

parameters = {
    'iaf.cm': 1.0, 'iaf.gl': 50.0, 'iaf.taurefrac': 5.0,
    'iaf.vrest': -65.0, 'iaf.vreset': -65.0, 'iaf.vthresh': -50.0,
    'AMPA.tau': 2.0, 'GABAa.tau': 5.0, 'AMPA.vrev': 0.0, 'GABAa.vrev': -70.0,
    'nmda.taur': 3.0, 'nmda.taud': 40.0, 'nmda.gmax': 1.2, 'nmda.E': 0.0,
    'nmda.gamma': 0.062, 'nmda.mgconc': 1.2, 'nmda.beta': 3.57
}

p = sim.Population(size, cell_type, parameters)
```

- allows multi-simulator use of arbitrary cell models, no more fixed menu
- based on Python lib9ML

http://www.flickr.com/photos/mcbarnicle/3286664183/

# Behind the scenes...

```
cell_type = nineml_cell_type("iaf_3coba",
                             "iaf.xml",
                             AMPA="coba_syn.xml",
                             NMDA="nmda_syn.xml",
                             GABAA="coba_syn.xml")
```

- Parse XML file
- Generate C++ (NEST) or NMODL (NEURON) file based on template
- Compile (gcc, nrnivmodl) and import new module/mechanism
- Create Python CellType object

2012 March

June 0.8

API cleaning

Functions as parameters

Neo output

9ML cell and synapse models

MOOSE backend

NeMo backend

NeuroML export

Developers' guide

Sphinx docs

Split cell and synapse models

MUSIC integration

9ML synaptic plasticity models

9ML export

9ML import

NEST-SLI backend

December 0.9 2013

Multi-compartmental models

# Multi-simulations in PyNN

http://software.incf.org/software/MUSIC

```
from pyNN import music

vizapp = music.Config("vizapp", 1, "/path/to/vizapp", "args")
music.setup(music.Config("neuron", 10), music.Config("nest", 20),
            vizapp)

sim1, sim2 = music.get_simulators("neuron", "nest")
sim1.setup(timestep=0.025)
sim2.setup(timestep=0.1)
pE = sim1.Population((100,100), sim.IF_cond_exp, label="excitatory")
pI = sim2.Population((50,50), sim.IF_cond_exp, label="inhibitory")

def connector(sim):
    DDPC = getattr(sim, "DistanceDependentProbabilityConnector")
    return DDPC("exp(-d**2/400.0)", weights=0.05, delays="0.5+0.01d")
e2e = sim1.Projection(pE, pE, connector(sim1), target="excitatory")
e2i = music.Projection(pE, pI, connector(music), target="excitatory")
i2i = sim2.Projection(pI, pI, connector(sim2), target="inhibitory")

output = music.Port(pE, "spikes", vizapp, "pE_spikes_viz")

music.run(1000.0)
```

2012 March

June 0.8

API cleaning

Functions as parameters

Neo output

9ML cell and synapse models

MOOSE backend

NeMo backend

NeuroML export

Developers' guide

Sphinx docs

Split cell and synapse models

MUSIC integration

9ML synaptic plasticity models

9ML export

9ML import

NEST-SLI backend

2013 December 0.9

Multi-compartmental models

# More sophisticated data-handling

## 0.7

```
p.record()
p.record_v()
p._record('foo')
```

```
data_spikes = p.getSpikes()
data_v = p.get_v()
data_foo = p.recorders['foo'].get()
```

## 0.8

```
p.record(['spikes', 'v', 'foo'])
```

```
data = p.get_data()
```

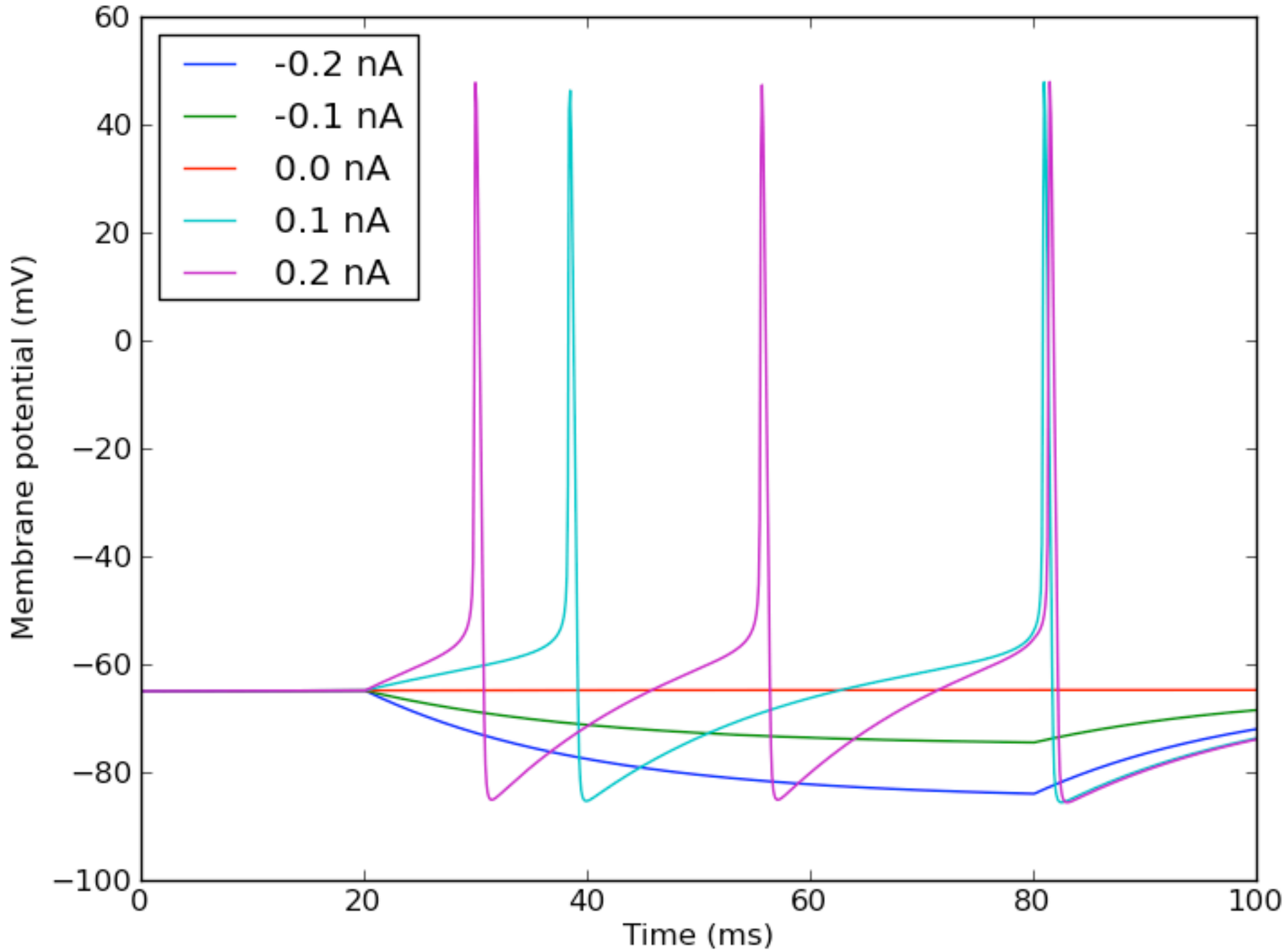
# More sophisticated data-handling

```
cell = sim.Population(1, sim.HH_cond_exp)
step_current = sim.DCSource(start=20.0, stop=80.0)
step_current.inject_into(cell)

cell.record('v')

for amp in (-0.2, -0.1, 0.0, 0.1, 0.2):
    step_current.amplitude = amp
    sim.run(100.0)
    sim.reset(annotations={"amplitude": amp*nA})

data = cell.get_data()

sim.end()

for segment in data.segments:
    vm = segment.analogsignalarrays[0]
    plt.plot(vm.times, vm,
             label=str(segment.annotations["amplitude"]))
plt.legend(loc="upper left")
plt.xlabel("Time (%s)" % vm.times.units._dimensionality)
plt.ylabel("Membrane potential (%s)" % vm.units._dimensionality)
```

http://neuralensemble.org/trac/PyNN/wiki/ChangesToDataHandling

# More sophisticated data-handling

```
cell = sim.Population(1, sim.HH_cond_exp)
step_curre
step_curre

cell.reco

for amp i
    step_
    sim.ru
    sim.re

data = ce

sim.end()

for segmen
    vm = s
    plt.pl

plt.legend
plt.xlabel
plt.ylabel
```



# http://neuralensemble.org/PyNN

PyNN

Search

Login | Settings | Help/Guide | About Trac

Wiki | Timeline | Roadmap | Browse Source | View Tickets | New Ticket | Search

Available Reports | Custom Query

## {3} Active Tickets by Milestone (44 matches)

This report shows how to color results by priority, while grouping results by milestone.

Last modification time and description are included as hidden fields for useful RSS export.

### 0.8.0 Release

| Ticket | Summary | Component | Version | Type | Owner | Created | Reporter |
|---|---|---|---|---|---|---|---|
| #155 | When using `Projection.set/getWeights()` with arrays, the case of multiple connections between source and target is not consistently handled or tested | all | trunk | defect | | 02/03/10 | apdavison |
| #172 | TsodyksMarkram Synapse - uses nest tsodyks_synapse but not tau_psc | nest | trunk | defect | apdavison * | 10/24/10 | emuller |
| #192 | Recording issue with multiple run() calls in NEST and to_file=True | Documentation | trunk | defect | | 05/04/11 | pierre |
| #62 | Draw dynamic synapses parameters from RNG Distributions | common | trunk | enhancement | pierre | 04/03/08 | pierre |
| #113 | Add `Projection.record_weights()` or something similar | all | trunk | enhancement | apdavison * | 08/27/08 | apdavison |
| #137 | Reimplement connectors in terms of arrays | common | trunk | enhancement | apdavison | 06/05/09 | apdavison |
| #138 | All issues of which connections are local should be dealt with within ConnectionManager, not in the Connectors | common | trunk | enhancement | apdavison | 06/05/09 | apdavison |
| #153 | A way to reset the list of things to record | common | trunk | enhancement | apdavison * | 01/26/10 | apdavison |
| #154 | Implement `Population.record_gsyn()` method and `record_gsyn()` function in the `pcsim` module | pcsim | trunk | enhancement | apdavison | 02/02/10 | apdavison |
| #161 | Rethink the connectors as convergent_connect() and problems with | all | trunk | enhancement | None | 03/26/10 | pierre |

http://neuralensemble.org/PyNN

FET (FACETS, BrainScaleS)
CNRS



Pierre Yger
Eilif Muller
Daniel Brüderle
Jochen Eppler
Jens Kremkow
Dejan Pecevski
Mike Hull
Michael Schmuker

@apdavison
http://andrewdavison.info

PyNN