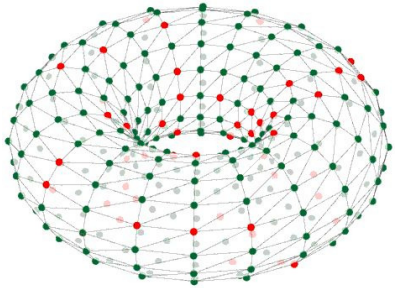
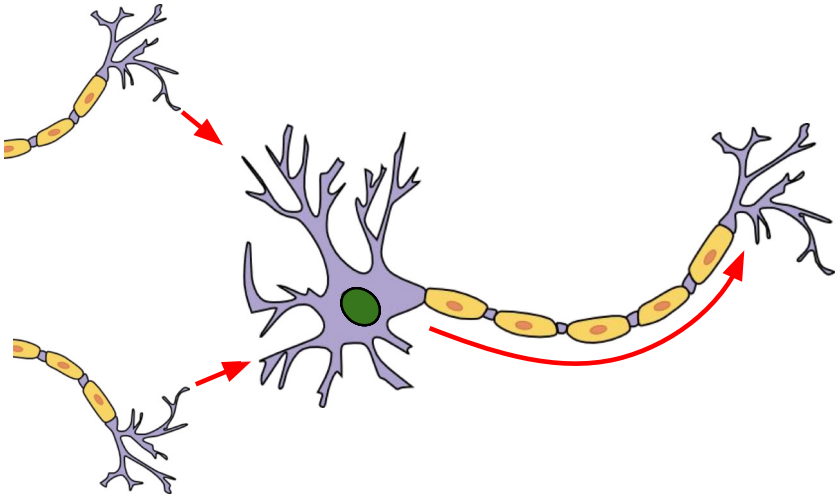
Running PyNN Simulations on SpiNNaker



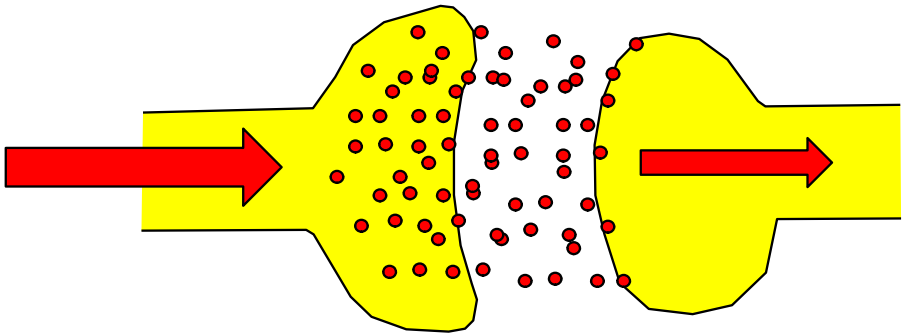
Andrew Rowley, Alan Stokes

SpiNNaker Workshop, September 2016

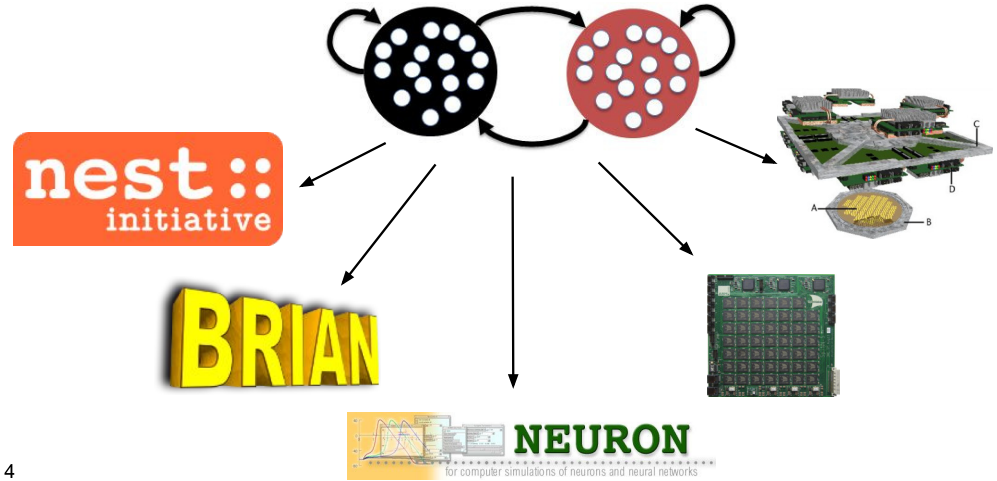
Spiking Neural Networks



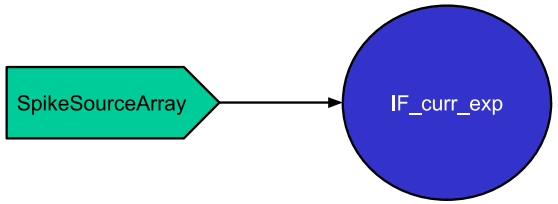
Spiking Neural Networks



What is PyNN?



A Simple PyNN Network



A Simple PyNN Network

```
import pyNN.spiNNaker as p
```


A Simple PyNN Network

```
import pyNN.spiNNaker as p
p.setup(timestep=1.0)
```


A Simple PyNN Network

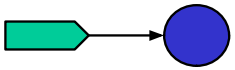
```
import pyNN.spiNNaker as p
p.setup(timestep=1.0)
pop_1 = p.Population(1, p.IF_curr_exp, {}, label="pop_1")
```

A Simple PyNN Network

```
import pyNN.spiNNaker as p
p.setup(timestep=1.0)
pop_1 = p.Population(1, p.IF_curr_exp, {}, label="pop_1")
input = p.Population(1, p.SpikeSourceArray,
                     {'spike_times': [0]}, label="input")
```

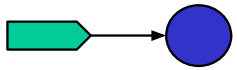
A Simple PyNN Network

```
import pyNN.spiNNaker as p
p.setup(timestep=1.0)
pop_1 = p.Population(1, p.IF_curr_exp, {}, label="pop_1")
input = p.Population(1, p.SpikeSourceArray,
                     {'spike_times': [0]}, label="input")
input_proj = p.Projection(input, pop_1, p.OneToOneConnector(
    weights=5.0, delays=1))
```



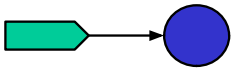
A Simple PyNN Network

```
import pyNN.spiNNaker as p
p.setup(timestep=1.0)
pop_1 = p.Population(1, p.IF_curr_exp, {}, label="pop_1")
input = p.Population(1, p.SpikeSourceArray,
                     {'spike_times': [0]}, label="input")
input_proj = p.Projection(input, pop_1, p.OneToOneConnector(
    weights=5.0, delays=1))
pop_1.record()
pop_1.record_v()
```



A Simple PyNN Network

```
import pyNN.spiNNaker as p
p.setup(timestep=1.0)
pop_1 = p.Population(1, p.IF_curr_exp, {}, label="pop_1")
input = p.Population(1, p.SpikeSourceArray,
                     {'spike_times': [0]}, label="input")
input_proj = p.Projection(input, pop_1, p.OneToOneConnector(
    weights=5.0, delays=1))
pop_1.record()
pop_1.record_v()
p.run(10)
```



Edit ~/.spynnaker.cfg

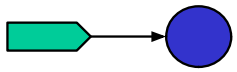
```
[Machine]
#-------
# Information about the target SpiNNaker board or machine:
# machineName:     The name or IP address of the target board
# version:         Version of the Spinnaker Hardware Board (1-5)
# machineTimeStep: Internal time step in simulations in usecs.
# timeScaleFactor: Change this to slow down the simulation time
#                      relative to real time.
#-------
machineName     = None
version         = None
#machineTimeStep = 1000
#timeScaleFactor = 1
```


Edit ~/.spynnaker.cfg

```
[Machine]
#-------
# Information about the target SpiNNaker board or machine:
# machineName:     The name or IP address of the target board
# version:         Version of the Spinnaker Hardware Board (1-5)
# machineTimeStep: Internal time step in simulations in usecs.
# timeScaleFactor: Change this to slow down the simulation time
#                      relative to real time.
#-------
machineName     = 192.168.240.253
version         = 3
#machineTimeStep = 1000
#timeScaleFactor = 1
```

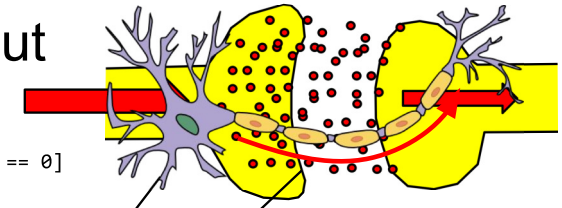

A Simple PyNN Network

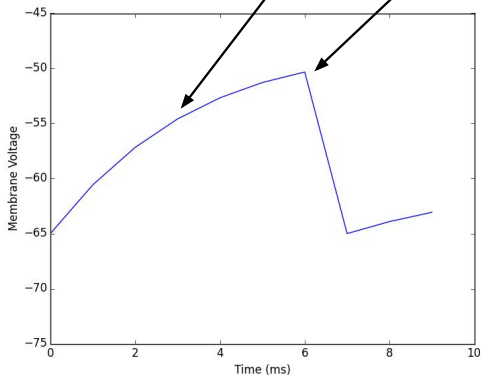
```
import pyNN.spiNNaker as p
p.setup(timestep=1.0)
pop_1 = p.Population(1, p.IF_curr_exp, {}, label="pop_1")
input = p.Population(1, p.SpikeSourceArray,
                     {'spike_times': [0]}, label="input")
input_proj = p.Projection(input, pop_1, p.OneToOneConnector(
    weights=5.0, delays=1))
pop_1.record()
pop_1.record_v()
p.run(10)
spikes = pop_1.getSpikes()
v = pop_1.get_v()
```



Plotting Output

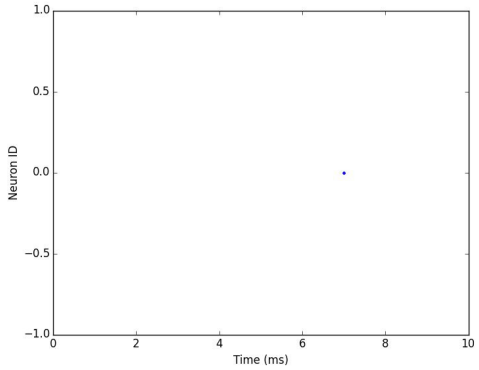
```
import pylab
time = [i[1] for i in v if i[0] == 0]
membrane_voltage = [i[2] for i in v if i[0] == 0]
pylab.plot(time, membrane_voltage)
pylab.xlabel("Time (ms)")
pylab.ylabel("Membrane Voltage")
pylab.axis([0, 10, -75, -45])
pylab.show()
```



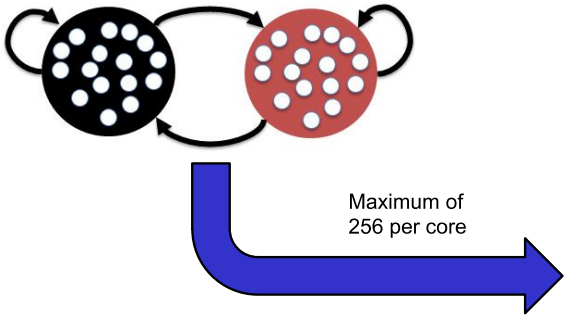


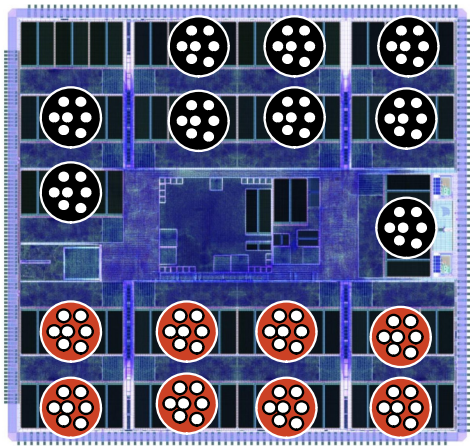
Plotting Output

```
import pylab
spike_time = [i[1] for i in spikes]
spike_id = [i[0] for i in spikes]
pylab.plot(spike_time, spike_id, ".")
pylab.xlabel("Time (ms)")
pylab.ylabel("Neuron ID")
pylab.axis([0, 10, -1, 1])
pylab.show()
```

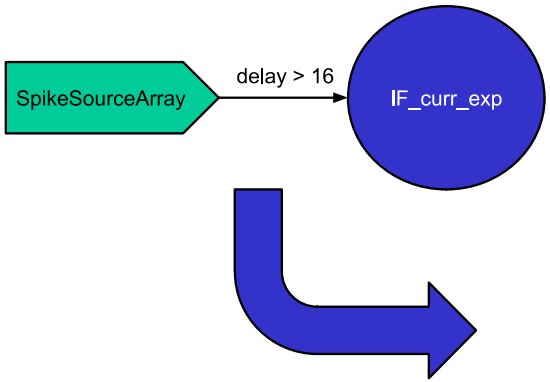


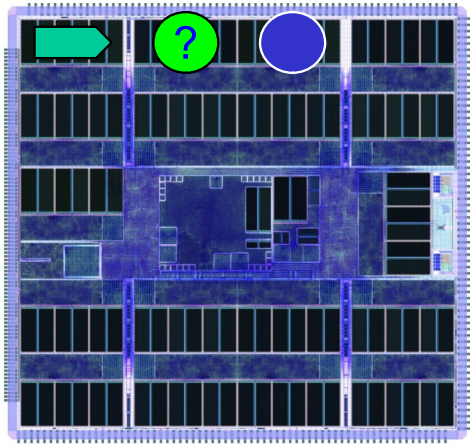
Limitations of PyNN on SpiNNaker:
Neurons Per Core





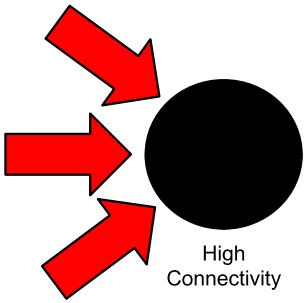
Limitations of PyNN on SpiNNaker:
Number of cores available

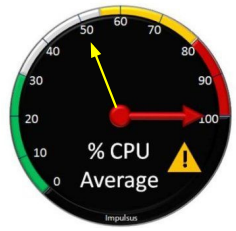




SpiNNaker-Specific PyNN

```
import pyNN.spiNNaker as p
p.setup(timestep=1.0)
p.set_number_of_neurons_per_core(p.IF_curr_exp, 100)
pop_1 = p.Population(1, p.IF_curr_exp, {}, label="pop_1")
```

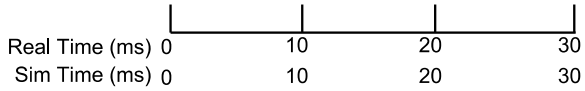




Plasticity

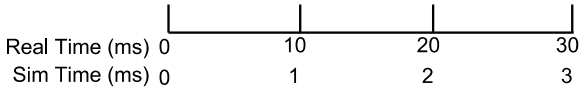
Configuration with spynnaker.cfg

```
[Machine]
machineName      = None
version          = None
timeScaleFactor  = 1
```

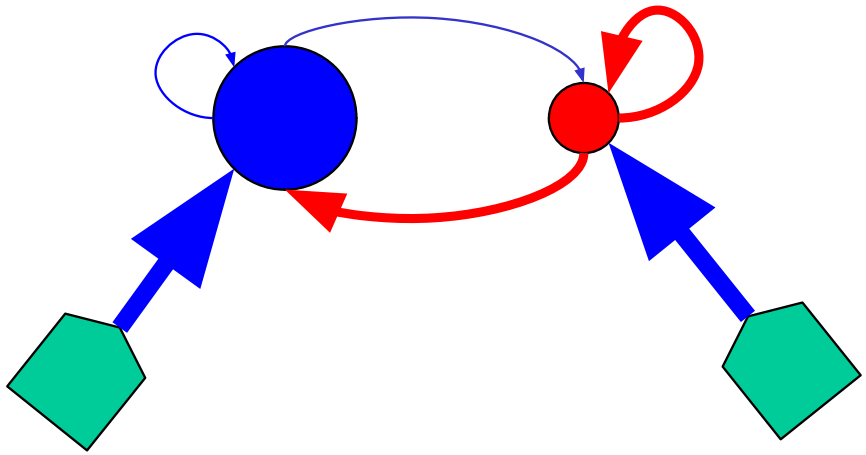


Configuration with spynnaker.cfg

```
[Machine]
machineName      = None
version          = None
timeScaleFactor  = 10
```



Balanced Random Network



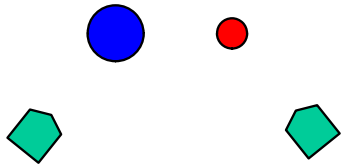
Balanced Random Network

```
import pyNN.spiNNaker as p
import pylab
from pyNN.random import RandomDistribution

p.setup(timestep=0.1)
n_neurons = 1000
n_exc = int(round(n_neurons * 0.8))
n_inh = int(round(n_neurons * 0.2))
```

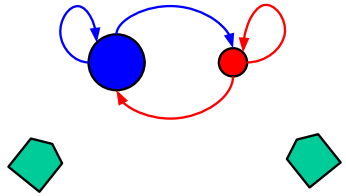
Balanced Random Network

```
pop_exc = p.Population(n_exc, p.IF_curr_exp, {},
                       label="Excitatory")
pop_inh = p.Population(n_inh, p.IF_curr_exp, {},
                       label="Inhibitory")
stim_exc = p.Population(n_exc, p.SpikeSourcePoisson,
                        {"rate": 10.0}, label="Stim_Exc")
stim_inh = p.Population(n_inh, p.SpikeSourcePoisson,
                        {"rate": 10.0}, label="Stim_Inh")
```



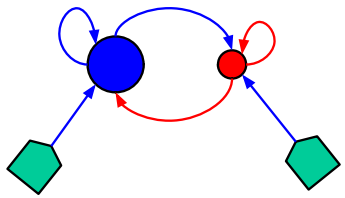
Balanced Random Network

```
conn_exc = p.FixedProbabilityConnector(0.1, weights=0.2,
                                       delays=2.0)
conn_inh = p.FixedProbabilityConnector(0.1, weights=-1.0,
                                       delays=2.0)
p.Projection(pop_exc, pop_exc, conn_exc, target="excitatory")
p.Projection(pop_exc, pop_inh, conn_exc, target="excitatory")
p.Projection(pop_inh, pop_inh, conn_inh, target="inhibitory")
p.Projection(pop_inh, pop_exc, conn_inh, target="inhibitory")
```



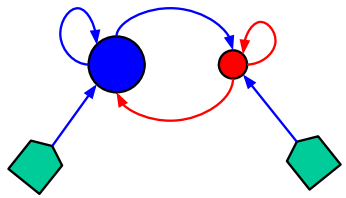
Balanced Random Network

```
delays_stim = RandomDistribution("uniform", [1.0, 1.6])
conn_stim = p.OneToOneConnector(weights=2.0,
                                delays=delays_stim)
p.Projection(stim_exc, pop_exc, conn_stim, target="excitatory")
p.Projection(stim_inh, pop_inh, conn_stim, target="excitatory")
```



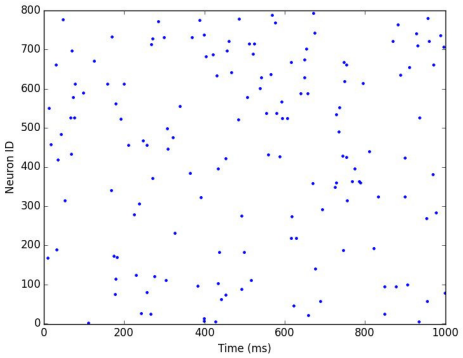
Balanced Random Network

```
pop_exc.initialize("v", RandomDistribution("uniform",
                                           [-65.0, -55.0]))
pop_inh.initialize("v", RandomDistribution("uniform",
                                           [-65.0, -55.0]))
pop_exc.record()
p.run(1000)
```

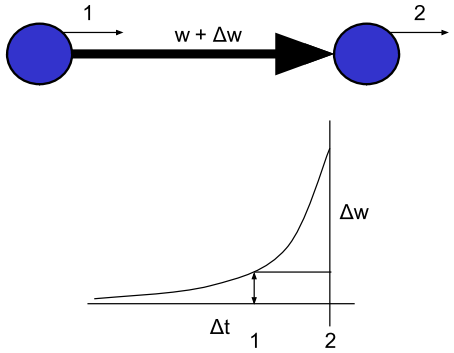


Balanced Random Network

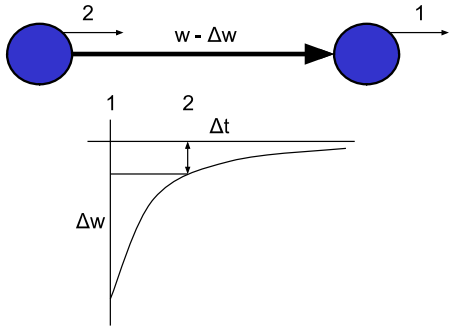
```
spikes = pop_exc.getSpikes()
pylab.plot([i[1] for i in spikes], [i[0] for i in spikes], ".")
pylab.xlabel("Time (ms)")
pylab.ylabel("Neuron ID")
pylab.axis([0, 1000, -1, n_exc + 1])
pylab.show()
```



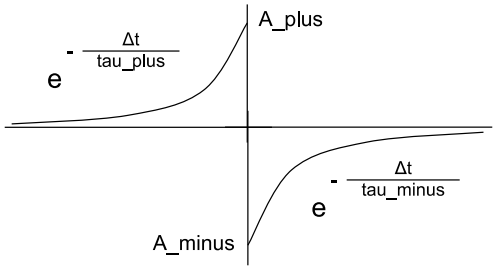
Spike Time Dependent Plasticity: Potentiation

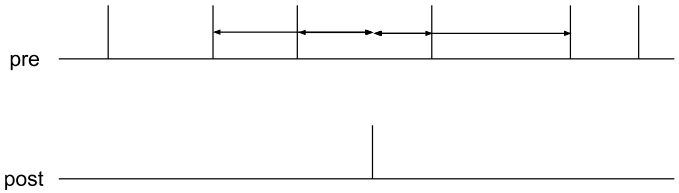


Spike Time Dependent Plasticity: Depression

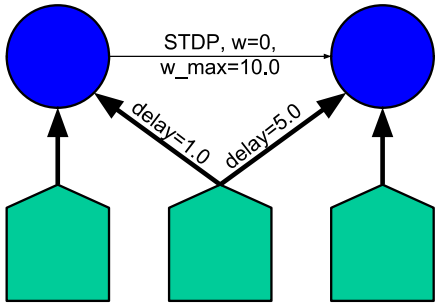


STDP Rules





STDP in PyNN



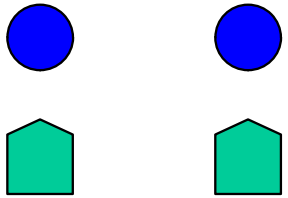
STDP in PyNN

```
import pyNN.spiNNaker as p
import pylab

p.setup(timestep=1.0)
n_neurons = 100

pre_pop    = p.Population(n_neurons, p.IF_curr_exp, {}, label="Pre")
post_pop   = p.Population(n_neurons, p.IF_curr_exp, {}, label="Post")
pre_noise  = p.Population(
    n_neurons, p.SpikeSourcePoisson, {"rate": 10.0}, label="Noise_Pre")
post_noise = p.Population(
    n_neurons, p.SpikeSourcePoisson, {"rate": 10.0}, label="Noise_Post")

pre_pop.record()
post_pop.record()
```

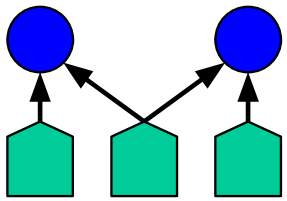


STDP in PyNN

```
training = p.Population(
    n_neurons, p.SpikeSourcePoisson,
    {"rate": 10.0, "start": 2000.0, "duration": 1000.0},
    label="Training")

p.Projection(pre_noise,  pre_pop,  p.OneToOneConnector(weights=2.0))
p.Projection(post_noise, post_pop, p.OneToOneConnector(weights=2.0))

p.Projection(training, pre_pop,  p.OneToOneConnector(weights=5.0, delays=1.0))
p.Projection(training, post_pop, p.OneToOneConnector(weights=5.0, delays=10.0))
```

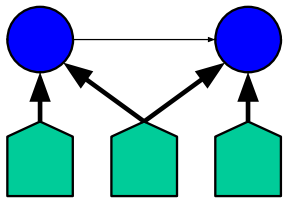


STDP in PyNN

```
timing_rule = p.SpikePairRule(tau_plus=20.0, tau_minus=20.0)
weight_rule = p.AdditiveWeightDependence(
    w_max=5.0, w_min=0.0, A_plus=0.5, A_minus=0.5)

stdp_model = p.STDPMechanism(
    timing_dependence=timing_rule, weight_dependence=weight_rule)

stdp_projection = p.Projection(
    pre_pop, post_pop, p.OneToOneConnector(weights=0.0, delays=5.0),
    synapse_dynamics=p.SynapseDynamics(slow=stdp_model))
```



STDP in PyNN

```
p.run(5000)

pre_spikes = pre_pop.getSpikes()
post_spikes = post_pop.getSpikes()

print stdp_projection.getWeights()
```

```
[ 4.83886719  5.          4.27246094  5.          3.80957031  4.87060547
  5.          5.          5.          5.          5.
  4.25048828  5.          5.          4.57763672  5.          5.          5.
  5.          5.          5.          5.          5.          5.          5.
  5.          3.76953125  5.          5.          5.          5.          5.
  5.          5.          5.          5.          5.          5.          5.
  4.81591797  5.          5.          5.          5.          5.
  2.73339844  5.          5.          5.          5.          5.          5.
  5.          5.          5.          5.          5.          5.          5.
  5.          5.          5.          5.          4.98046875  5.          5.
  5.          5.          5.          5.          4.23388672  5.          5.
  5.          5.          5.          5.          4.69433594  5.          5.
  5.          5.          5.          5.          5.          5.          5.
  3.85400391  5.          5.          5.          4.07617188  5.          5.
  5.          5.          5.          5.          5.        ]
```


STDP in PyNN

```
pylab.figure()
pylab.xlim((0, 5000))
pylab.plot([i[1] for i in pre_spikes],  [i[0] for i in pre_spikes],  "r.")
pylab.plot([i[1] for i in post_spikes], [i[0] for i in post_spikes], "b.")
pylab.xlabel('Time/ms')
pylab.ylabel('spikes')
pylab.show()
```