SpiNNaker Tutorial
Accessing SpiNNaker
Accessing SpiNNaker via Jupyter

https://spinn-20.cs.man.ac.uk/

User Guide

Sign in with EBRAINS Credentials

Sign in with HBP Credentials
Accessing SpiNNaker via Jupyter

User Guide

Jupyter Notebook Interface  JupyterLab Interface
Accessing SpiNNaker via Jupyter
Jupyter Folders
EBRAINS Drive
Running sPyNNaker
Running sPyNNaker
Changing Kernel
Close Notebooks
Jupyter Autocomplete
Jupyter Function Help
Install Libraries

```bash
$ pip install gym
```
Terminal Access
Terminal Access
Start the NRP...
Connect to the NRP

https://spinn-20.cs.man.ac.uk/user/<username>/proxy/9000/#/esv-private
Logout!
Spiking Neural Networks
Spiking Neural Networks
Spiking Neural Networks
PyNN
What is PyNN?
import pyNN.spiNNaker as p

p.setup(timestep=1.0)

p.setup(timestep=0.1)
PyNN - Populations

```python
pop_1 = p.Population(4, p.IF_curr_exp(), label="Fred")
pop_2 = p.Population(2, p.IF_curr_exp(), label="Bob")
```
PyNN Populations - Models

\[ p.\text{IF\_curr\_exp(...)} \]

\[ p.\text{IF\_cond\_exp(...)} \]

\[ p.\text{Izhikevich(...)} \]
PyNN Populations - Models

```python
p.SpikeSourceArray(
    spike_times=[...])
```

```python
p.SpikeSourcePoisson(
    rate=...)  
```
PyNN Populations - Recording

```python
pop_1.record(["V", "spikes"])
```
pop_1.initialize(v=-60)
PyNN - Random Parameters

```
pop_1.initialize(v=p.RandomDistribution("uniform", low=-65.0, high=-55.0))
```
proj = p.Projection(
    pop_1, pop_2,
    p.OneToOneConnector(),
    p.StaticSynapse(
        weight=1.0, delay=2.0))
PyNN - Projections

```python
proj = p.Projection(
    pop_1, pop_2,
    p.OneToOneConnector(),
    p.StaticSynapse(weight=1.0),
    receptor_type="excitatory")
```
PyNN - Projections

```python
proj = p.Projection(
    pop_1, pop_2,
    p.OneToOneConnector(),
    p.StaticSynapse(weight=1.0),
    receptor_type="inhibitory")
```
PyNN Projections - Connectors

```python
p.OneToOneConnector()
```
p.AllToAllConnector()
p.FixedProbabilityConnector(p=0.1)
p.FixedProbabilityConnector(p=0.5)
PyNN Projections - Connectors

```
p.FixedTotalNumberConnector(5)
```
PyNN Projections - Connectors

```python
p.FromListConnector([(1, 1), (2, 1), (3, 2), (4, 2)])
```
p.StaticSynapse(weight=1.0, delay=2.0)
p.StaticSynapse(weight=5.0, delay=2.0)
PyNN - Static Synapse Types

```python
p.StaticSynapse(weight=5.0, delay=3.0)
```

(timestep <= delay <= 144 * timestep)
weight_dist = p.RandomDistribution("normal", mu=0.5, sigma=0.1)
p.StaticSynapse(
    weight=weight_dist, delay=3.0)
weight_dist = p.RandomDistribution("normal_clipped",
    mu=0.5, sigma=0.1, low=0.2 high=0.8)
weight_dist = p.RandomDistribution(“normal_clipped_to_boundary”,
mu=0.5, sigma=0.1, low=0.2 high=0.8)
Spike Timing Dependent Plasticity

Pre-synaptic Neuron Sends Spike

Pre-Spike received at post-synapse

\[ \Delta t = \text{post} - \text{pre} \]

Post-Spike received at post-synapse

Post-synaptic Neuron Sends Spike

time
Spike Timing Dependent Plasticity

Pre-synaptic Neuron Sends Spike

Pre-Spike received at post-synapse

$\Delta t = \text{post} - \text{pre}$

Post-Spike received at post-synapse

Post-synaptic Neuron Sends Spike

Time
STDP - Deferred Execution

Pre-synaptic Neuron Sends Spike

Pre-Spike received at post-synapse

Post-Spike received at post-synapse

Post-synaptic Neuron Sends Spike

Time
p.STDPMechanism(
    timing_dependence=?,
    weight_dependence=?,
    weight=0.0, delay=2.0)
sim.SpikePairRule(tau_plus=20.0, tau_minus=20.0, A_plus=0.5, A_minus=0.5)
PyNN - Timing Dependence

```python
sim.SpikePairRule(tau_plus=20.0, tau_minus=20.0,
                 A_plus=0.5, A_minus=0.5)
```
PyNN - Weight Dependence

\[ \Delta w = \Delta a \ (w_{\text{max}} - w_{\text{min}}) \]
PyNN - Weight Dependence

```
sim.MultiplicativeWeightDependence(w_max=5.0, w_min=0.0)
```

\[ \Delta w = \Delta a (w - w_{\text{min}}) \text{ if } \Delta a < 0 \text{ (Depression)} \]
\[ \Delta w = \Delta a (w_{\text{max}} - w) \text{ if } \Delta a > 0 \text{ (Potentiation)} \]
Running on SpiNNaker
PyNN - Run

p.run(100)
p.run(100)
PyNN - Run

```python
p.run(100)
```
PyNN - Run

p.run(100)
PyNN - Change and Run Again

```python
pop_1.set(i_offset=5.0)
p.run(50)
```
PyNN - Get Data

data = pop_1.get_data(["v", "spikes"])
Reading Results
PyNN - Neo Data

Block → Segment

- segments
- spiketrains
- analogsignals

SpikeTrain

AnalogSignal

SpikeTrain

AnalogSignal

SpikeTrain

AnalogSignal

SpikeTrain

AnalogSignal

SpikeTrain

AnalogSignal
PyNN - Neo Data

data = pop_1.get_data(["v", "spikes"])
v = data.segments[0].analogsignals
spikes = data.segments[0].spiketrains
import matplotlib.pyplot as plt

plt.figure()
plt.plot(v[0].times, v[0])
plt.show()
plt.figure()
y = [1 for i in range(len(spikes[0]))]
plt.plot(spikes[0], y, ".")
plt.show()
from pyNN.utility.plotting import Figure, Panel
Figure(
    Panel(*data.segments[0].analogsignals),
    Panel(data.segments[0].spiketrains)
)
PyNN - Get Weights and Delays

```python
synapses = proj.get(
    ['weight', 'delay'], 'list')

array([(0, 5, 0.756, 1.), (0, 6, 0.316, 1.), (0, 7, 0.885, 2.),
       (0, 8, 0.421, 1.), (1, 4, 0.618, 1.), (1, 7, 0.438, 1.),
       (1, 9, 1.607, 1.), (2, 0, 0.129, 1.), (2, 2, 1.055, 1.),
       (2, 3, 1.319, 1.), (2, 9, 0.422, 1.), (3, 1, 0.328, 1.),
       (3, 3, 0.456, 1.), (3, 6, 0.566, 1.), (4, 0, 1.046, 1.),
       (4, 1, 1.199, 1.), (4, 2, 0.831, 1.), (5, 0, 1.643, 1.),
       (5, 2, 1.165, 1.), (5, 3, 0.902, 1.), (5, 5, 1.627, 1.),
       (6, 0, 2.143, 1.), (6, 5, 0.635, 1.), (6, 7, 0.704, 1.),
       (7, 0, 1.914, 1.), (7, 4, 0.289, 1.), (7, 5, 2.058, 1.),
       (7, 6, 0.428, 2.), (7, 7, 0.639, 1.), (7, 9, 0.616, 2.),
       (8, 0, 1.039, 1.), (8, 1, 0.576, 1.), (8, 4, 1.563, 2.),
       (8, 8, 0.995, 1.), (9, 0, 1.686, 1.), (9, 9, 0.631, 2.])]```
```python
synapses = proj.get(  
    "weight", "array")
```

```
array([[    nan,      nan,      nan,      nan,      nan,  0.756,  0.316,  0.885,  0.421,      nan],  
        [    nan,      nan,      nan,      nan,  0.618,      nan,      nan,  0.438,      nan,  1.607],  
        [ 0.129,      nan,  1.055,  1.319,      nan,      nan,      nan,      nan,      nan,  0.422],  
        [    nan,  0.328,      nan,  0.456,      nan,      nan,  0.566,      nan,      nan,      nan],  
        [ 1.046,  1.199,  0.831,      nan,      nan,      nan,      nan,      nan,      nan,      nan],  
        [ 1.643,      nan,  1.165,  0.902,      nan,  1.627,      nan,      nan,      nan,      nan],  
        [ 2.143,      nan,      nan,      nan,  0.635,      nan,  0.704,      nan,      nan,      nan],  
        [ 1.914,      nan,      nan,      nan,  0.289,  2.058,  0.428,  0.639,      nan,  0.616],  
        [ 1.039,  0.576,      nan,      nan,  1.563,      nan,      nan,      nan,  0.995,      nan],  
        [ 1.686,      nan,      nan,      nan,      nan,      nan,      nan,      nan,  0.631]])
```
PyNN: documentation

- Introduction
- Installation
- Quickstart
- Building networks
- Injecting current
- Recording spikes and state variables
- Data handling
- Simulation control
- Model parameters and initial values
- Random numbers
- Backends
- Running parallel simulations
- Units
- Importing from and exporting to other formats
- Examples
- Publications about, relating to or using PyNN
- Contributors and licence
- Release notes
Live Input and Output
Input from Environment: Spikes

```
SpynnakerLiveSpikesConnection(
    send_labels=['injector'])
```

```
send_spikes('injector', [0])
```

```
SpynnakerLiveSpikesConnection(
    send_labels=['injector'])
```

Multicast Key(s) (Spike)
Input from Environment: Rates

```python
add_poisson_live_rate_control
set_rates(label="input")
set_rates("input", [(0, 10)])
SpynnakerPoissonControlConnection(poisson_labels=["input")
Multicast Key(s) and Payload(s)
```
Output to Environment: Spikes

activate_live_output_for

recv

recv("pop", time, neuron_ids)

SpynnakerLiveSpikesConnection(
    receive_labels=["pop"])

Multicast Key(s) (Spike)
Output to Environment: Voltage

`translator` = `EthernetControlPopulation(lif_device_model)`

`translate_control_packet(pkt)`
SpiNNaker Tutorials on Jupyter

https://spinn-20.cs.man.ac.uk/

01. Running PyNN Simulations
02. Live Input And Output

PyNN Documentation:
http://neuralensemble.org/docs/PyNN/

For NRP: New → Terminal, then run cle-nginx, then cle-start

https://spinn-20.cs.man.ac.uk/user/<username>/proxy/9000/#/esv-private