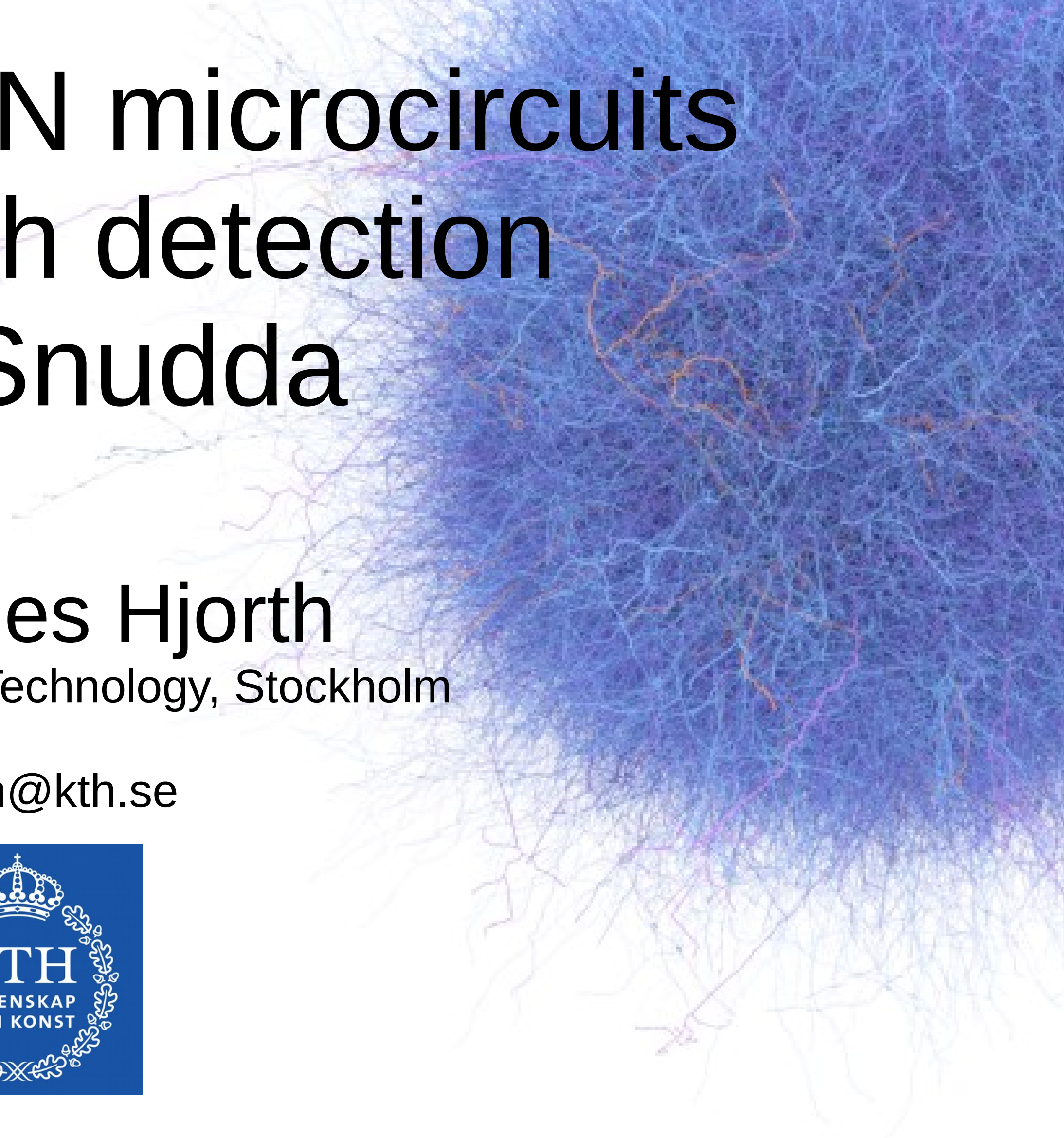
Build NEURON microcircuits using touch detection with Snudda

Johannes Hjorth
Royal Institute of Technology, Stockholm

hjorth@kth.se



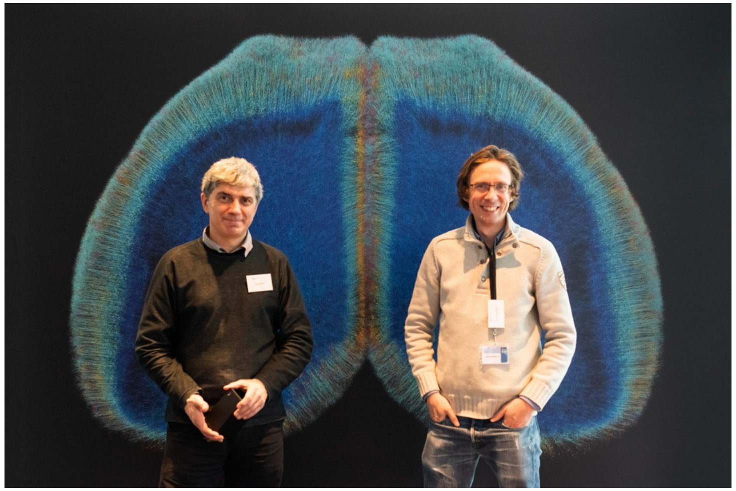
Collaborators and Funding

Jeanette Hellgren Kotaleski
Sten Grillner
Alexander Kozlov
Ilaria Carannante
Johanna Frost Nylén
Robert Lindroos
Wilhelm Tunberg
Kadri Pajo
Bo Bekkouche

Gilad Silberberg

Yvonne Johansson
Shreyas Suryanarayana
Anna Tokarska
Mathijs Dorst

Kai Du


Alexander Kozlov and Johannes Hjorth

Swedish Research Council

KTH VETENSKAP OCH KONST

SciLifeLab

Karolinska Institutet

EBRAINS

SeRC Swedish e-Science Research Centre

dBrain

Hjorth, Kozlov et al, 2020. The microcircuits of striatum in silico. PNAS https://doi.org/10.1073/pnas.2000671117

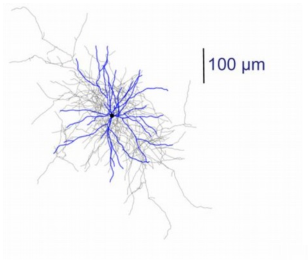
Hjorth, Hellgren Kotaleski, Kozlov 2021.
Predicting Synaptic Connectivity for Large-Scale Microcircuit Simulations Using Snudda
Neuroinformatics https://doi.org/10.1007/s12021-021-09531-w

Frost Nylén, Hjorth, Grillner, Hellgren Kotaleski 2021.
Dopaminergic and Cholinergic Modulation of Large Scale Networks in silico Using Snudda
https://dx.doi.org/10.3389%2Ffncir.2021.748989

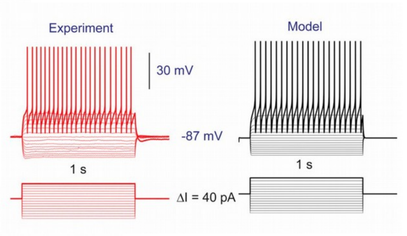
Model building pipeline

Open Source
https://github.com/a1eko/treem
https://github.com/BlueBrain/BluePyOpt
https://github.com/Hjorthmedh/Snudda
https://github.com/neuronsimulator/nrn

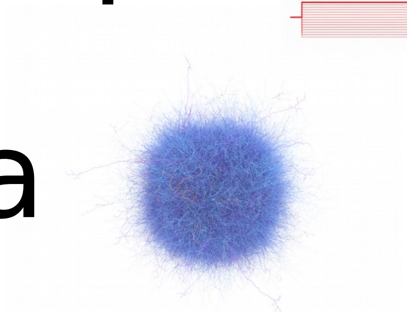
Morphology repair
Treem



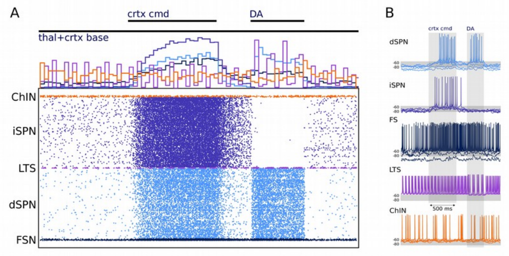
Model optimisation
BluePyOpt



Network creation
Snudda



Simulation
NEURON



What is Snudda?

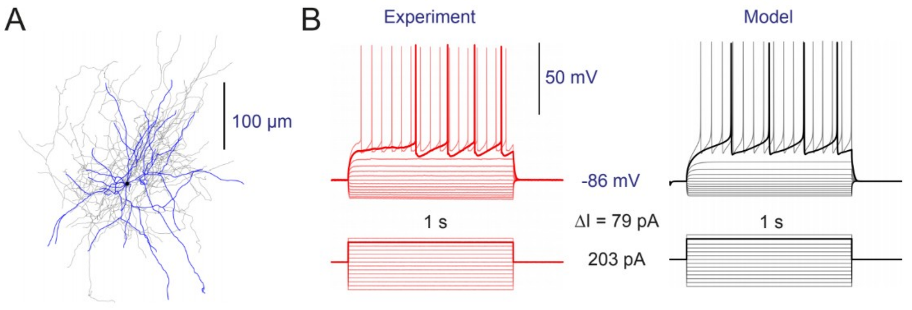
- Voxel-based touch detection
- Rule-based pruning of synapses
- Network analysis
- Simulation of network

- Open source -- https://github.com/hjorthmedh/Snudda
- Runs on Linux, Mac and Windows desktops
  and on super computers

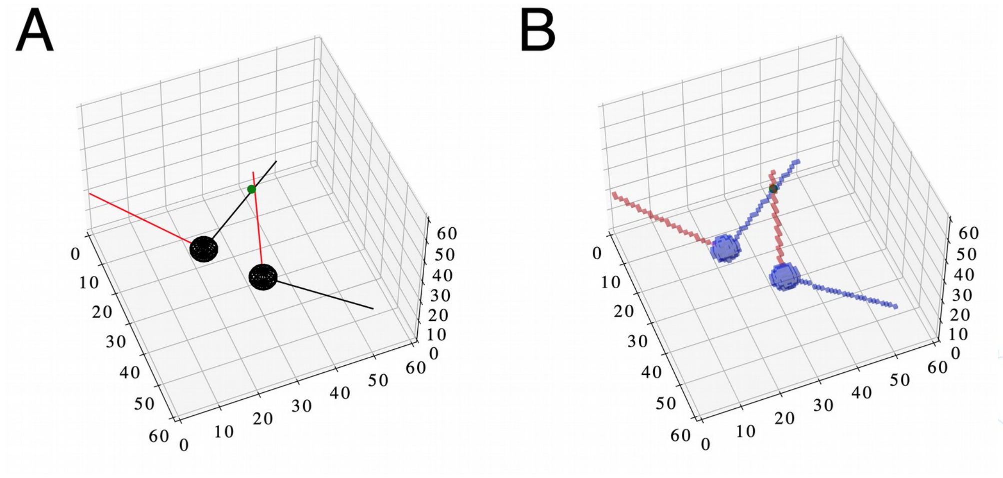
Components needed

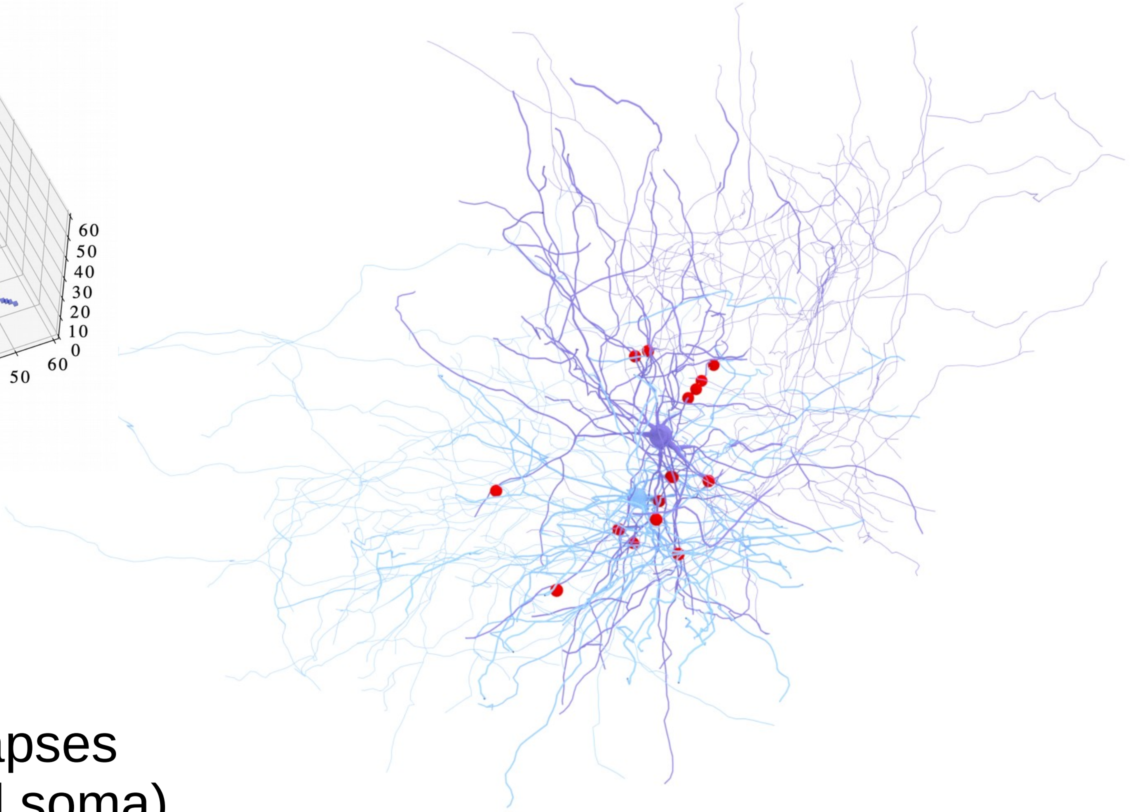
Requires:

- Complete morphologies with dendrites, and preferably also axons
- Detailed electrophysiological neuron models optimised using BluePyOpt
- Synapse models
- Connectivity statistics of the network



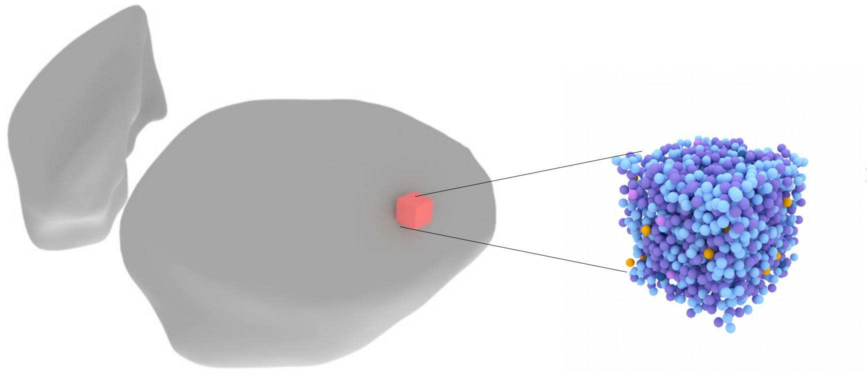
Voxel based touch detection

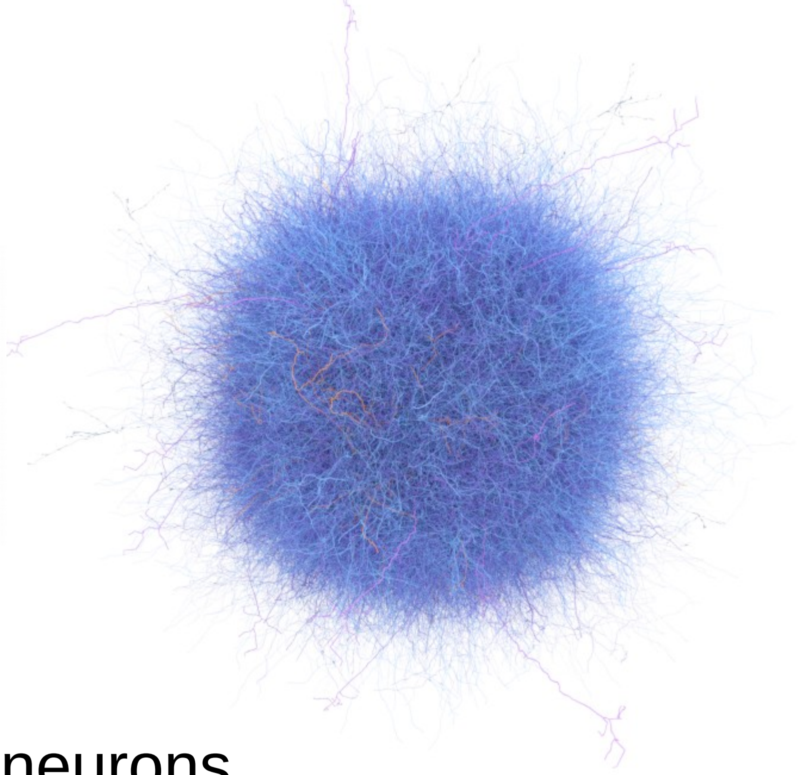




Space divided into voxels, 3µm side

Touch detection places putative synapses where axons are near dendrites (and soma)



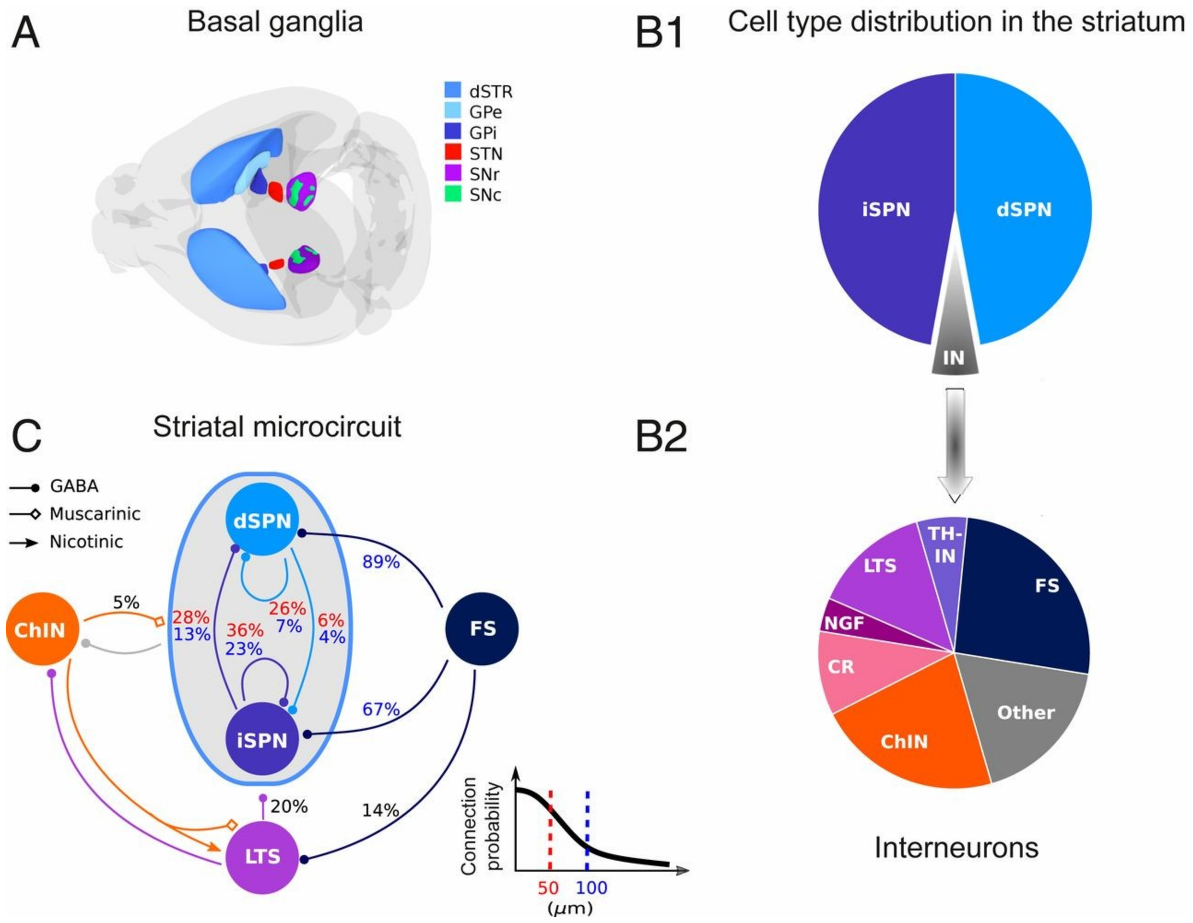


Entire mouse striatum has 1.72 million neurons

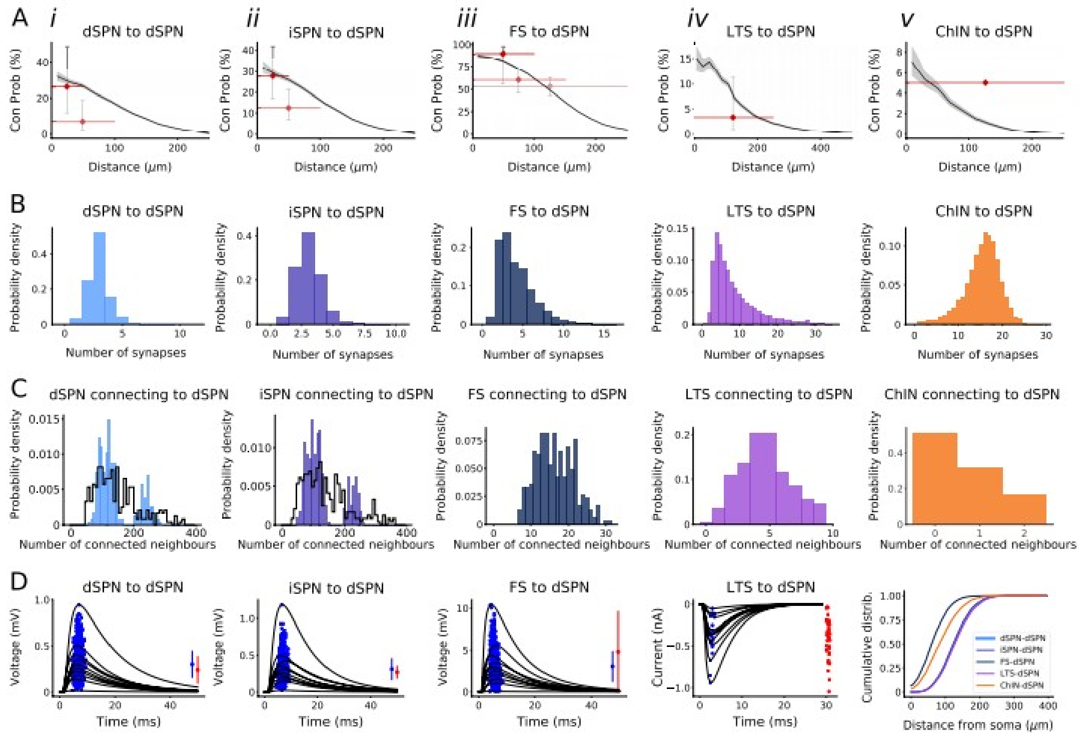
We detect 31 billion putative synapse locations

All locations does not have synapses -- we need to prune!

Basal Ganglia

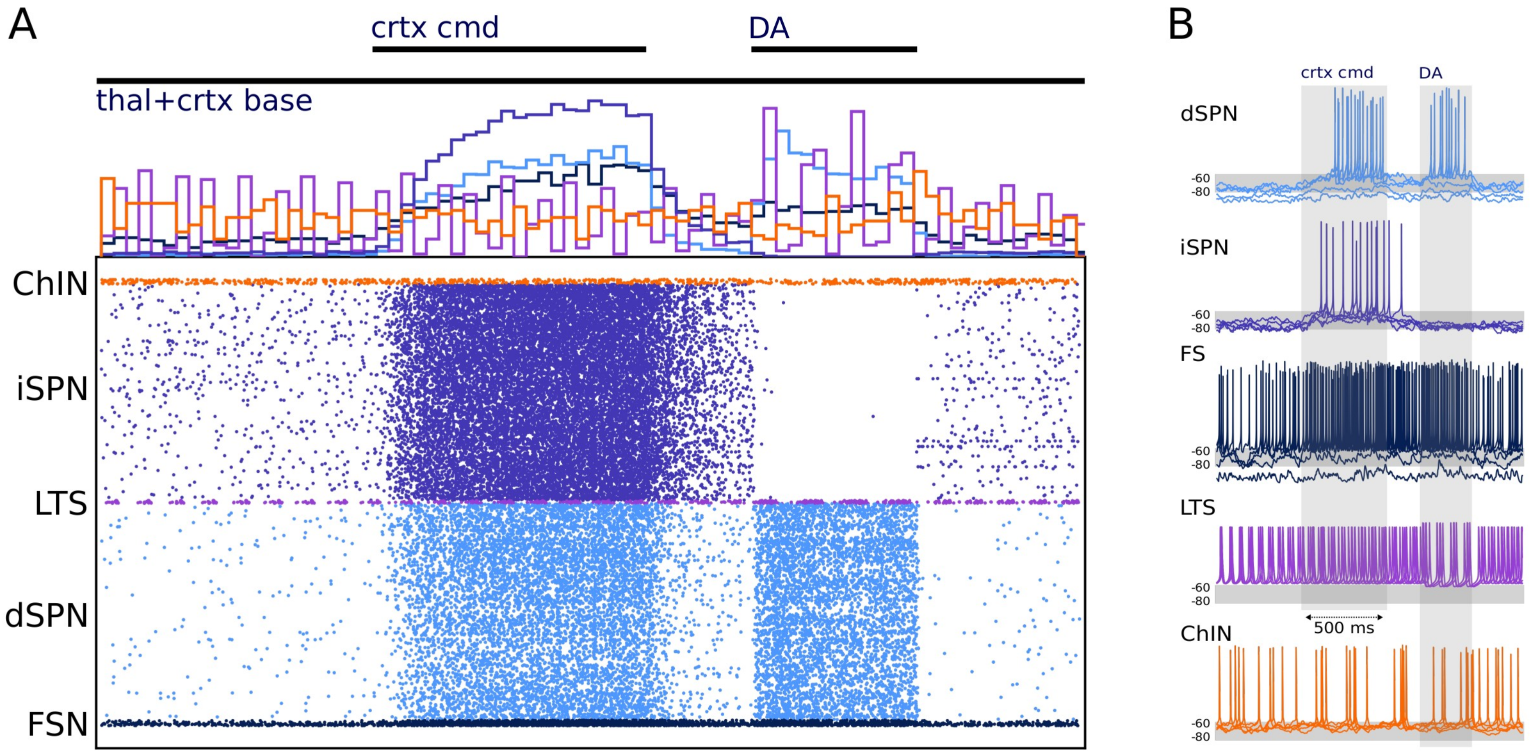


Prune synapses to match experimental pair-wise connection probability



Hjorth & Kozlov et al 2020

Simulation of Striatum



Hjorth & Kozlov et al 2020

Snudda

```
> pip install snudda
```

Repository:
https://github.com/Hjorthmedh/Snudda

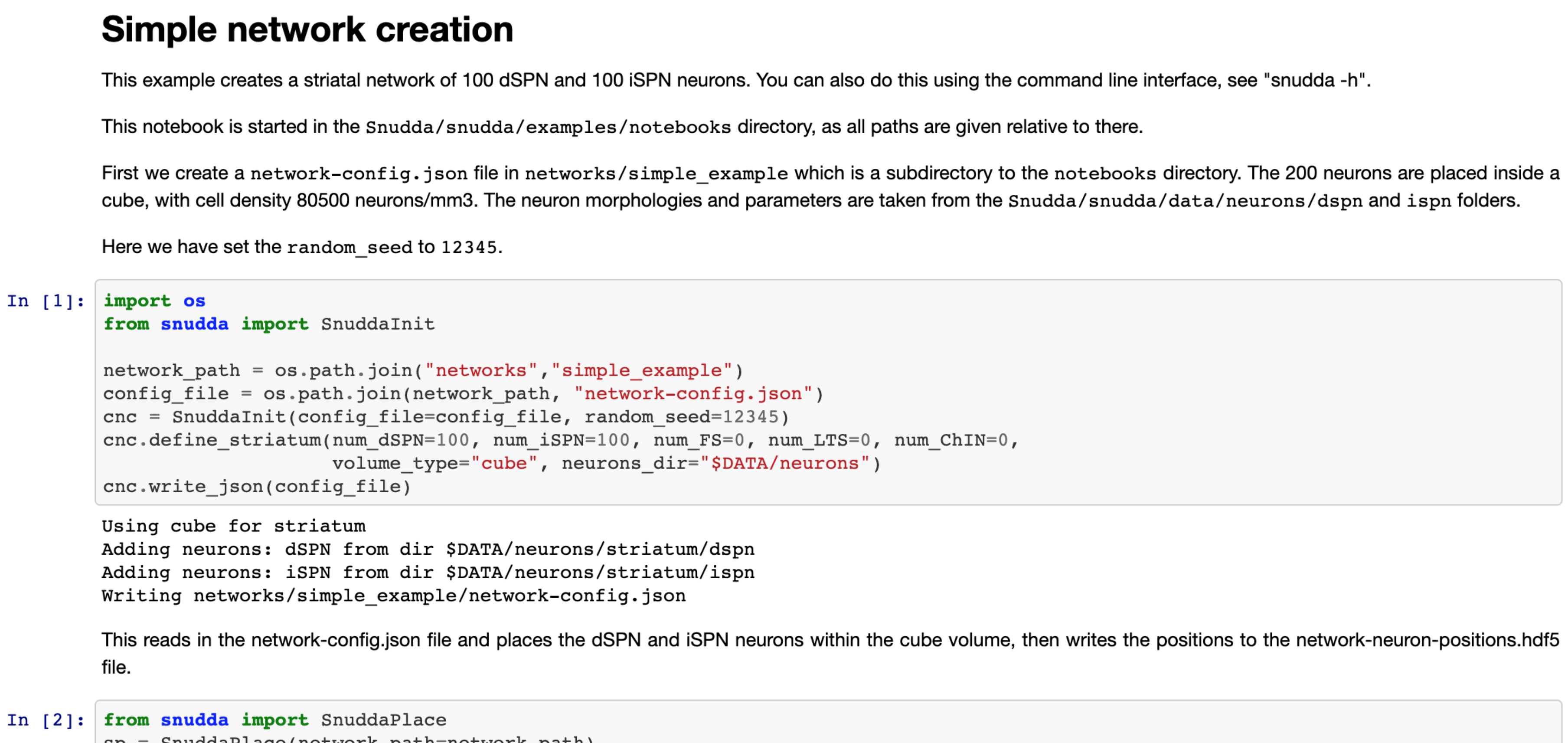
Jupyter Notebook:

```
> pip install notebook
> jupyter notebook
```

https://github.com/Hjorthmedh/Snudda/tree/master/examples/notebooks

Jupyter Notebooks on Github

https://github.com/Hjorthmedh/Snudda/tree/master/examples/notebooks



Simple network creation

This example creates a striatal network of 100 dSPN and 100 iSPN neurons. You can also do this using the command line interface, see "snudda -h".

This notebook is started in the `Snudda/snudda/examples/notebooks` directory, as all paths are given relative to there.

First we create a `network-config.json` file in `networks/simple_example` which is a subdirectory to the `notebooks` directory. The 200 neurons are placed inside a cube, with cell density 80500 neurons/mm3. The neuron morphologies and parameters are taken from the `Snudda/snudda/data/neurons/dspn` and `ispn` folders.

Here we have set the `random_seed` to `12345`.

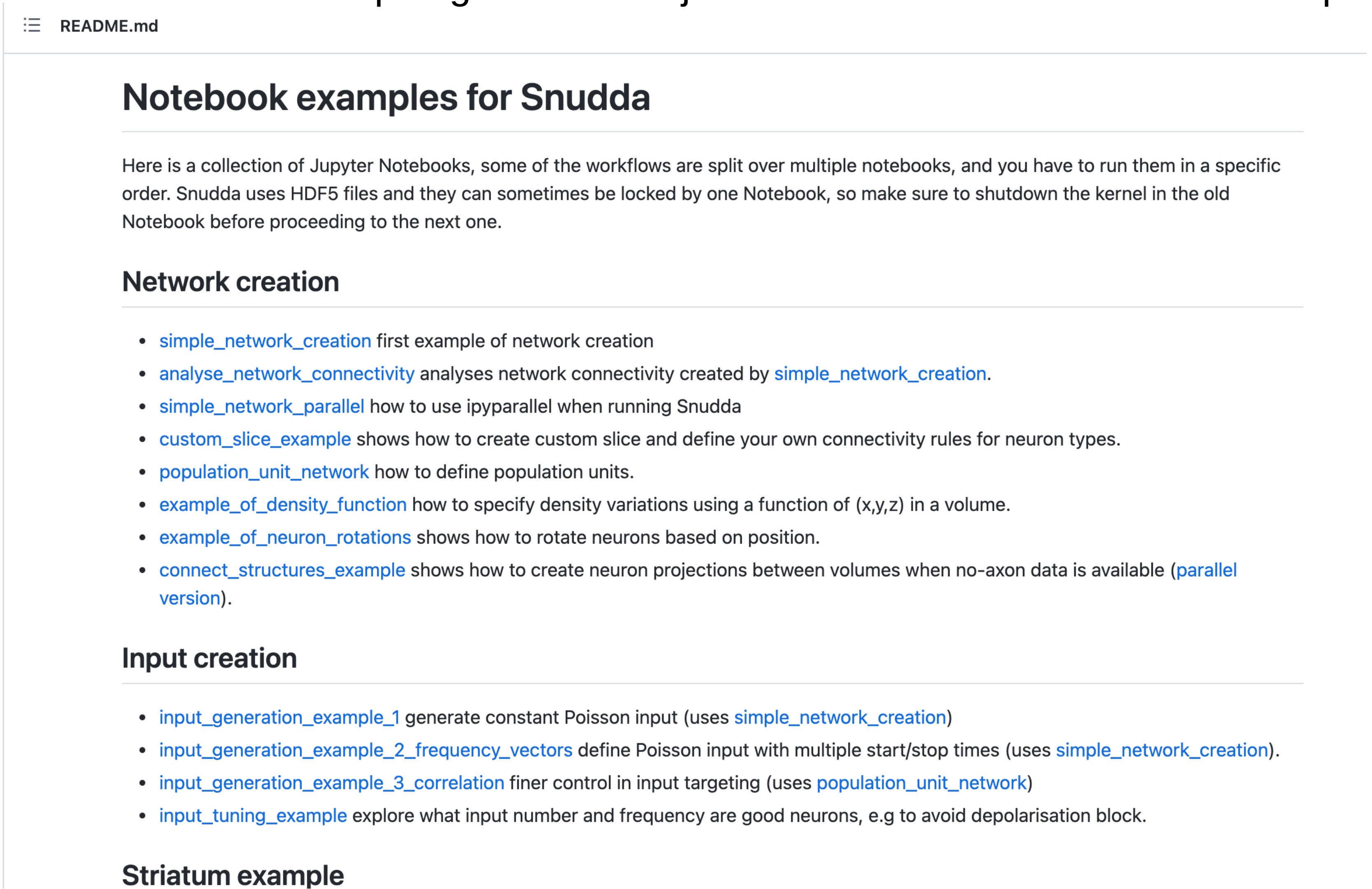
In [1]:
```
import os
from snudda import SnuddaInit

network_path = os.path.join("networks","simple_example")
config_file = os.path.join(network_path, "network-config.json")
cnc = SnuddaInit(config_file=config_file, random_seed=12345)
cnc.define_striatum(num_dSPN=100, num_iSPN=100, num_FS=0, num_LTS=0, num_ChIN=0,
                    volume_type="cube", neurons_dir="$DATA/neurons")
cnc.write_json(config_file)
```

```
Using cube for striatum
Adding neurons: dSPN from dir $DATA/neurons/striatum/dspn
Adding neurons: iSPN from dir $DATA/neurons/striatum/ispn
Writing networks/simple_example/network-config.json
```

This reads in the network-config.json file and places the dSPN and iSPN neurons within the cube volume, then writes the positions to the network-neuron-positions.hdf5 file.

In [2]:
```
from snudda import SnuddaPlace
sp = SnuddaPlace(network_path=network_path)
```

https://github.com/Hjorthmedh/Snudda/tree/master/examples/notebooks



README.md

Notebook examples for Snudda

Here is a collection of Jupyter Notebooks, some of the workflows are split over multiple notebooks, and you have to run them in a specific order. Snudda uses HDF5 files and they can sometimes be locked by one Notebook, so make sure to shutdown the kernel in the old Notebook before proceeding to the next one.

Network creation

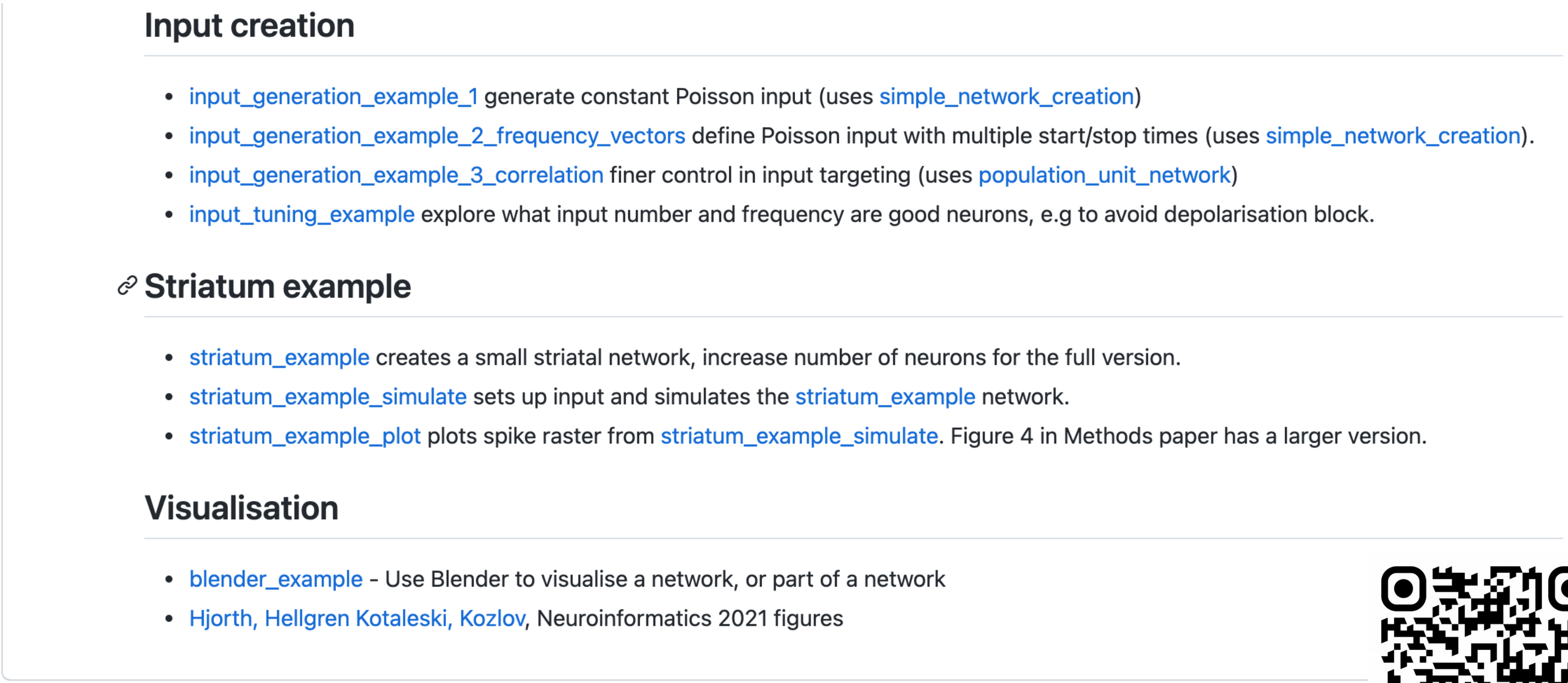
- simple_network_creation first example of network creation
- analyse_network_connectivity analyses network connectivity created by simple_network_creation.
- simple_network_parallel how to use ipyparallel when running Snudda
- custom_slice_example shows how to create custom slice and define your own connectivity rules for neuron types.
- population_unit_network how to define population units.
- example_of_density_function how to specify density variations using a function of (x,y,z) in a volume.
- example_of_neuron_rotations shows how to rotate neurons based on position.
- connect_structures_example shows how to create neuron projections between volumes when no-axon data is available (parallel version).

Input creation

- input_generation_example_1 generate constant Poisson input (uses simple_network_creation)
- input_generation_example_2_frequency_vectors define Poisson input with multiple start/stop times (uses simple_network_creation).
- input_generation_example_3_correlation finer control in input targeting (uses population_unit_network)
- input_tuning_example explore what input number and frequency are good neurons, e.g to avoid depolarisation block.

Striatum example

https://github.com/Hjorthmedh/Snudda/tree/master/examples/notebooks



Input creation

- input_generation_example_1 generate constant Poisson input (uses simple_network_creation)
- input_generation_example_2_frequency_vectors define Poisson input with multiple start/stop times (uses simple_network_creation).
- input_generation_example_3_correlation finer control in input targeting (uses population_unit_network)
- input_tuning_example explore what input number and frequency are good neurons, e.g to avoid depolarisation block.

Striatum example

- striatum_example creates a small striatal network, increase number of neurons for the full version.
- striatum_example_simulate sets up input and simulates the striatum_example network.
- striatum_example_plot plots spike raster from striatum_example_simulate. Figure 4 in Methods paper has a larger version.

Visualisation

- blender_example - Use Blender to visualise a network, or part of a network
- Hjorth, Hellgren Kotaleski, Kozlov, Neuroinformatics 2021 figures

To run in parallel

```
> export IPYTHONDIR="`pwd`/.ipython"
> export IPYTHON_PROFILE=Snudda_LOCAL

> ipcluster start -n 4 --profile=$IPYTHON_PROFILE --ip=127.0.0.1&
> sleep 20
> simName=networks/mynetwork
> snudda init $simName --size 10062 --overwrite
> snudda place $simName --parallel
> snudda detect $simName --parallel
> snudda prune $simName --parallel

> cp -a data/input-config/input-tinytest-v9-freq-vectors.json $simName/input.json
> snudda input $simName --parallel
> ipcluster stop
> mpiexec -n 6 snudda simulate $simName
```