Aston University

iqr simulator for large scale neural systems

Ulysses Bernardet
<u.bernardet@aston.ac.uk>

Aston University
Birmingham, UK

Agenda

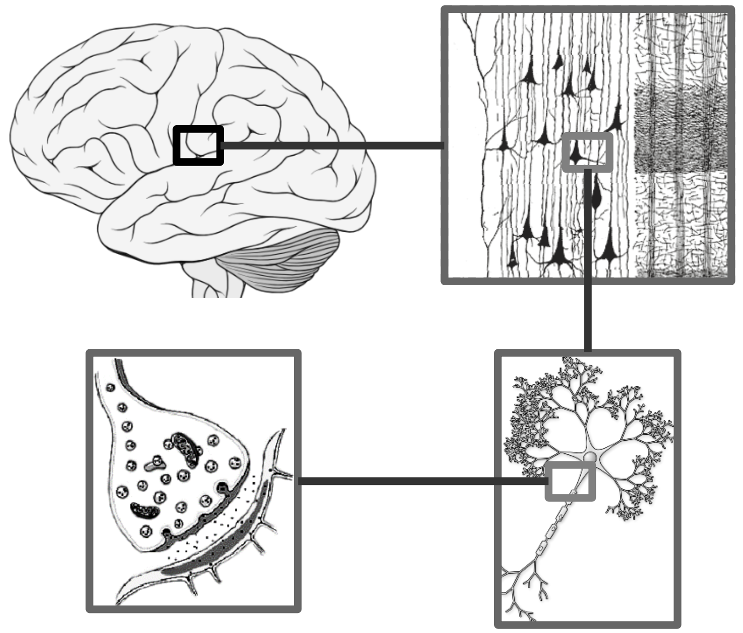
Topic	Type
Background and introduction to iqr	Lecture
Getting started with iqr	Tutorial
Neurons and connectivity	Lecture
Neuron types and connectivity patterns	Tutorial
Modules and interfaces	Lecture
Interfacing a camera	Tutorial
Advanced Features	Lecture

Ulysses Bernardet
<u.bernardet@aston.ac.uk>

Background and introduction to iqr

Understanding the brain

- The brain is an extraordinarily complex machine
- Workings can be described at a multitude of levels of abstractions (Sub-cellular…Areas)
- Different levels not mutually exclusive
  - Need to combine into a multi-level description
  - Only a holistic, systemic view can adequately explain the system under investigation



Bernardet, U., & Verschure, P. F. M. J. (2010). iqr: A Tool for the Construction of Multi-level Simulations of Brain and Behaviour. Neuroinformatics, 8(2), 113–34.

Ulysses Bernardet
<u.bernardet@aston.ac.uk>

The synthetic approach

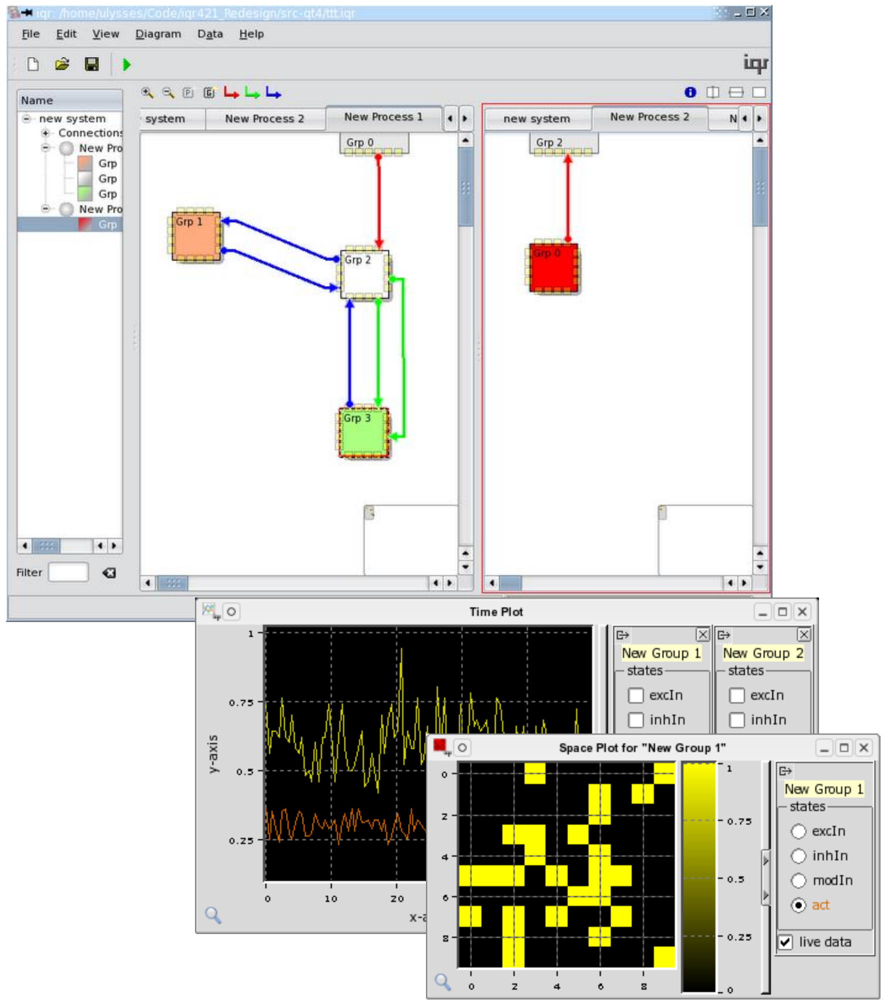
- Understanding systems through building them
- Process of building as such yields insights;
  - Explicitly state the target function of the system
  - Assign meaning to all elements and relations between elements of the system
- Building functioning systems, rather than detached models
  - Span from sensory processing to the behavioral output
- Synthetic systems are open to unlimited measurement and manipulation

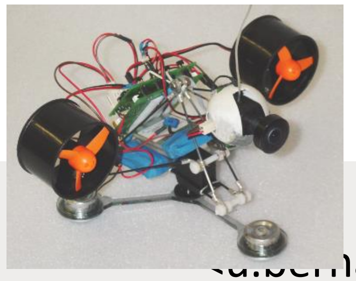
Verschure, P. F. M. J., & Althaus, P. (2003). A real-world rational agent: unifying old and new AI. Cognitive Science, 27(4), 561–590.

Ulysses Bernardet
<u.bernardet@aston.ac.uk>

The large-scale neuronal system simulator iqr

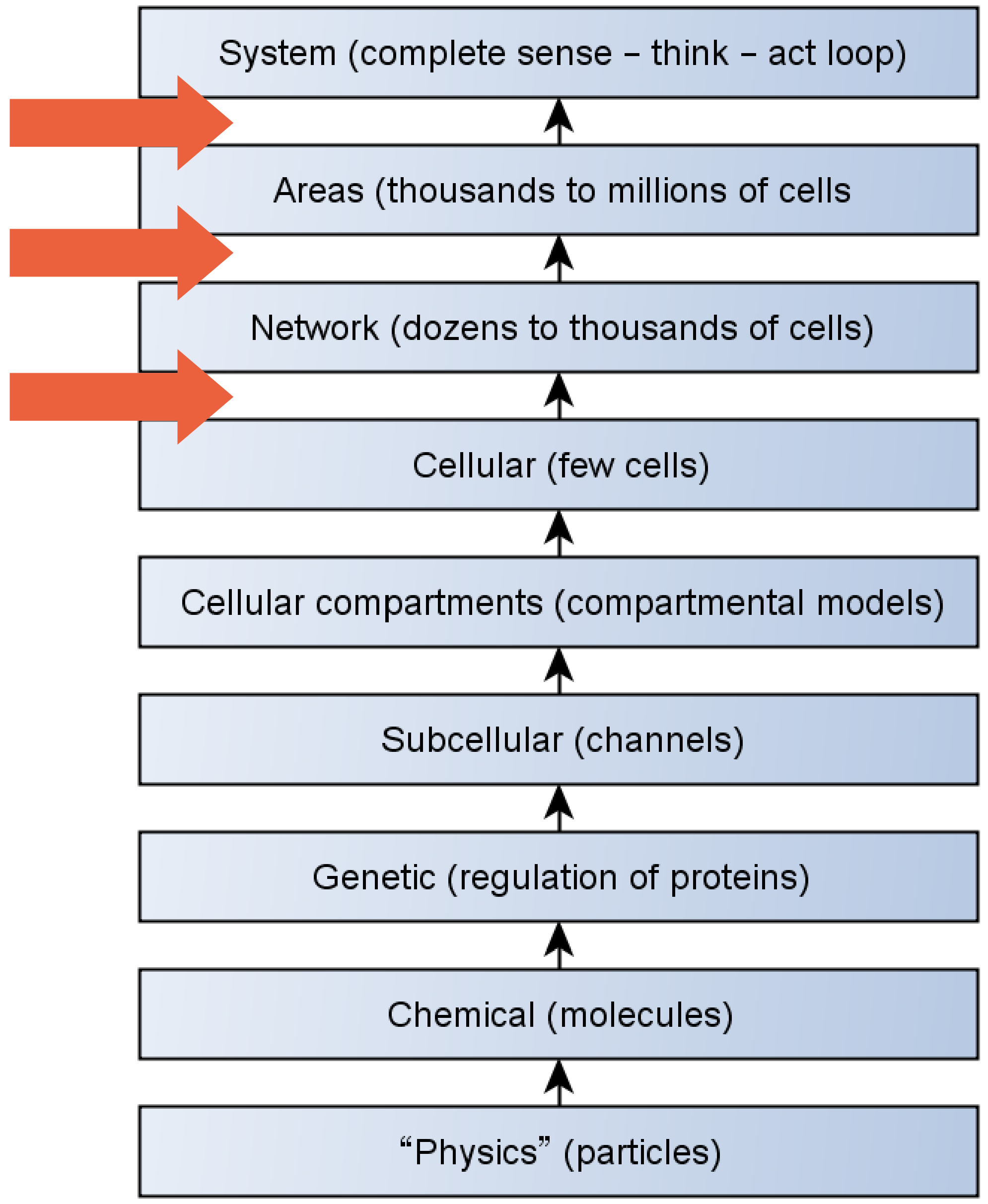
- Tool to support the synthetic approach
  - Real-world devices
  - Support for large-scale models
- Features
  - Graphical user interface for designing
  - On-line visualization and analysis of data
- Pre-defined neuron and synapse types
- Open architecture for new neurons, synapses, and hardware interfaces
- Open source: http://iqr.sourceforge.net





The “neuro-architectural” approach

- Information-flow approach that integrates
  - Structural and computational constraints from biology
  - Control principles from systems theory
  - Behavioral constraints from psychology / ethology
- Systems perspective → Functional goal (needed for behavioral constraints)
- Systems
  - Span from sensory processing to the behavioral output
  - large-scale, where the overall architecture is of critical importance

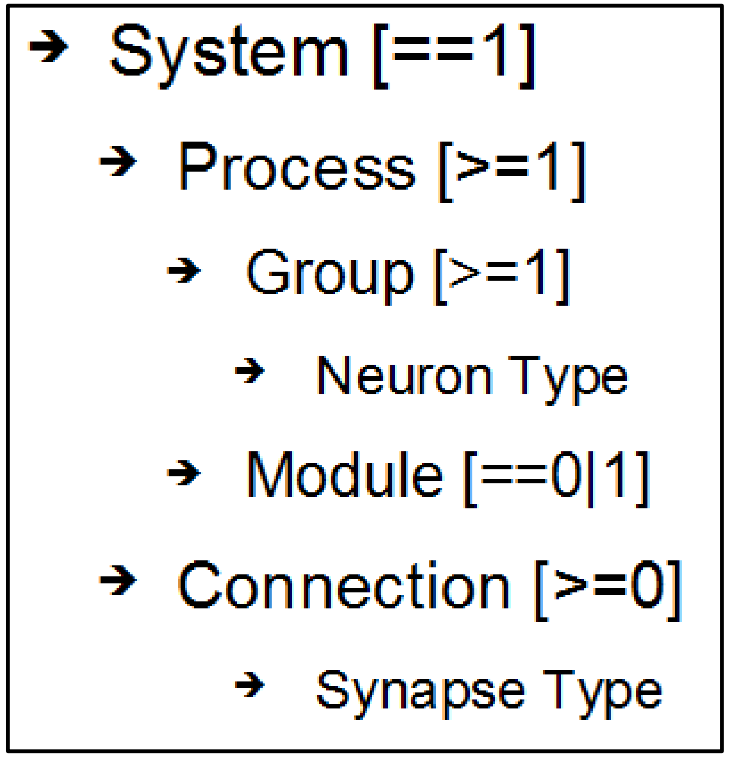

System (complete sense – think – act loop)
Areas (thousands to millions of cells
Network (dozens to thousands of cells)
Cellular (few cells)
Cellular compartments (compartmental models)
Subcellular (channels)
Genetic (regulation of proteins)
Chemical (molecules)
“Physics” (particles)

Aston University

Ulysses Bernardet
<u.bernardet@aston.ac.uk>

System architecture

- System: Top-level unit
- Processes: Organizational units → no computational functions
- Group: Specific aggregation of neurons of identical type
- Connection:
  - Feed information from group to group
  - Aggregation of synapses of identical type
  - Definition of the arrangement of the synapses
- Connection framework:
  - Deals with axon, synapse, and dendrite of the neurons
  - Computational element: Synapse



- ➔ System [==1]
  - ➔ Process [>=1]
    - ➔ Group [>=1]
      - ➔ Neuron Type
    - ➔ Module [==0|1]
  - ➔ Connection [>=0]
    - ➔ Synapse Type

Aston University

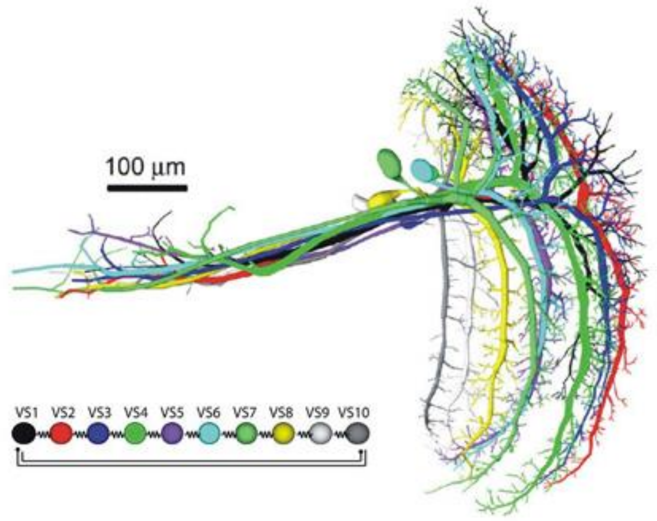
Ulysses Bernardet
<u.bernardet@aston.ac.uk>

Neurons in iqr

- Neurons are the core computational units in brain-like information processing system
- Different “Standard Types” provided by iqr
- iqr provides a simple framework for defining your own types of neurons
- No difference between user-defined neurons and “standard” types

Ulysses Bernardet
<u.bernardet@aston.ac.uk>

Connectivity



- A human brain consists of ~85 billion neurons
- Neurons outnumbered by synapses
  - Synapses per average neuron: 1,000 - 10,00
- Ratio of number of neurons vs. number of synapses
  - Biological neuronal networks draw their computational power from the large-scale integration of information in the connectivity between neurons
- Large scale neuronal models require heterogeneous connectivity patterns
- Providing a flexible, yet easy to use way to define connectivity is a cornerstone of iqr

Ulysses Bernardet
<u.bernardet@aston.ac.uk>

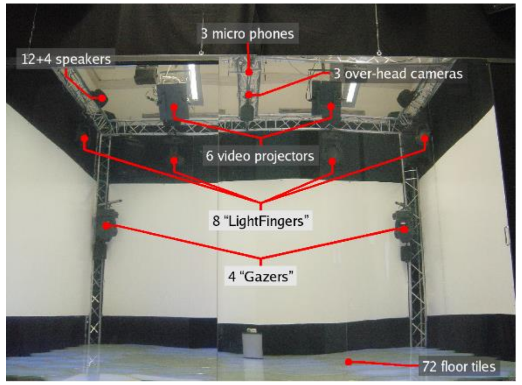
Simulations in iqr

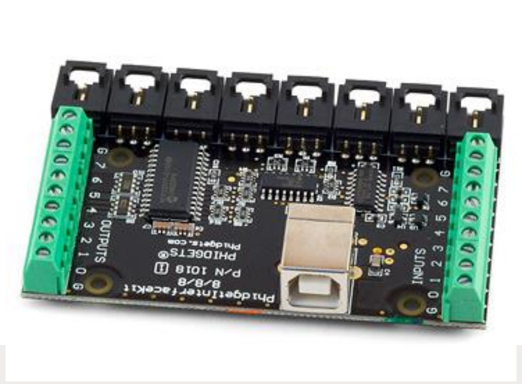
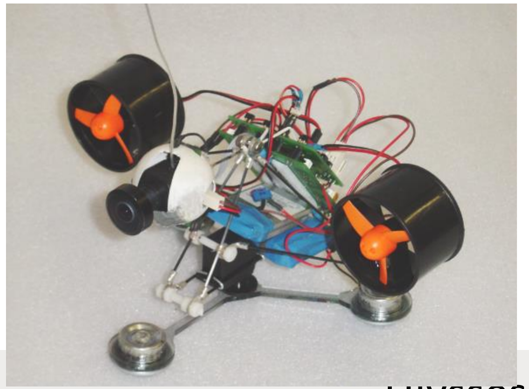
- Computation
  - Spiking / rate coding
  - Compartmental / abstract
  - Discrete vs continuous simulation
- Architecture
  - Feedforward / feedback
  - Learning / non-learning

Ulysses Bernardet
<u.bernardet@aston.ac.uk>

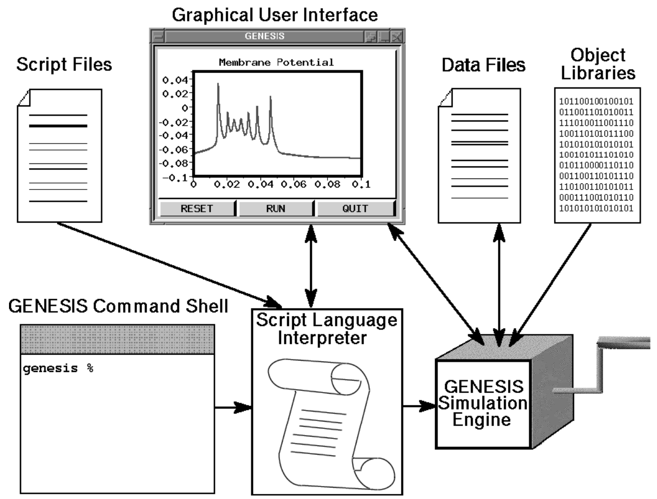
Modules: Interfaces to real-world devices

- Cameras
- Audio
- Pan-tilt system
- Robots
  - K-team robots
  - E-puck
  - Blimps
  - Strider
  - iCub (via YARP)
  - …..

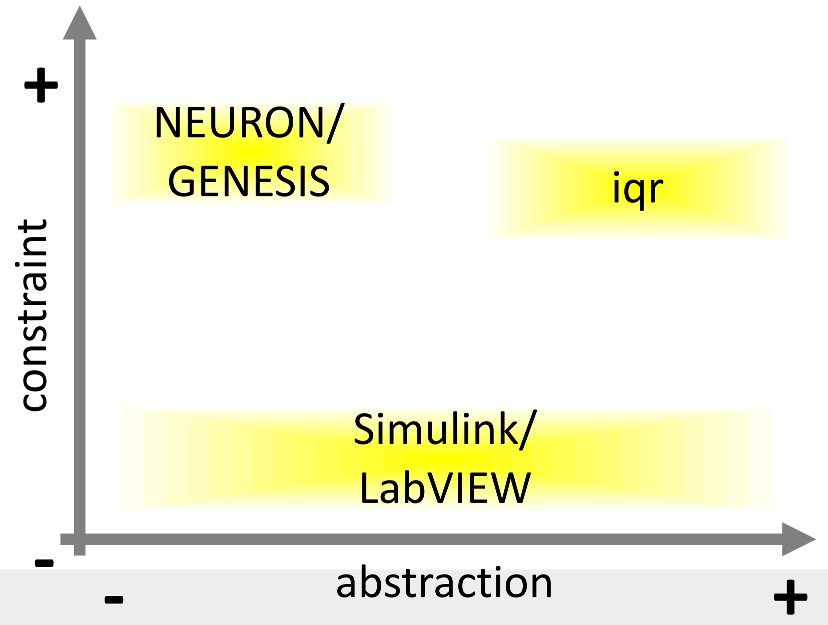


Simulators



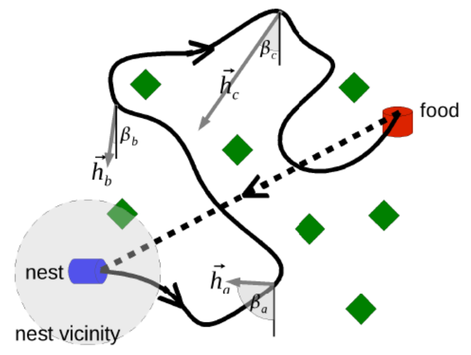
- ➔ GENESIS
  Simulation environment for constructing realistic models of neurobiological systems at many levels of scale including subcellular processes, individual neurons, networks of neurons, and neuronal systems.
- ➔ NEURON
  models individual neurons via the use of sections which are subdivided into individual compartments by the program. Programs can be written interactively in a shell, or loaded from a file

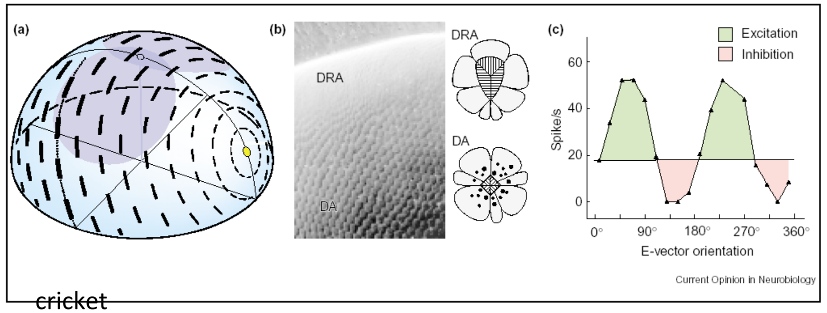


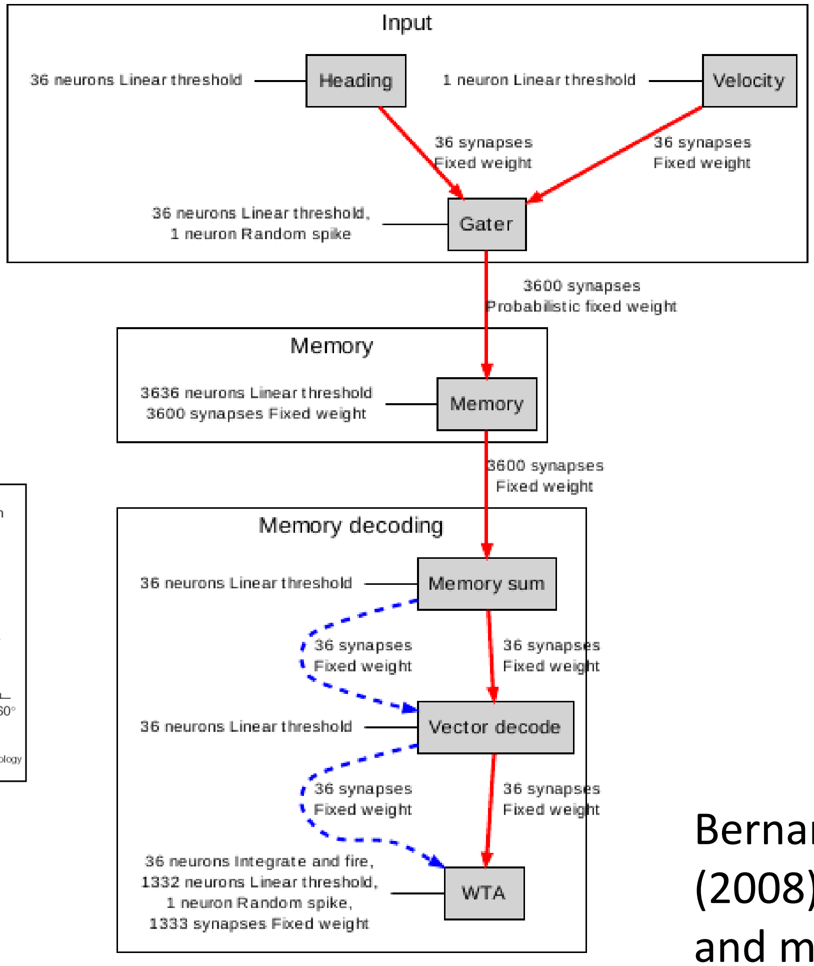
- ➔ LabVIEW
  Platform and development environment for a visual programming language. Commonly used for data acquisition, instrument control, and industrial automation.
- ➔ Simulink
  Tool for modeling, simulating and analyzing multidomain dynamic systems. Its primary interface is a graphical block diagramming tool and a customizable set of block libraries.

Ulysses Bernardet
<u.bernardet@aston.ac.uk>

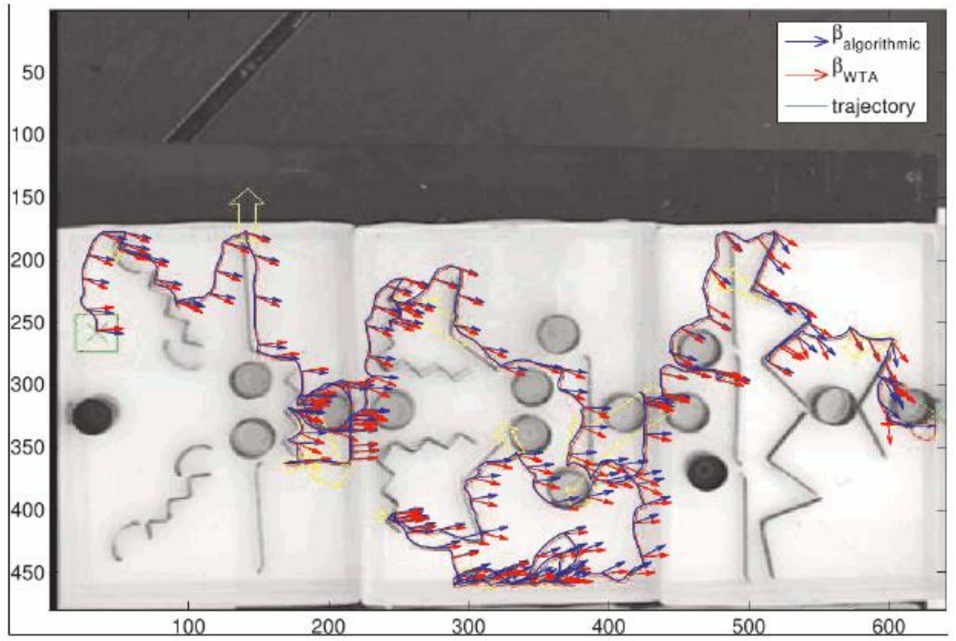
Example: Path integration and orientation in insects







Input
36 neurons Linear threshold
Heading
1 neuron Linear threshold
Velocity
36 synapses Fixed weight
36 synapses Fixed weight
36 neurons Linear threshold, 1 neuron Random spike
Gater
3600 synapses Probabilistic fixed weight
Memory
3636 neurons Linear threshold 3600 synapses Fixed weight
Memory
3600 synapses Fixed weight
Memory decoding
36 neurons Linear threshold
Memory sum
36 synapses Fixed weight
36 synapses Fixed weight
36 neurons Linear threshold
Vector decode
36 synapses Fixed weight
36 synapses Fixed weight
36 neurons Integrate and fire, 1332 neurons Linear threshold, 1 neuron Random spike, 1333 synapses Fixed weight
WTA



Bernardet, U., Bermúdez I Badia, S., & Verschure, P. F. M. J. (2008). A model for the neuronal substrate of dead reckoning and memory in arthropods: a comparative computational and behavioral study. Theory in biosciences, 127(2), 163-175. Springer Berlin

Aston University

Ulysses Bernardet
<u.bernardet@aston.ac.uk>

iqr facts

- Open Source (Gnu Public License)
  hosted on SourceForge http://sourceforge.net/projects/iqr
  – Repository of iqr packages
  – Documentation
  – Bug reports
  – Feature request
  – Browse and check out the source code of iqr
- 60’512 lines of C++ code
  (development effort estimate in Person-Months: 190.66, estimated cost to develop: 2’146’350$ (generated using David A. Wheeler’s “SLOCCount”))
- Qt widget set
- Multi-threaded
- XML system description

WORKING WITH IQR

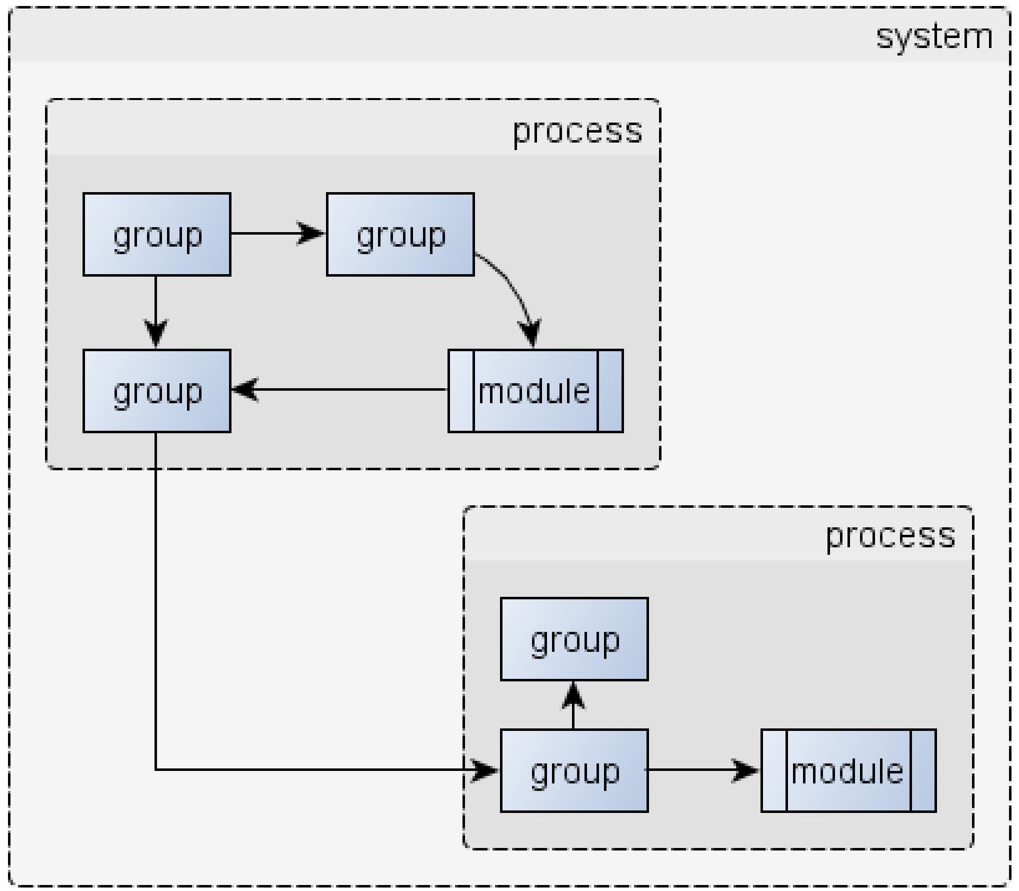
Ulysses Bernardet
<u.bernardet@aston.ac.uk>

Demo

GUI INTRO

Models in iqr

- System [==1]
  - Process [>=1]
    - Group [>=1]
      - Neuron [>=1]
    - Module [==0|1]
  - Connection [>=0]
    - Synapse [>=1]

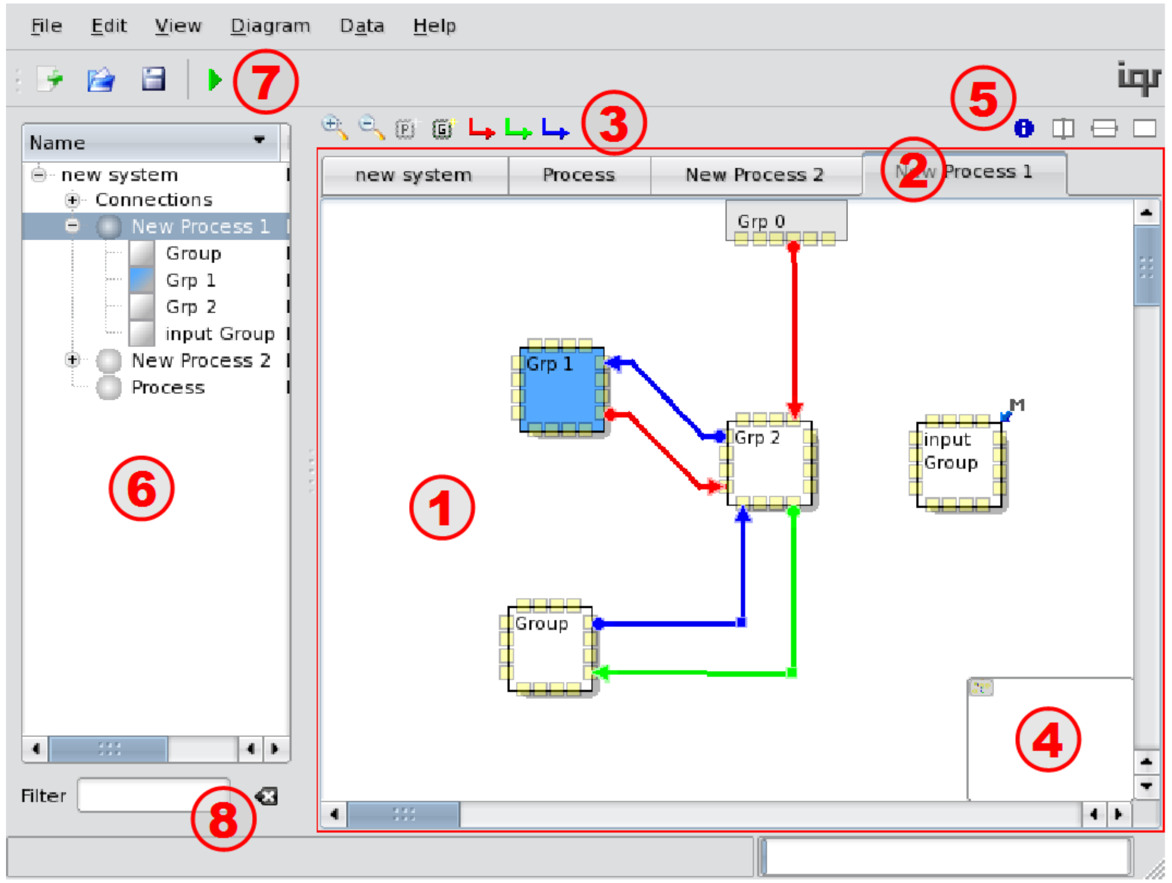


Aston University

Ulysses Bernardet
<u.bernardet@aston.ac.uk>

The iqr GUI

1. diagram editing pane
2. tab-bar
3. diagram toolbar
4. panner
5. splitter
6. browser
7. simulation toolbar



Split the diagram editing pane into two separate views by using the **splitter** (fig. 2.1⑤). From left to right: split vertically, horizontally, revert to single window view.

Zoom in and out of the diagram

Add a new process to the system level

Add a new group to the current process

Add a new connection between groups: excitatory (red), modulatory (green), inhibitory (blue)

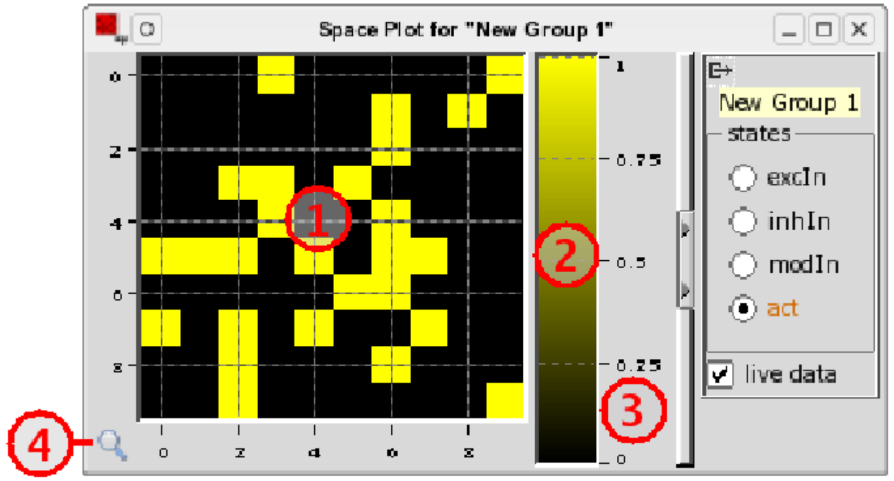
Ulysses Bernardet
<u.bernardet@aston.ac.uk>

Data Visualization and Collection

Ulysses Bernardet
<u.bernardet@aston.ac.uk>

Space plots

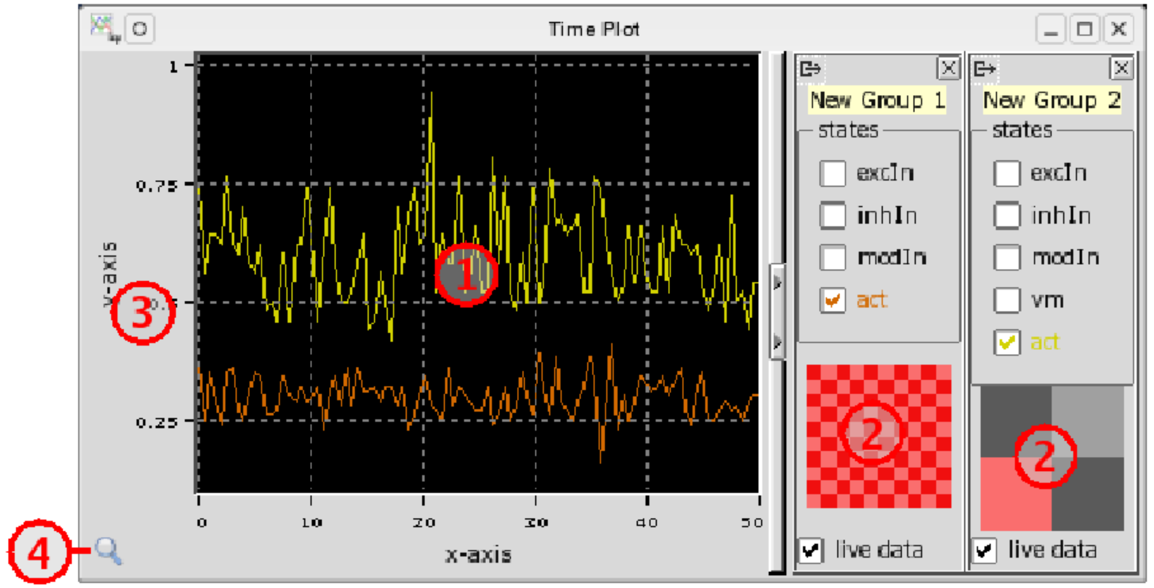
- A Space plot as shown displays the state of each cell in a group in the plot area
- To create a new Space plot
  - right click on a group in diagram editing pane or browser
  - select Space plot from the context-menu
- The value for the selected state is color coded, the range indicated in the color bar
- A Space plot plots one state of one single group



Ulysses Bernardet
<u.bernardet@aston.ac.uk>

Time plots

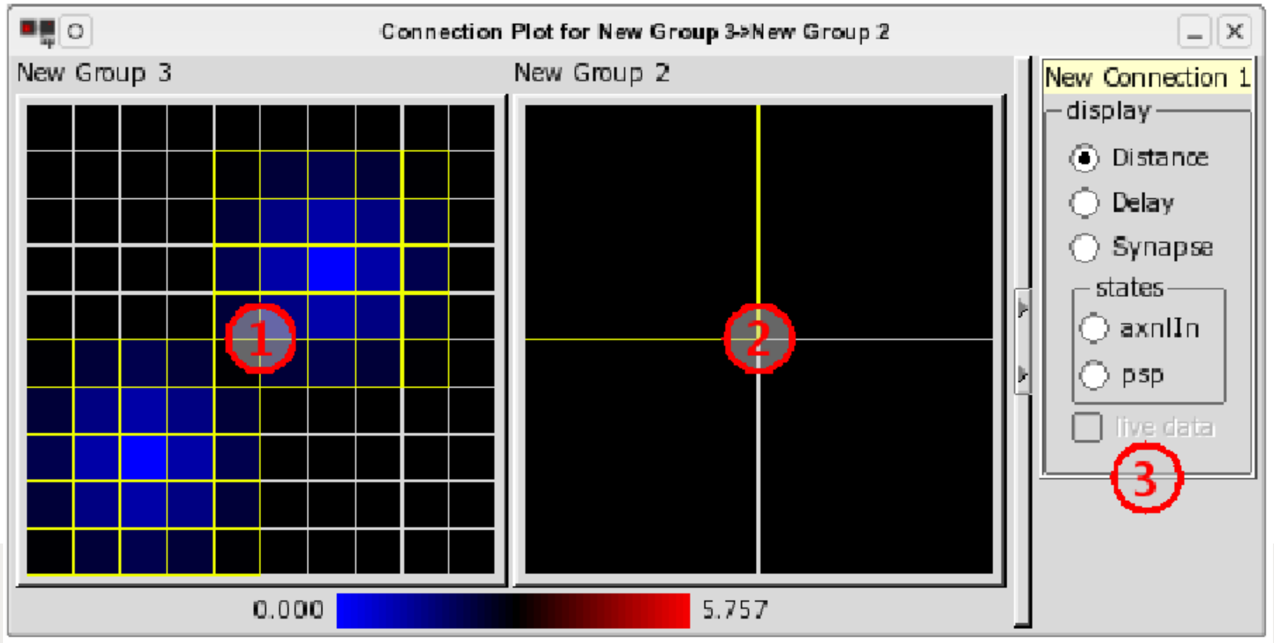
- Plots state(s) of neurons against time
- Time plots display the average value of the states of an entire group or region of a group
- A Time plot can
  - plot several states
  - states from different groups
- Each state is plotted as separate trace
- drag & drop
  - group
  - region



Ulysses Bernardet
<u.bernardet@aston.ac.uk>

Connection plots

- Visualize the static and dynamic properties of connections and synapses.
- left: source group, right: target group
- Static properties of a connection, i.e. the Distance or Delay
- Internal states by selecting Synapse , and one of the states in the list



Ulysses Bernardet
<u.bernardet@aston.ac.uk>

Saving + loading configurations

- You can save the arrangement and the properties of the current set of plots and Data Sampler
- menu:
  - Data → Save Configuration
  - Data → Open Configuration

Ulysses Bernardet
<u.bernardet@aston.ac.uk>

Hands-on

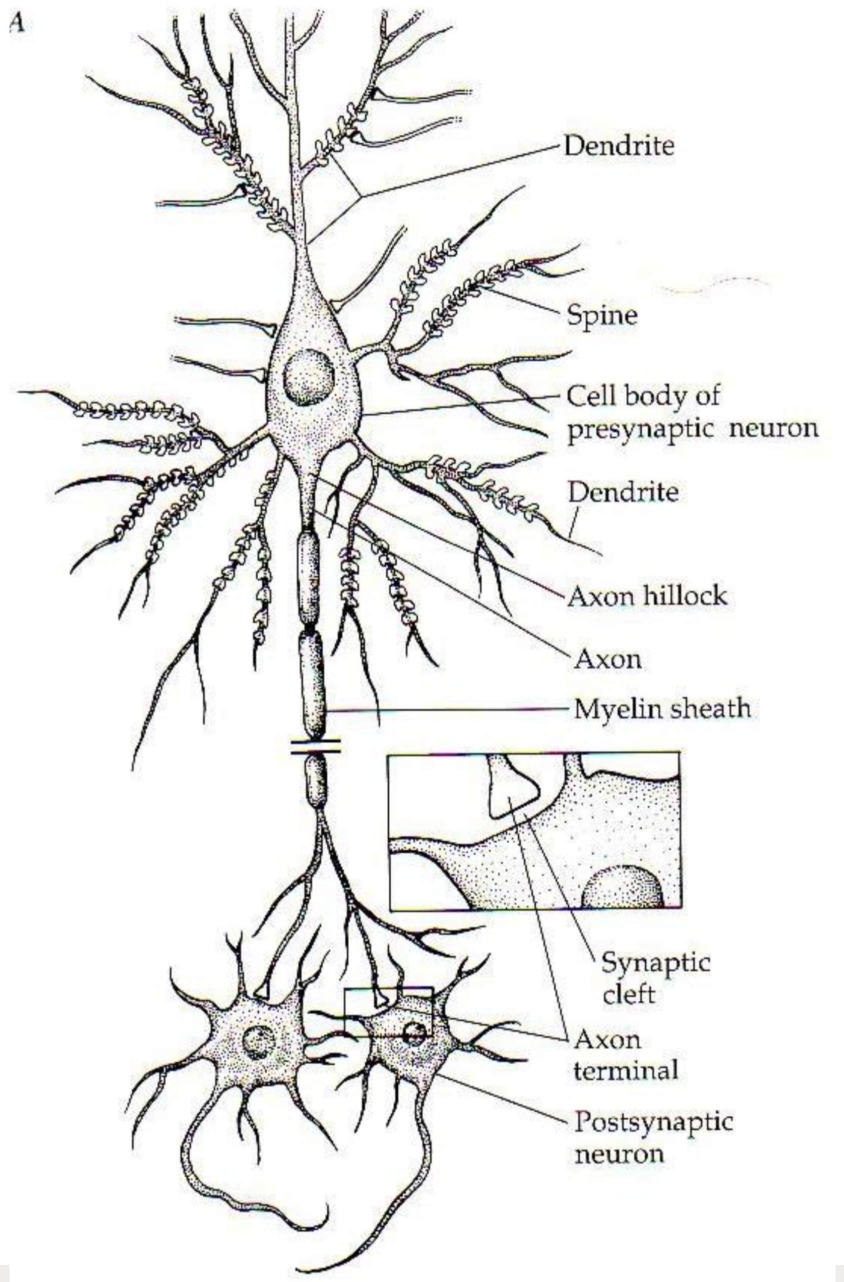
iqr Tutorial 01-Creating a simulation

Neurons in iqr

Ulysses Bernardet
<u.bernardet@aston.ac.uk>

Basic morphology of neurons

- Neurons are the functional cellular unit the nervous system
- Three main parts:
  - Soma (1)
  - Dendrite (>=1)
  - Axon (==1)
- Soma and dendrites receive signals from other neurons
- Axon are transmitting signals to other neurons
- Synapse
  - Contact point between the axon of one neuron and a dendrite or soma of another
  - Can be excitatory or inhibitory



Ulysses Bernardet
<u.bernardet@aston.ac.uk>

Neurons in iqr

- Neurons are the core computational units in brain-like information processing system
- Different "Standard Types" provided by iqr
- iqr provides a simple framework for defining your own types of neurons
- No difference between user-defined neurons and "standard" types

Ulysses Bernardet
<u.bernardet@aston.ac.uk>

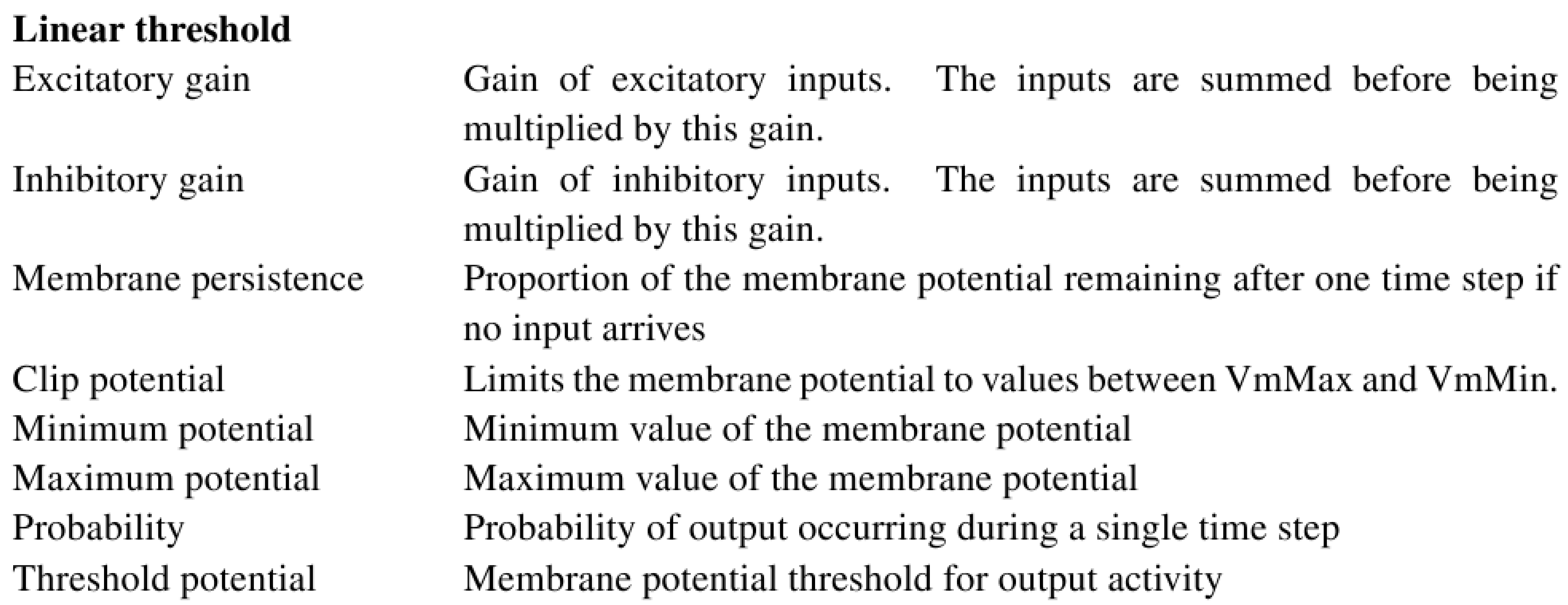
Linear Threshold neurons

$$a_i(t+1) = \begin{cases} v_i(t+1) & \text{with probability } Prob \text{ for } v_i(t+1) \geq ThSet \\ 0 & \text{otherwise} \end{cases}$$

$$\begin{aligned} v_i(t+1) = VmPrs_i v_i(t) \\ &+ ExcGain_i \sum_{j=1}^{m} w_{ij} a_j(t - \delta_{ij}) \\ &- InhGain_i \sum_{k=1}^{n} w_{ik} a_k(t - \delta_{ik}) \end{aligned}$$

Ulysses Bernardet
<u.bernardet@aston.ac.uk>

Linear Threshold neurons



Linear threshold	
Excitatory gain	Gain of excitatory inputs. The inputs are summed before being multiplied by this gain.
Inhibitory gain	Gain of inhibitory inputs. The inputs are summed before being multiplied by this gain.
Membrane persistence	Proportion of the membrane potential remaining after one time step if no input arrives
Clip potential	Limits the membrane potential to values between VmMax and VmMin.
Minimum potential	Minimum value of the membrane potential
Maximum potential	Maximum value of the membrane potential
Probability	Probability of output occurring during a single time step
Threshold potential	Membrane potential threshold for output activity

Ulysses Bernardet
<u.bernardet@aston.ac.uk>

Integrate & Fire neurons

$$a_i(t+1) = \begin{cases} SpikeAmpl & \text{with probability } Prob \text{ for } v_i(t+1) \geq ThSet \\ 0 & \text{otherwise} \end{cases}$$

$$\begin{aligned} v_i(t+1) = VmPrs_i v_i(t) & \\ & + ExcGain_i \sum_{j=1}^{m} w_{ij} a_j(t - \delta_{ij}) \\ & - InhGain_i \sum_{k=1}^{n} w_{ik} a_k(t - \delta_{ik}) \end{aligned}$$

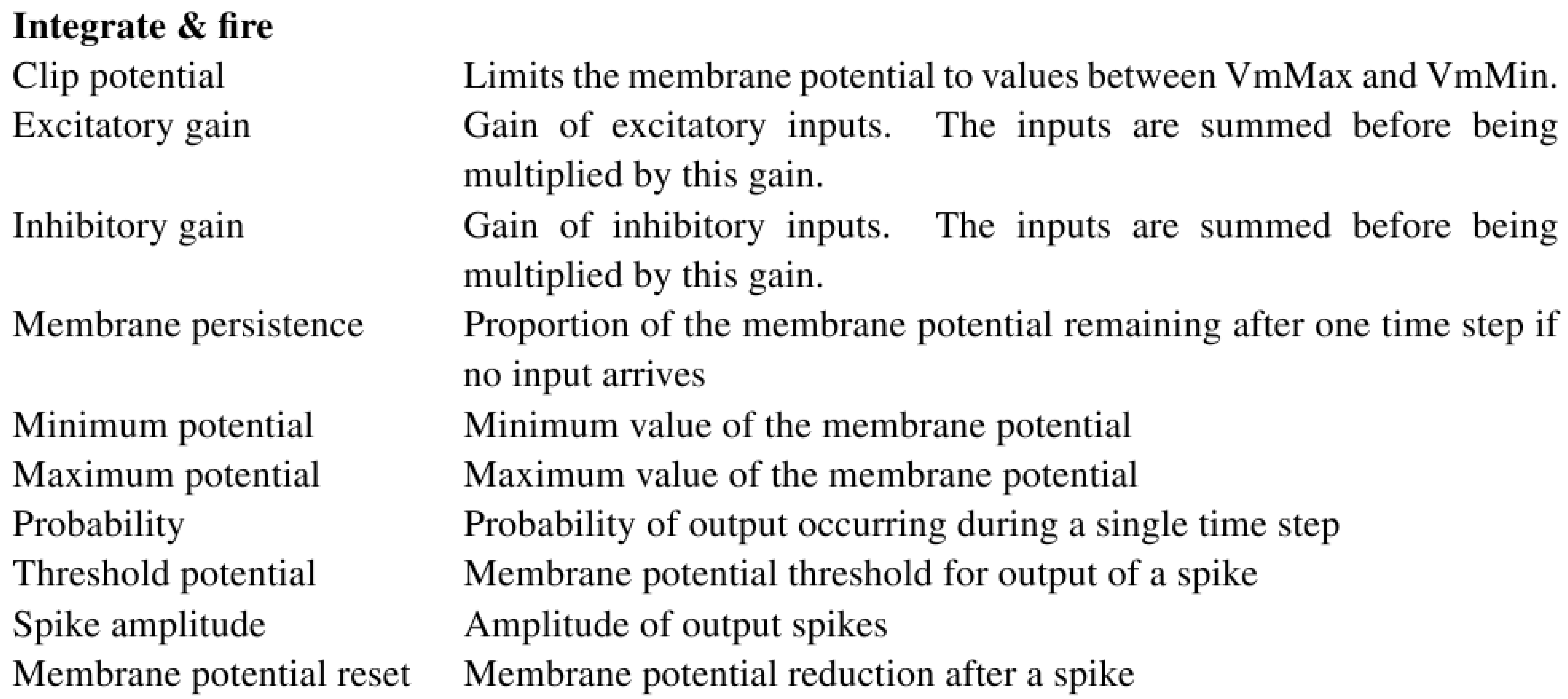
After cell i produces a spike, the membrane potential is hyperpolarized such that

$$v_i'(t+1) = v_i(t+1) - VmReset$$

Aston University

Ulysses Bernardet
<u.bernardet@aston.ac.uk>

Integrate & Fire neurons

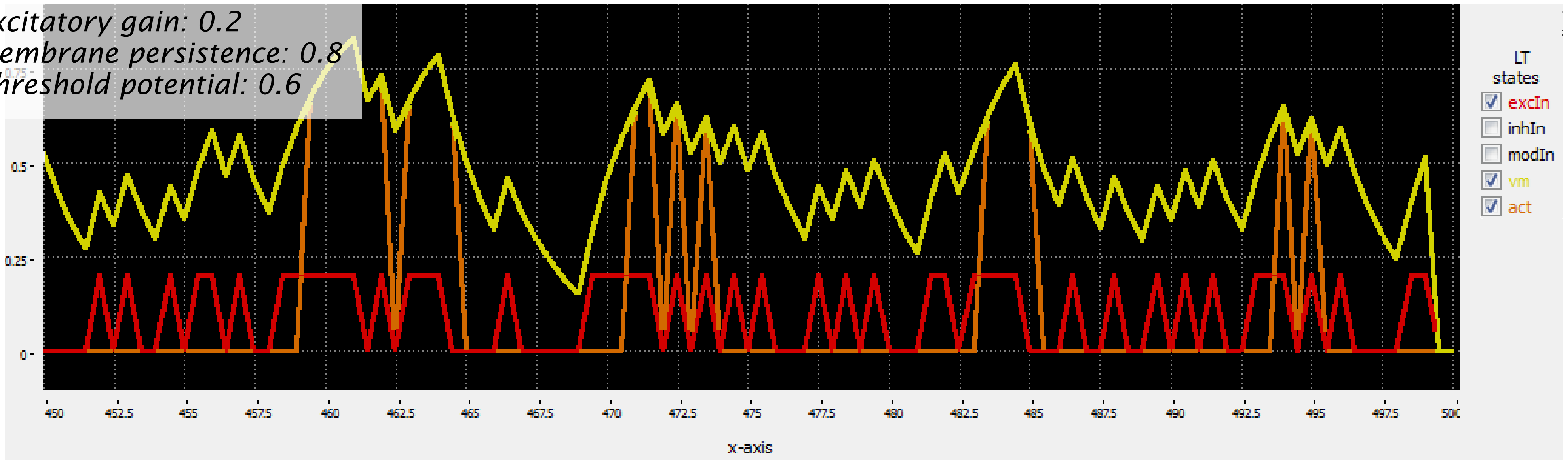


Integrate & fire	
Clip potential	Limits the membrane potential to values between VmMax and VmMin.
Excitatory gain	Gain of excitatory inputs. The inputs are summed before being multiplied by this gain.
Inhibitory gain	Gain of inhibitory inputs. The inputs are summed before being multiplied by this gain.
Membrane persistence	Proportion of the membrane potential remaining after one time step if no input arrives
Minimum potential	Minimum value of the membrane potential
Maximum potential	Maximum value of the membrane potential
Probability	Probability of output occurring during a single time step
Threshold potential	Membrane potential threshold for output of a spike
Spike amplitude	Amplitude of output spikes
Membrane potential reset	Membrane potential reduction after a spike

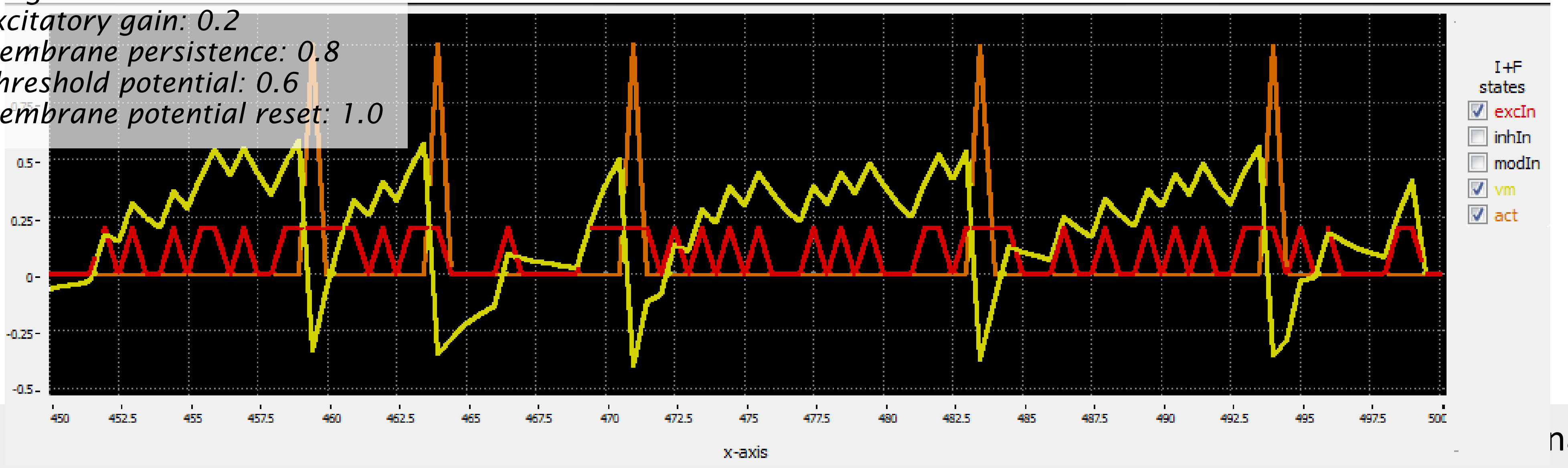
Ulysses Bernardet
<u.bernardet@aston.ac.uk>

Neuron behavior

Linear Threshold
Excitatory gain: 0.2
Membrane persistence: 0.8
Threshold potential: 0.6



Integrate + Fire
Excitatory gain: 0.2
Membrane persistence: 0.8
Threshold potential: 0.6
Membrane potential reset: 1.0



Random spike neurons

- Releases a spike per timestep with a user-defined spike probability
- Time series of the output spikes forms a Poisson process (i.e. large gaps between events are less likely to occur than short ones)
- Receives no input and has no membrane potential

Random spike	
Probability	Probability of a spike occurring during a single time step
Spike amplitude	Amplitude of each spike

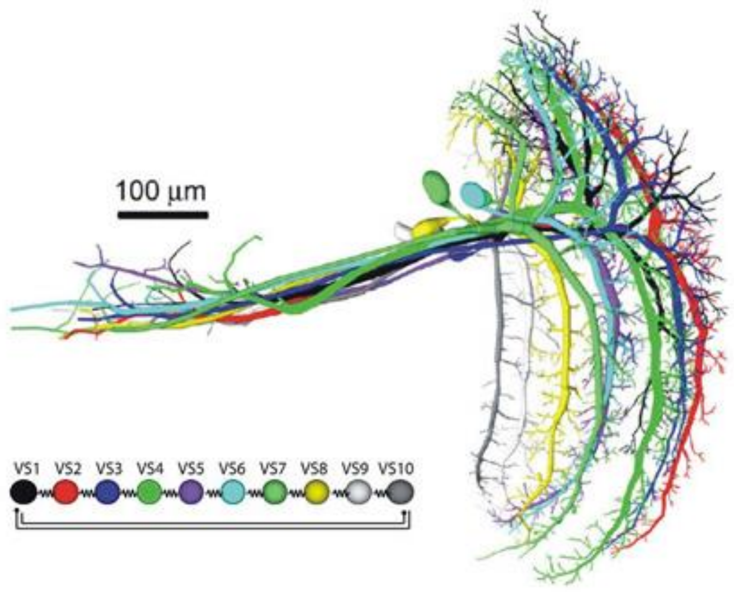
$$a_i(t+1) = \begin{cases} SpikeAmpl & \text{with probability } Prob \\ 0 & \text{otherwise} \end{cases}$$

CONNECTIVITY FRAMEWORK

Ulysses Bernardet
<u.bernardet@aston.ac.uk>

The connections challenge

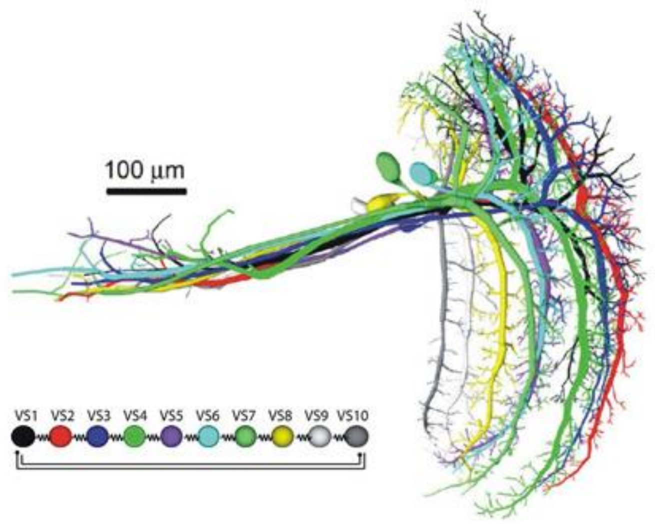
- Large scale neuronal models require heterogeneous connectivity patterns
- Providing a flexible, yet easy to use way to define connectivity is a cornerstone of iqr
- The definition needs to be compact and broad
- A generic way to define connectivity can be used to describe anatomical data in general



Aston University

Ulysses Bernardet
<u.bernardet@aston.ac.uk>

The power of connectivity

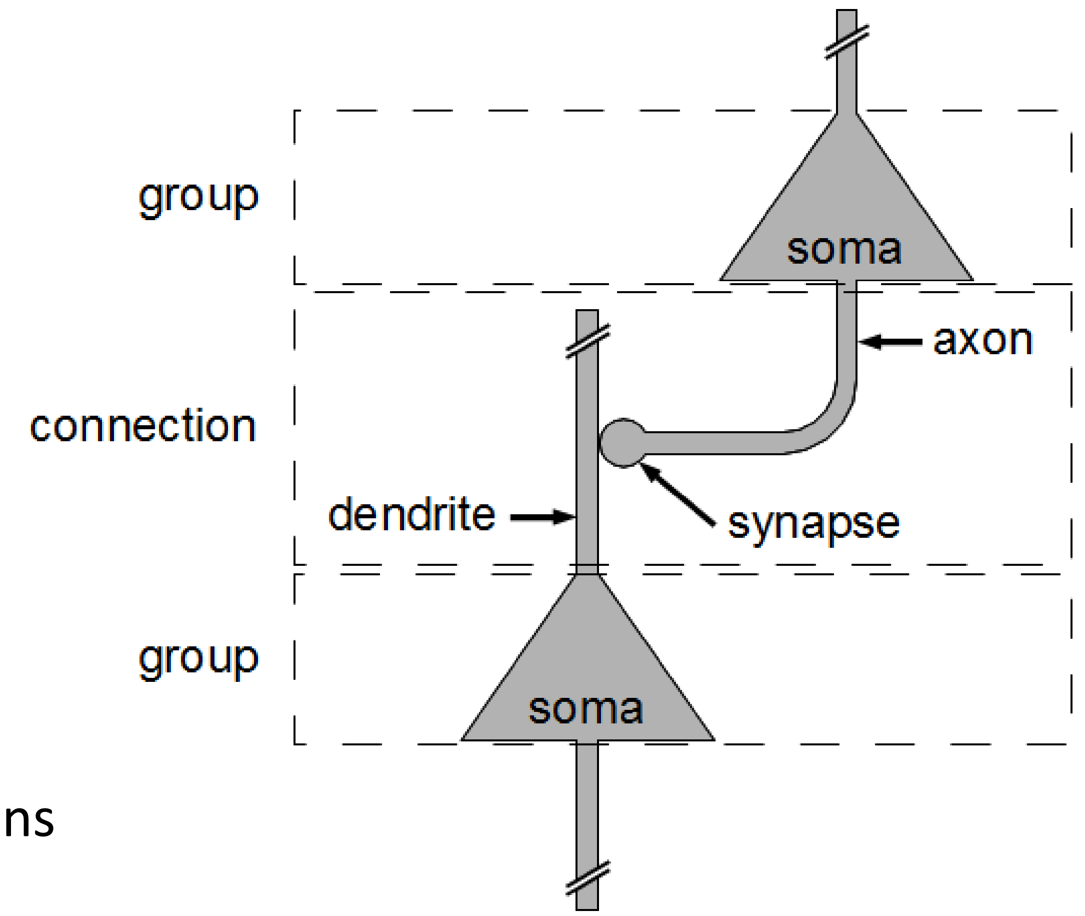


- A human brain consists of ~85 billion neurons
- Neurons outnumbered by synapses
  - Synapses per average neuron: 1,000 - 10,00
- Ratio of number of neurons vs. number of synapses
  - Biological neuronal networks draw their computational power from the large-scale integration of information in the connectivity between neurons
- Large scale neuronal models require heterogeneous connectivity patterns
- Providing a flexible, yet easy to use way to define connectivity is a cornerstone of iqr
- The definition needs to be compact and broad
- A generic way to define connectivity can be used to describe anatomical data in general

Ulysses Bernardet
<u.bernardet@aston.ac.uk>

Neurons and synapses

- Group
  - Specific aggregation of neurons of identical type
- Connection
  - Feed information from group to group
  - Aggregation of synapses of identical type
  - Definition of the arrangement of the synapses
- Connection framework
  - Deals with axon, synapse, and dendrite of the neurons
  - Computational element: Synapse
- Distinction between group and connection is not a biological fact
  - An abstraction within framework that allows easier approach to modeling biological systems



Ulysses Bernardet
<u.bernardet@aston.ac.uk>

Complete definitions

- Complete definition of a connection twofold
  - Definition of the update function of the synapse
  - Definition of the connectivity
- “Connectivity” refers to the spacial layout of the axons, synapses and dendrites
- Update function and connectivity are not as easily separable:
  - The delays in the dendrites are derived from the connectivity

Ulysses Bernardet
<u.bernardet@aston.ac.uk>

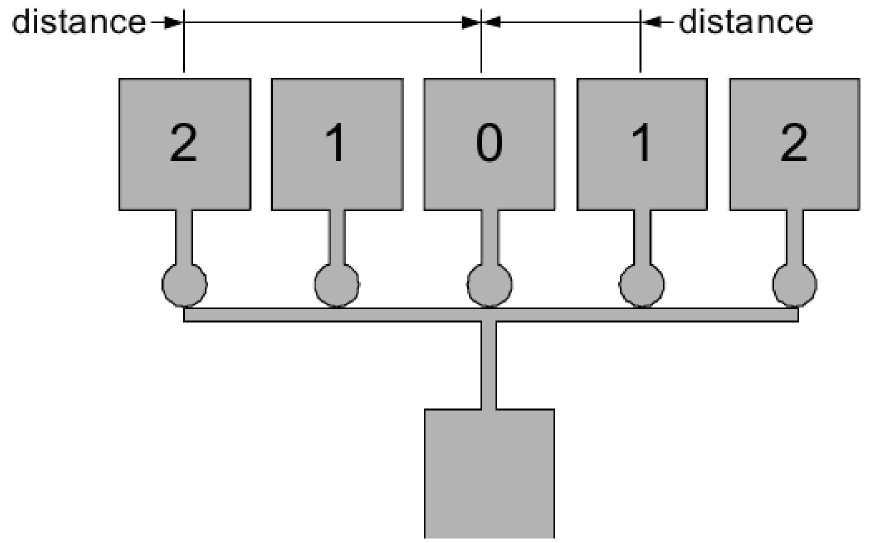
Connection assumptions

- Assumptions concerning connections
  - There is no delay in axons
  - The computation takes place in synapses
  - The transmission delay is dependent on the length of the dendrite
  - Any back-propagating signals are limited to the dendrite

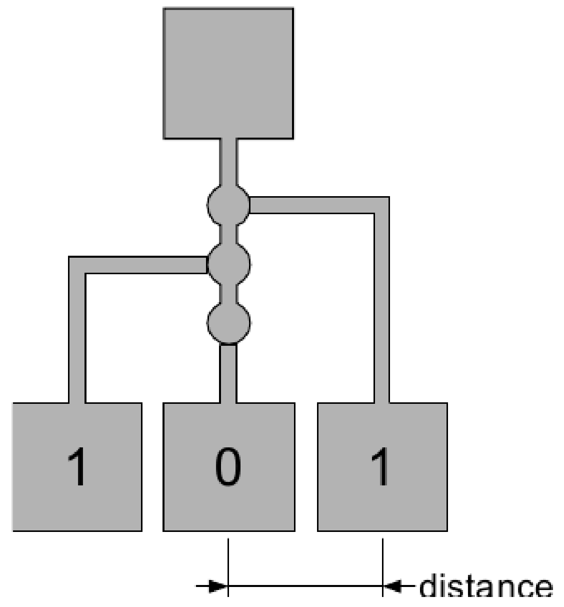
Ulysses Bernardet
<u.bernardet@aston.ac.uk>

Multiplicity: RF and PF

- Possible cases:
  - “receptive field” (RF): Many-to-one
  - “projective field” (PF): One-to-many



(a) Receptive field (RF)

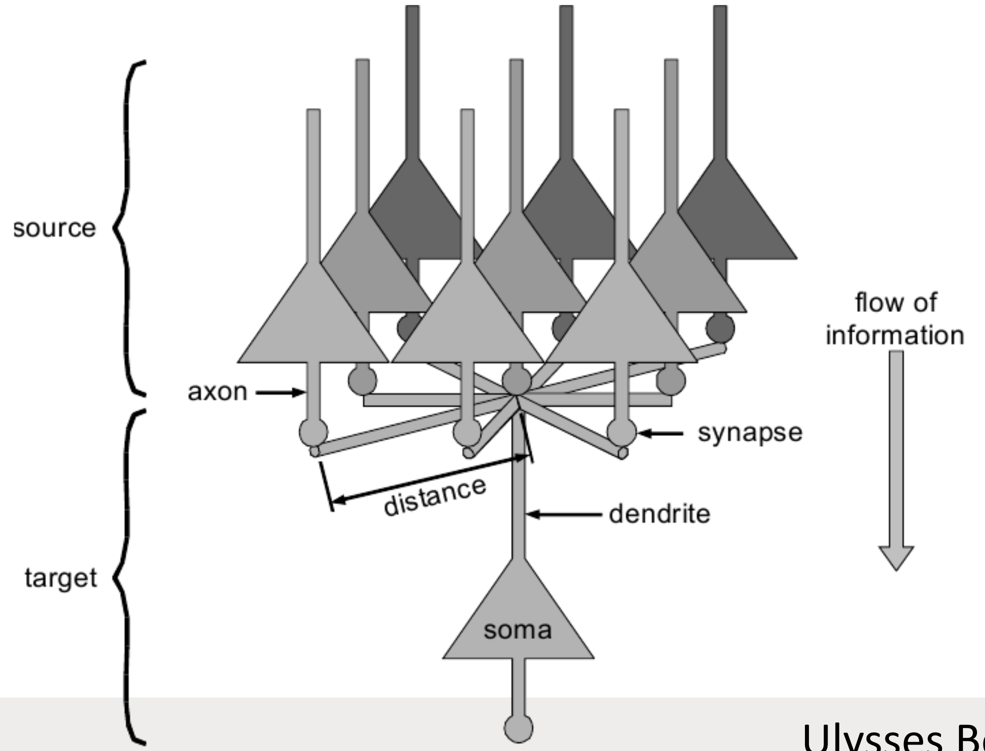


(b) Projective field (PF)

Ulysses Bernardet
<u.bernardet@aston.ac.uk>

A Single nexus

- Basis of computation of the delay: Distance
- Distance = eccentricity of a neuron
- Receptive field:
  - Eccentricity is defined by the position of the sending cell relative to the position of the one receiving cell.
- Projective field:
  - Eccentricity is defined with respect of the position of the multiple post-synaptic cells relative to the one pre-synaptic neuron



Ulysses Bernardet
<u.bernardet@aston.ac.uk>

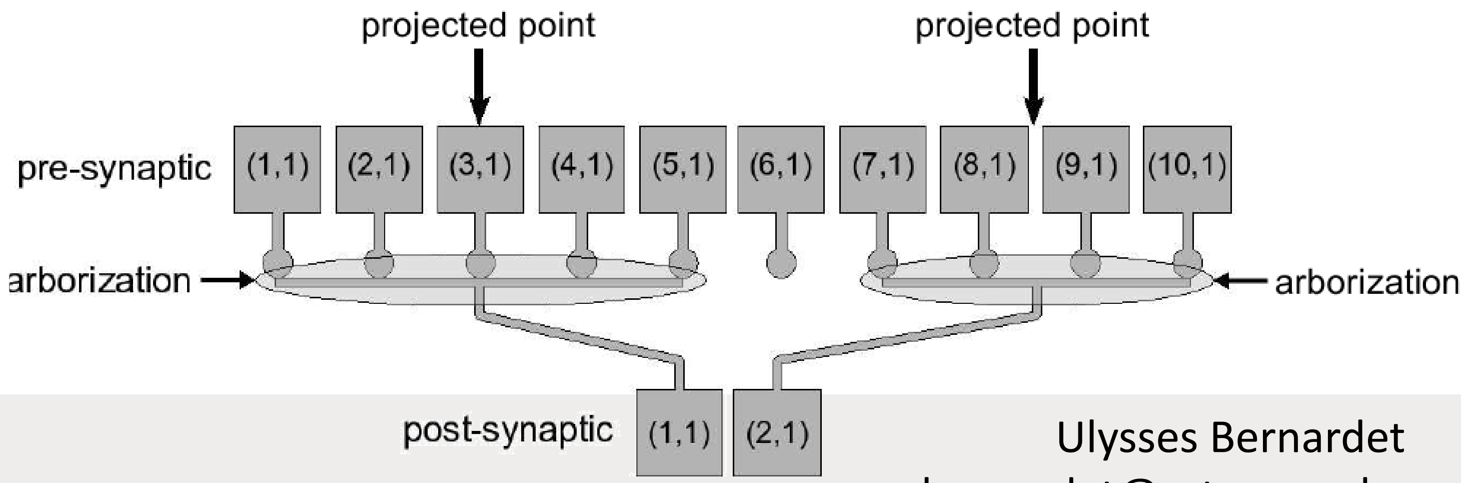
Defining the connectivity: Pattern + Arborization

Aston University

Ulysses Bernardet
<u.bernardet@aston.ac.uk>

Pattern and Arborization

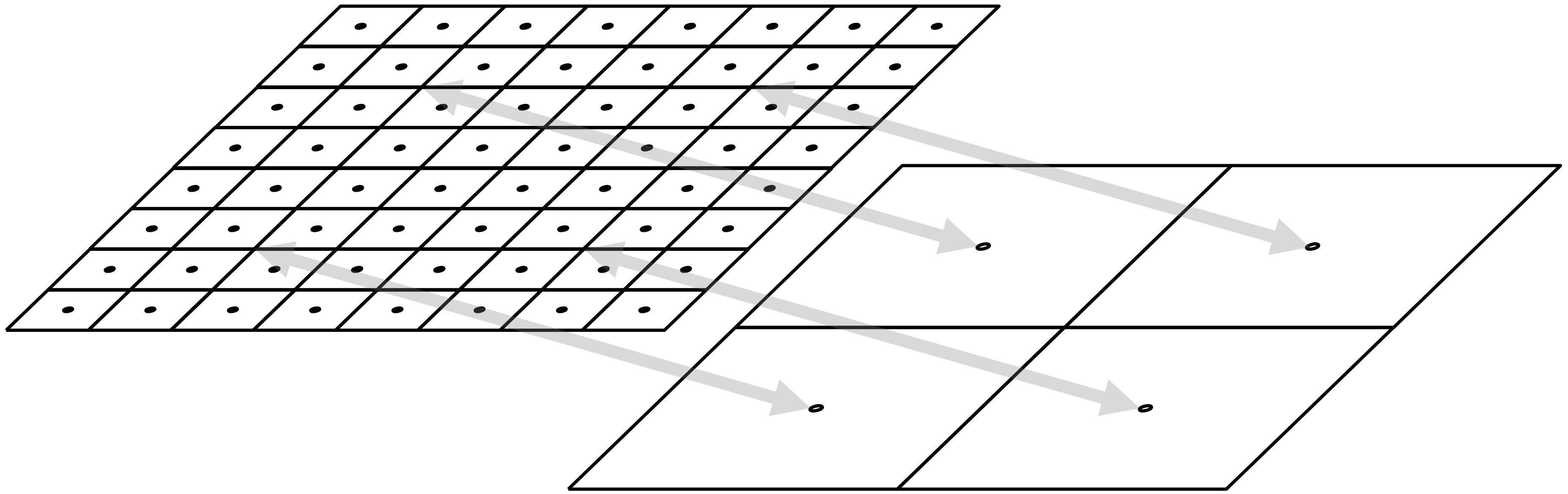
- Pattern
  - Defines pairs of points in the lattice of the pre- and post-synaptic group
  - Points are not neurons, but (x,y) coordinates
- Arborization
  - Defines the real neurons that send and receive input
  - Is applied to every pair of point defined by the pattern
  - Receptive field:
    - Applied to source group
  - Projective field:
    - Applied to target group



Aston University

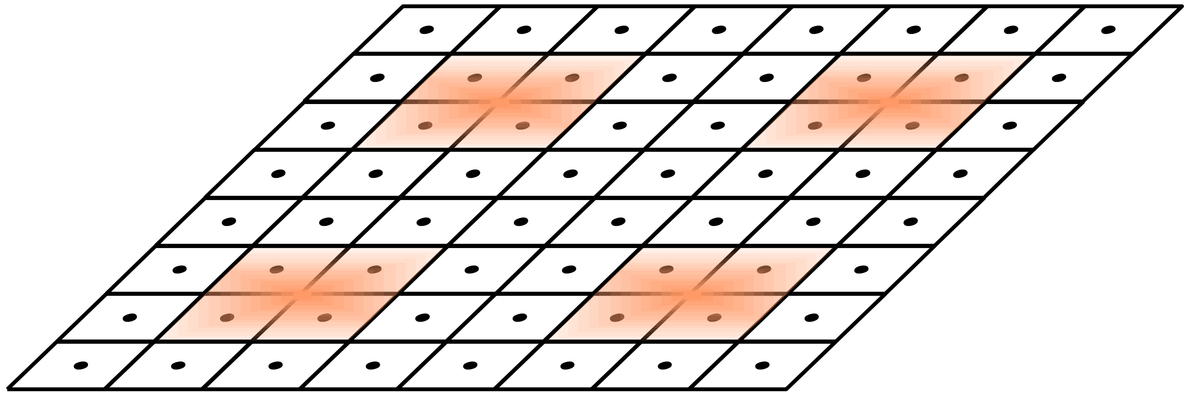
Ulysses Bernardet
<u.bernardet@aston.ac.uk>

Pattern



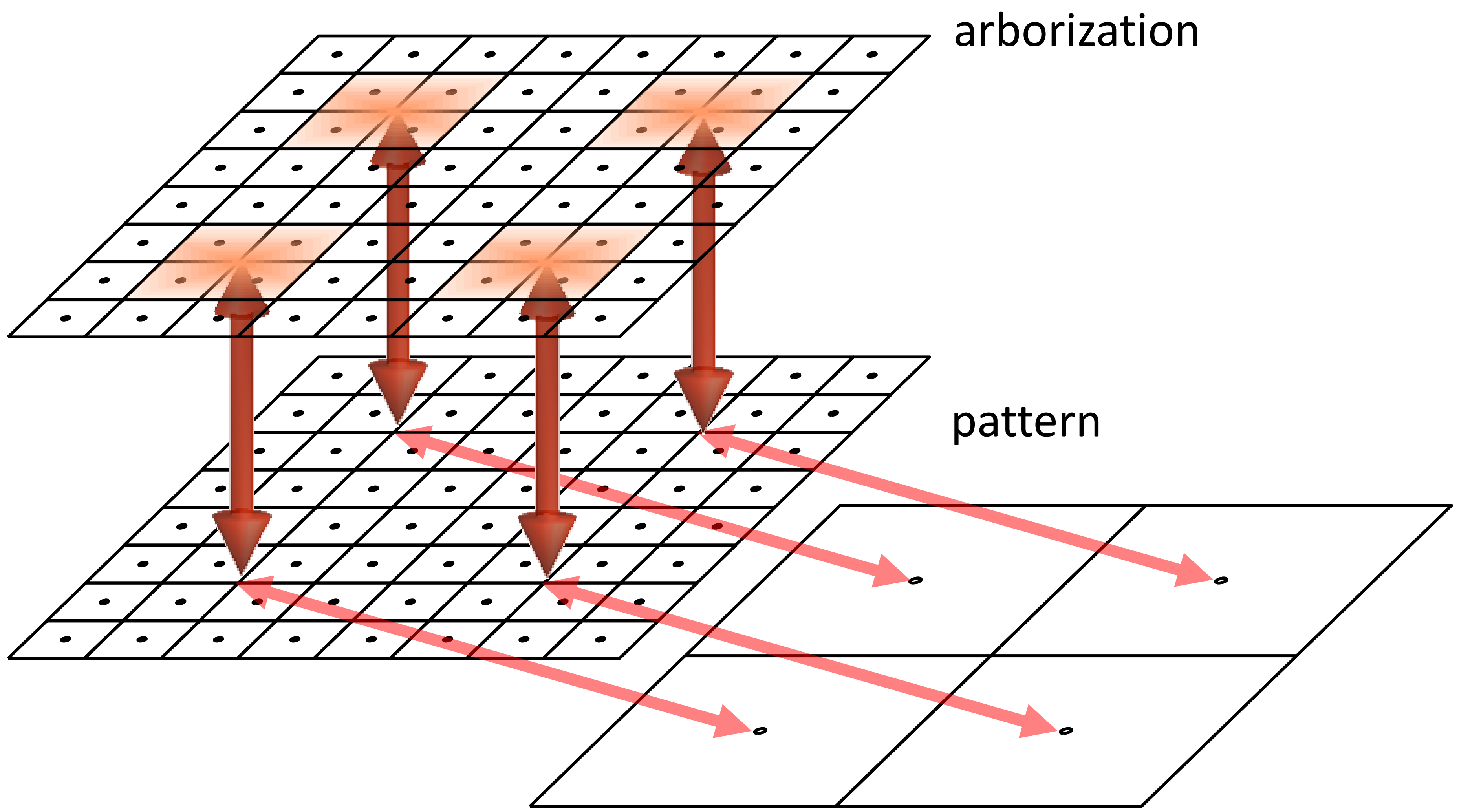
Ulysses Bernardet
<u.bernardet@aston.ac.uk>

Arborization



Ulysses Bernardet
<u.bernardet@aston.ac.uk>

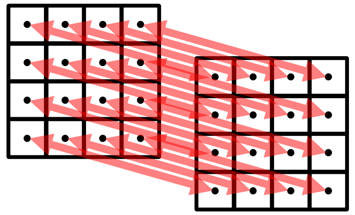
Pattern + Arborization



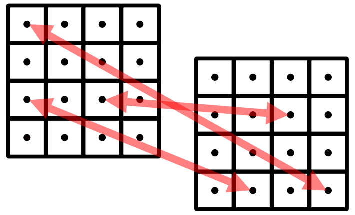
Ulysses Bernardet
<u.bernardet@aston.ac.uk>

Patterns and Arborizations

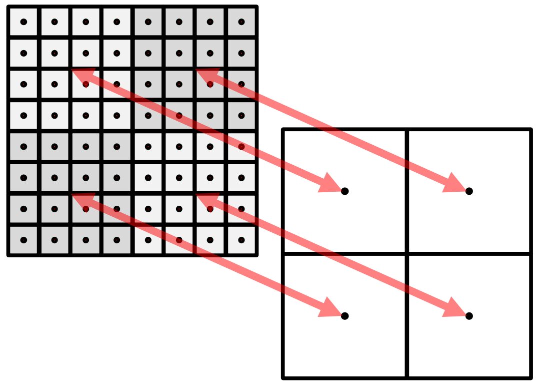
- ➔ for-each



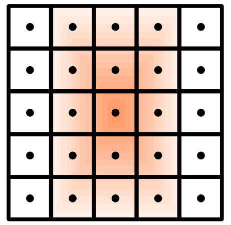
- ➔ tuples



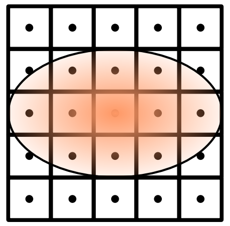
- ➔ mapped



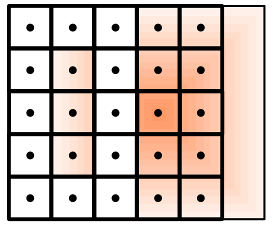
- ➔ rectangular



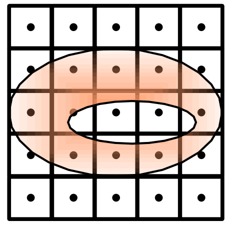
- ➔ elliptic



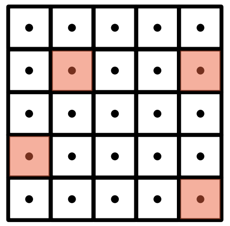
- ➔ rect. w. window



- ➔ elliptic w. window



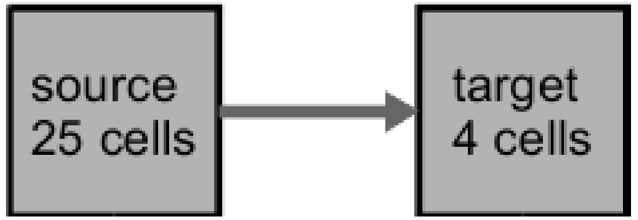
- ➔ random

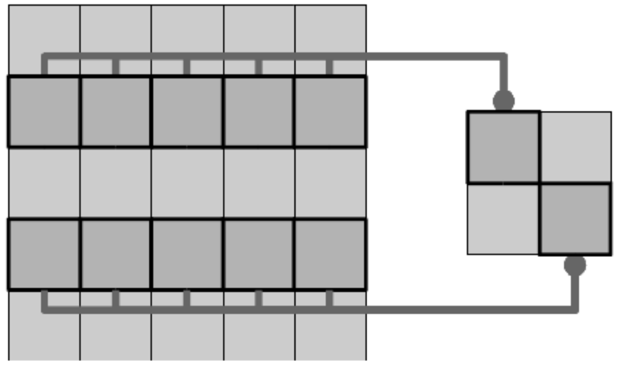


Ulysses Bernardet
<u.bernardet@aston.ac.uk>

Multiplicity: The axon-synapse-dendrite nexus

- Connections are represented at three different levels
  - Process
  - Group
  - Synapse
- Multiplicity
  - Single connection is an assembly of axon-synapse-dendrite nexuses
  - Single nexus in turn can connect several pre-synaptic neurons to one post-synaptic cell, or feed information from one pre-synaptic into multiple post-synaptic neurons
- One nexus can comprise of several axons, synapses, and dendrites

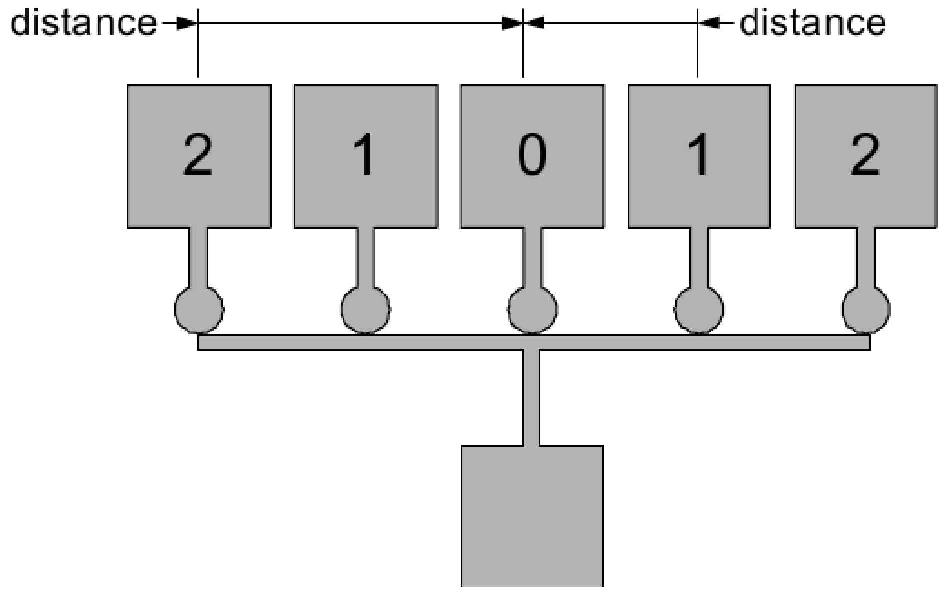




Ulysses Bernardet
<u.bernardet@aston.ac.uk>

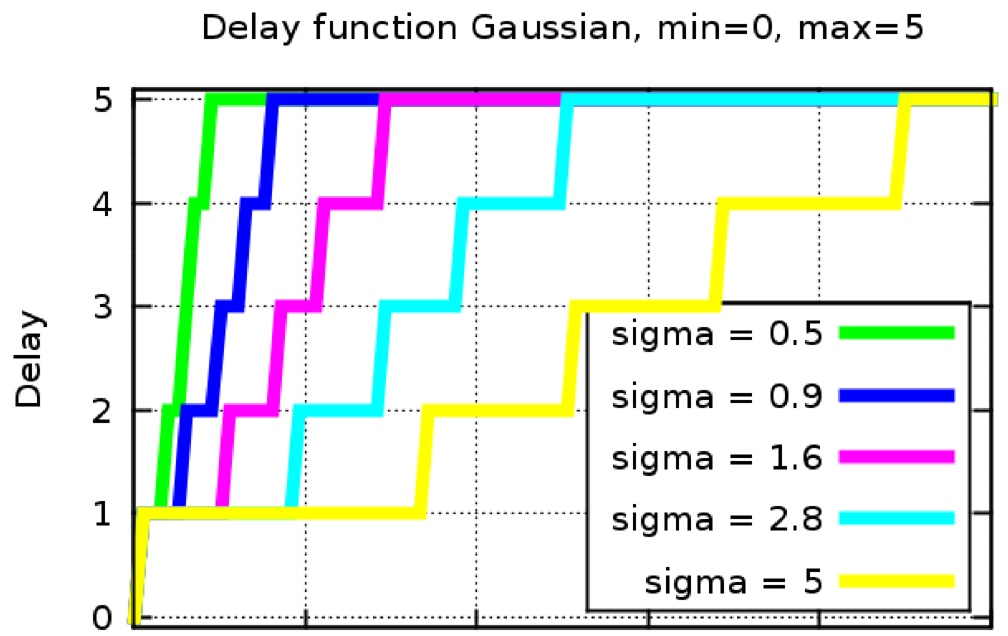
Delay function

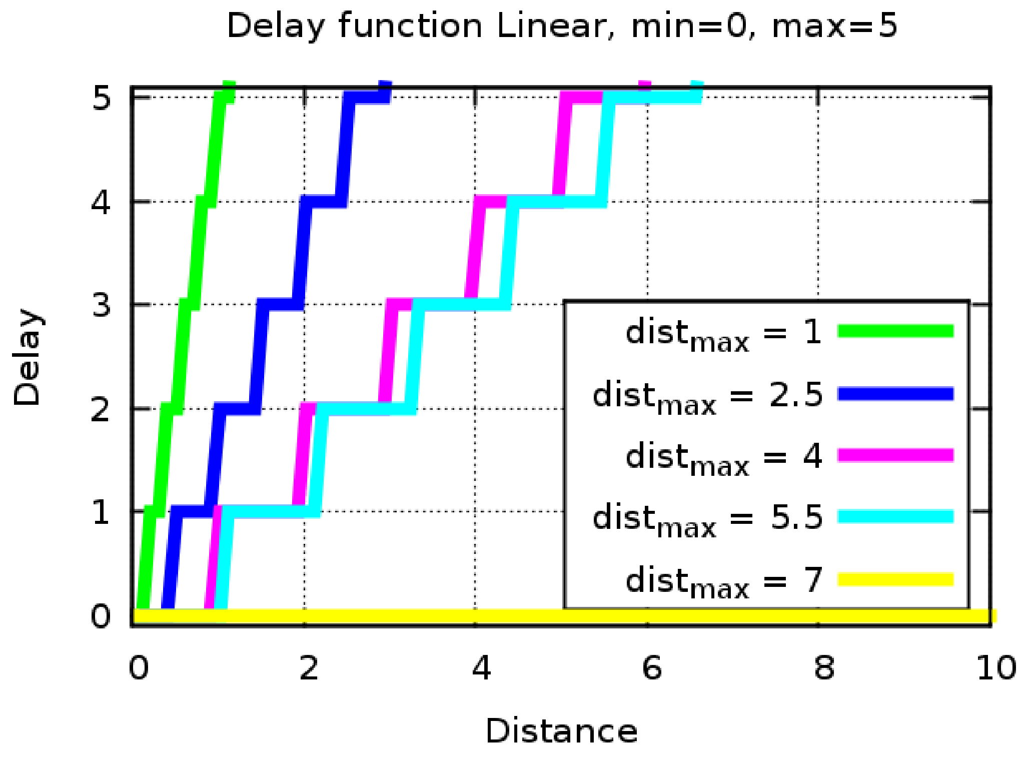
- Used to compute delay for each synapse belonging to an arborization
- In most delay functions the calculation is depending on the size of the arborization, i.e. height/width, or outer height/outer width respectively
- Types:
  - FunLinear
  - FunGaussian
  - FunBlock
  - FunRandom
  - FunUniform

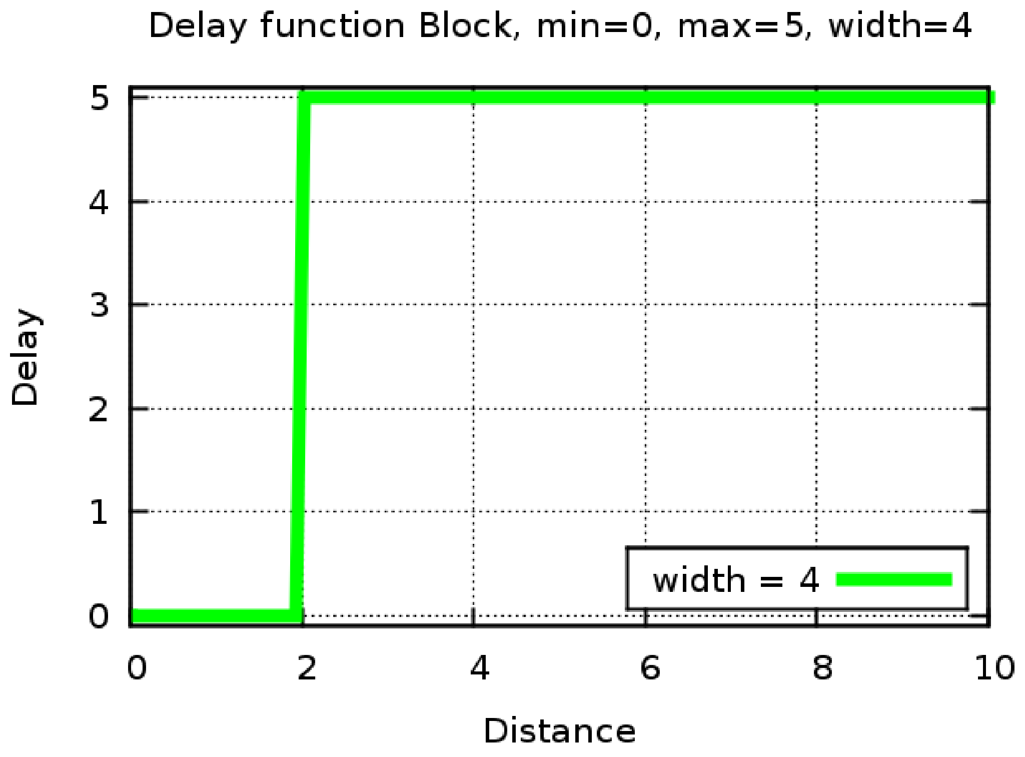


Ulysses Bernardet
<u.bernardet@aston.ac.uk>

Delay function





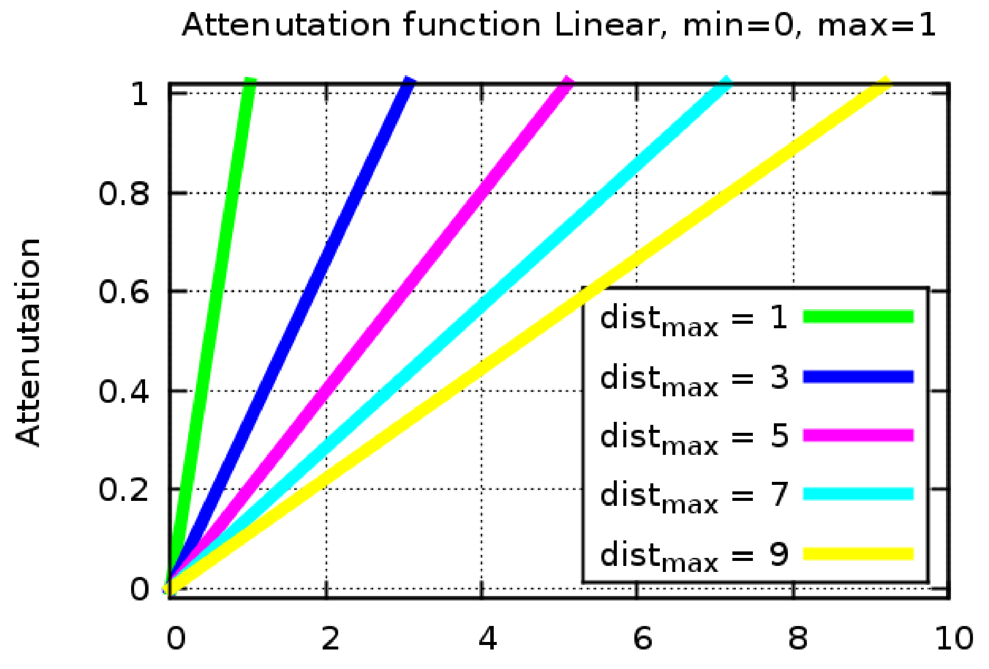


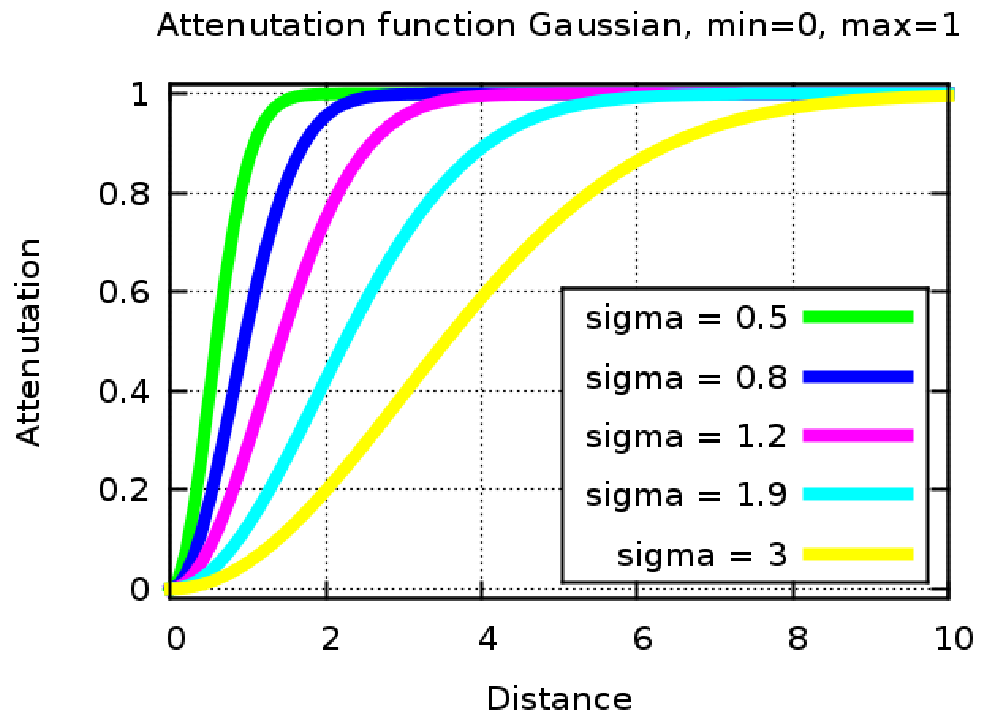
Aston University

Ulysses Bernardet
<u.bernardet@aston.ac.uk>

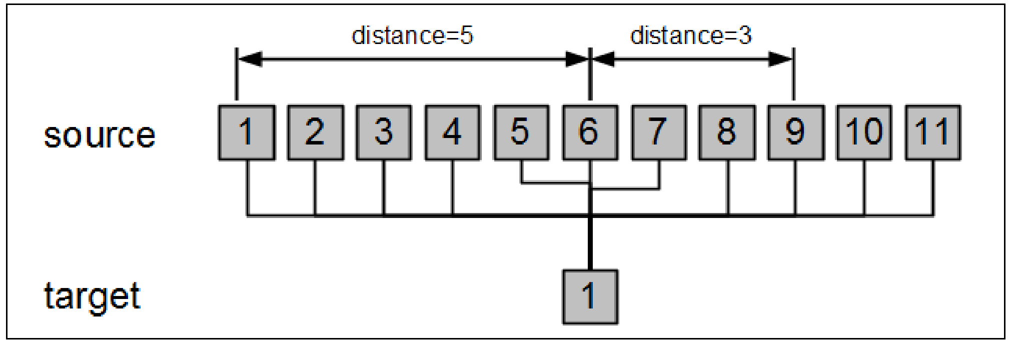
Attenuation function

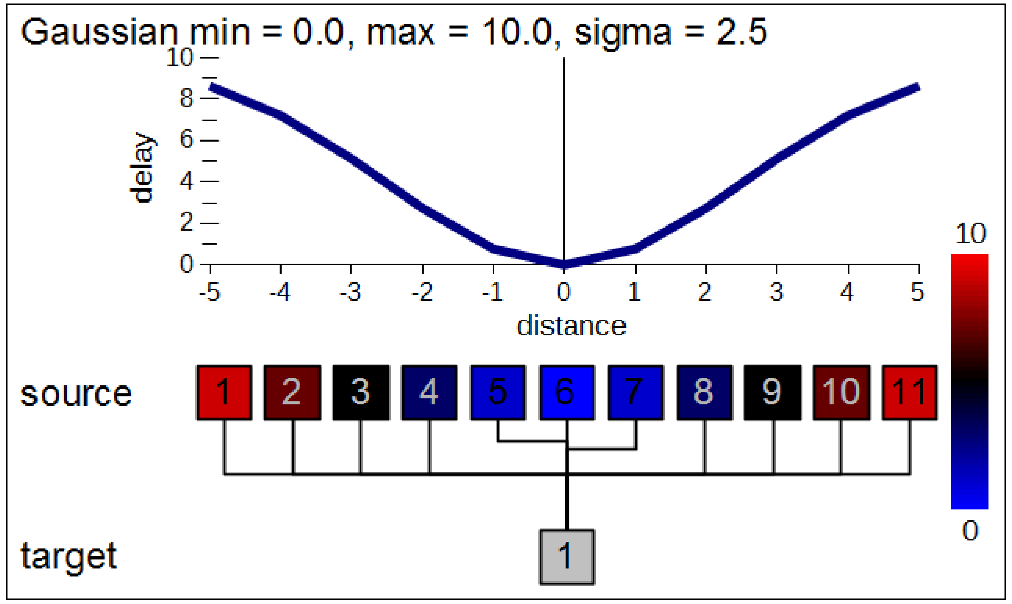
- Attenuation is the "damping" of the signal on the dendrite
- Attenuation -> inverse of strength

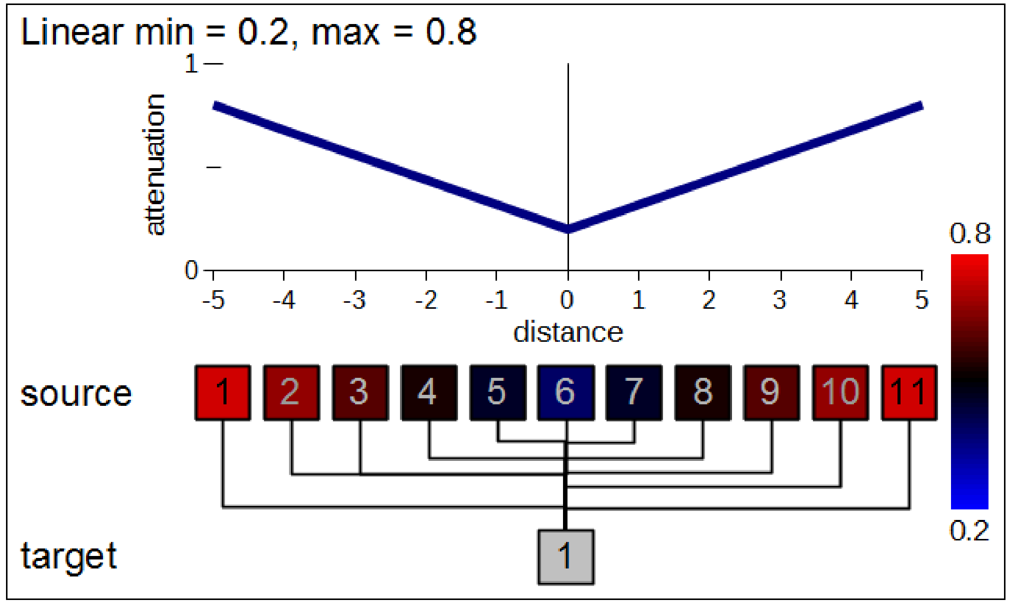




Attenuation and delay example







Aston University

Ulysses Bernardet
<u.bernardet@aston.ac.uk>

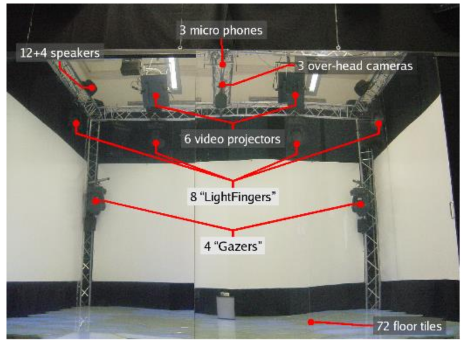
Hands-on

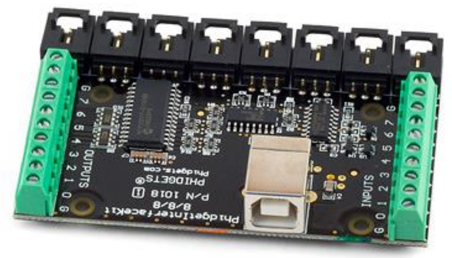
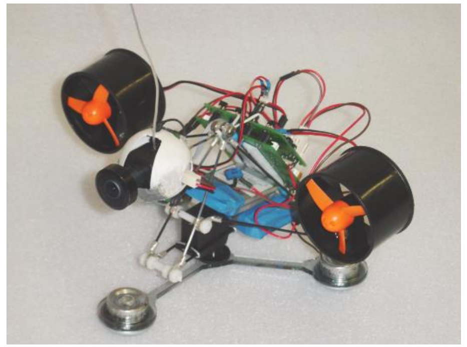
iqr Tutorial 02-Cell Types, Synapses & Run-time State Manipulation

Working with Modules

iqr Modules

- A Module is a plug-in to exchange data with an external entity
  - Sensor (camera, microphone, etc.)
  - Actuator (robot, motor)
  - Algorithm
- Modules read and write the activity of groups
- One process can have only one single module

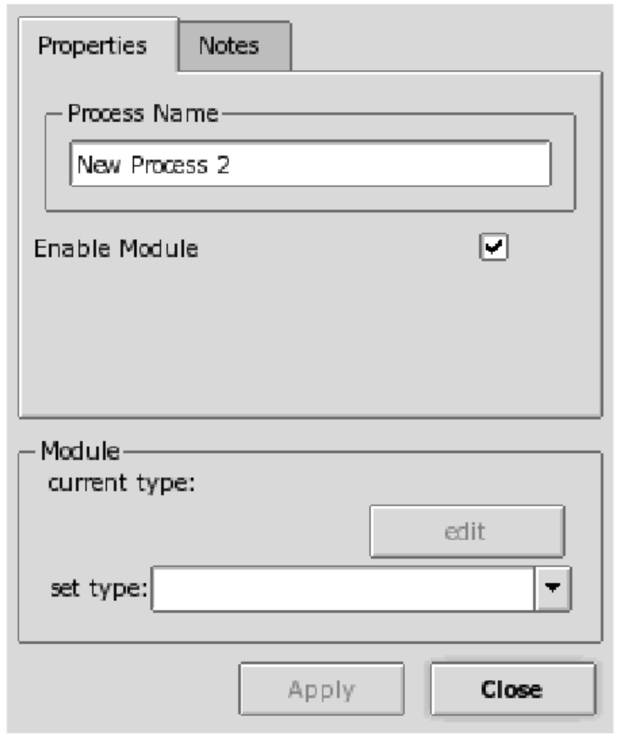


Ulysses Bernardet
<u.bernardet@aston.ac.uk>

iqr Modules

- Assigning a module to a group
  - Process properties dialog
  - Select the module type
- Edit button in the Module frame → Module property dialog
- “Groups to Module” and “Module to Groups” -> contain the allocation information between modules and groups
- Disable the update of a module during the simulation:
  - un-check the Enable Module check-box in the process properties dialog

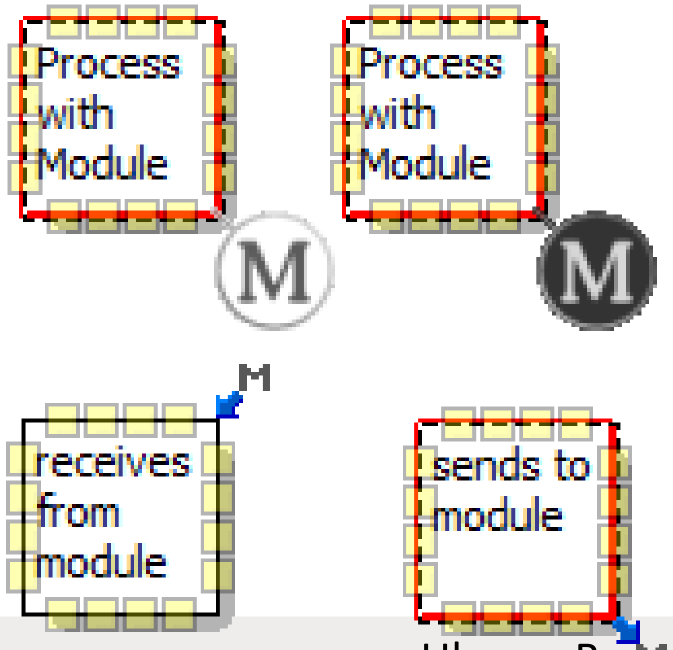

Properties | Notes

Process Name: New Process 2

Enable Module ✓

Module
current type:
edit
set type:

Apply | Close



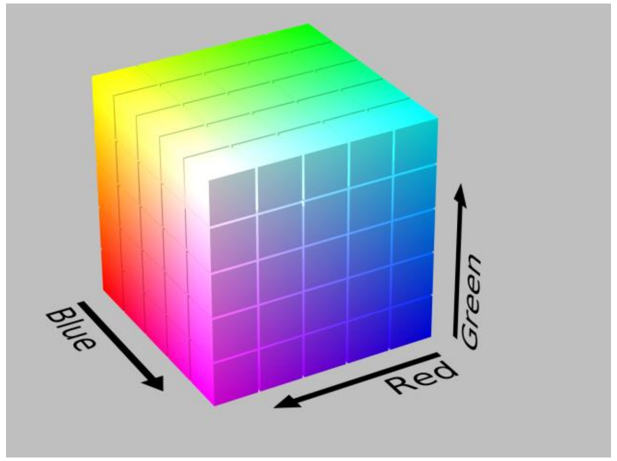
Aston University

Ulysses Bernardet
<u.bernardet@aston.ac.uk>

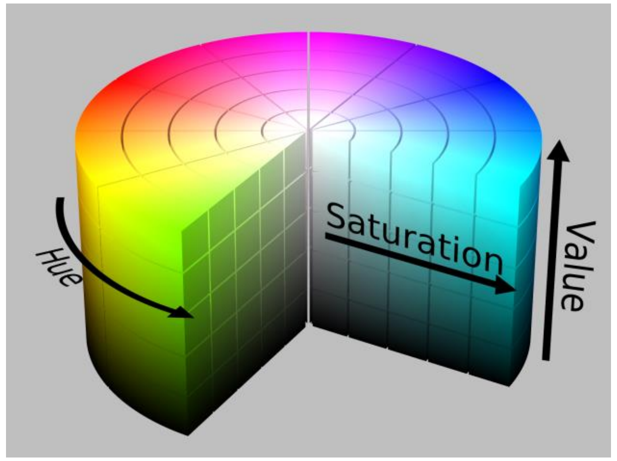
Video modules

- Module parameters
  - Monochrome
  - HSV color space

RGB color space



HSV color space



- On Linux you can use “xawtv” to check video
- Windows:
  - Module uses first camera found
  - Use virtual webcam software (e.g. Manycam) to use multiple cameras

Ulysses Bernardet
<u.bernardet@aston.ac.uk>

Other modules

- Joystick Module
- Phidgets
- Wii remote
- Speech synthesizer
- Network communication (TCP/IP, UDP, OSC)
- More at: http://sourceforge.net/projects/iqr-extension/

Ulysses Bernardet
<u.bernardet@aston.ac.uk>

Hands-on

IQR TUTORIAL 05-CAMERA-BASIC INTERFACING

IQR TUTORIAL 05-CAMERA-NEGATIVE IMAGE

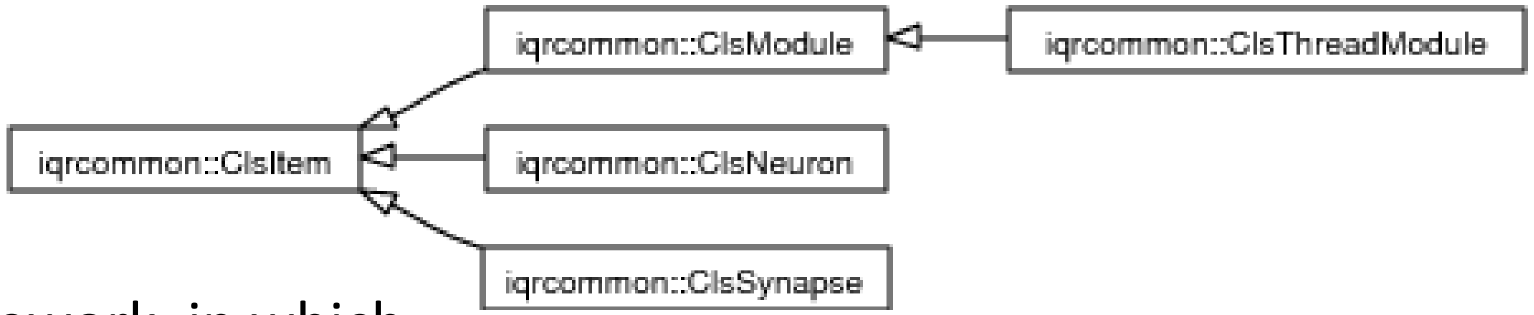
AND THERE'S MORE…

- User-defined types
- Advanced features

Ulysses Bernardet
<u.bernardet@aston.ac.uk>

Classes for (User-defined) types

- iqr does not make a distinction between types that are defined by the user, and those that come with the installation; both are implemented in the same way
- Base-classes for all three types (neurons, synapses, modules) derived from ClsItem
- The classes derived from ClsItem are parent class for the specific types
  - E.g. a specific neuron type will be derived from ClsNeuron



- Inheritance schema defines the framework, in which
  - parameters are defined
  - data is represented and accessed
  - input, output, and feedback is added
- All types defined in namespace iqrcommon

Ulysses Bernardet
<u.bernardet@aston.ac.uk>

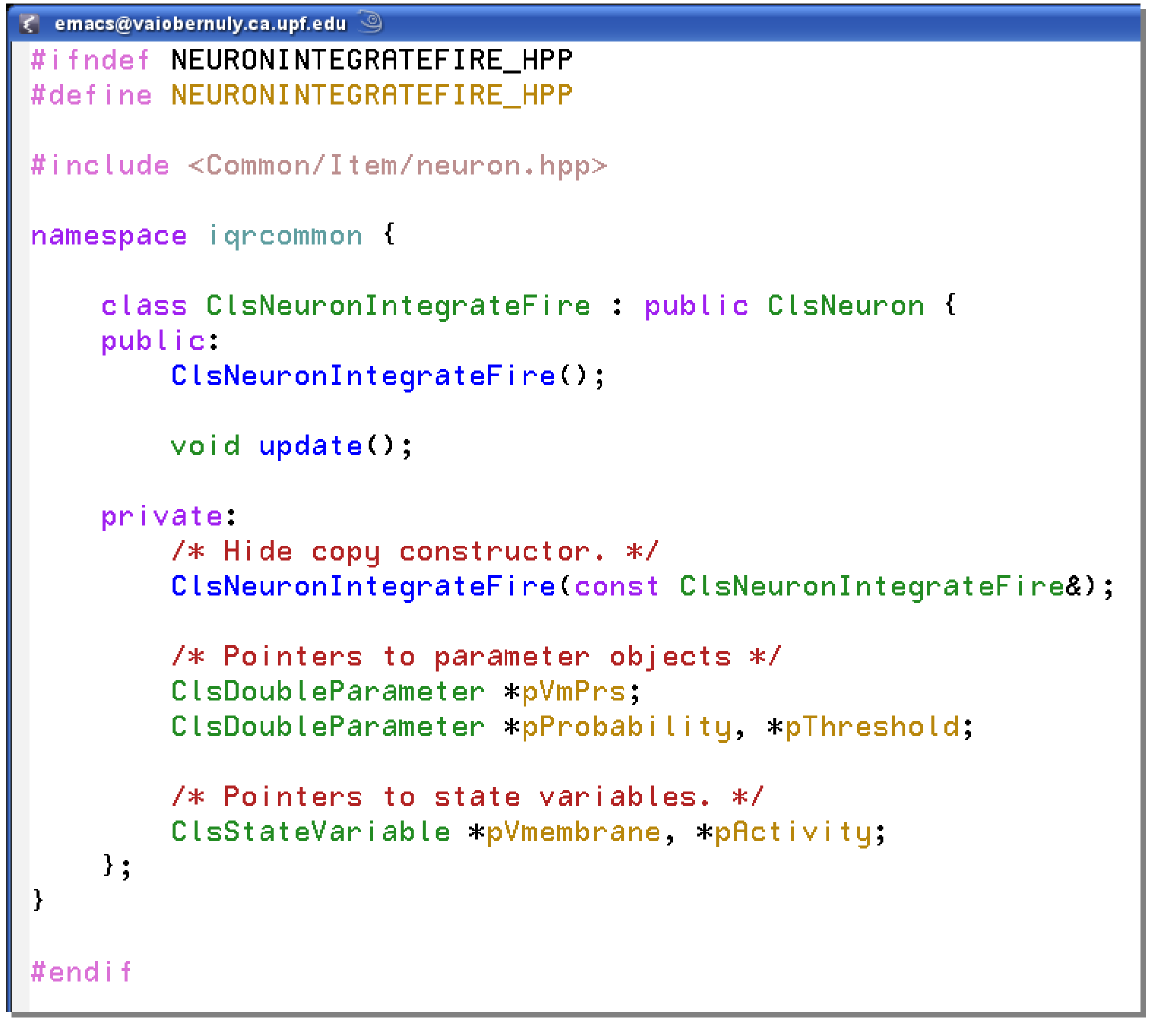
Programming Neurons, Synapses, Modules

- Base classes
- Class Methods (initialize, update, ...)
- Parameters
- State arrays
- Tricks about accessing history
- ...etc

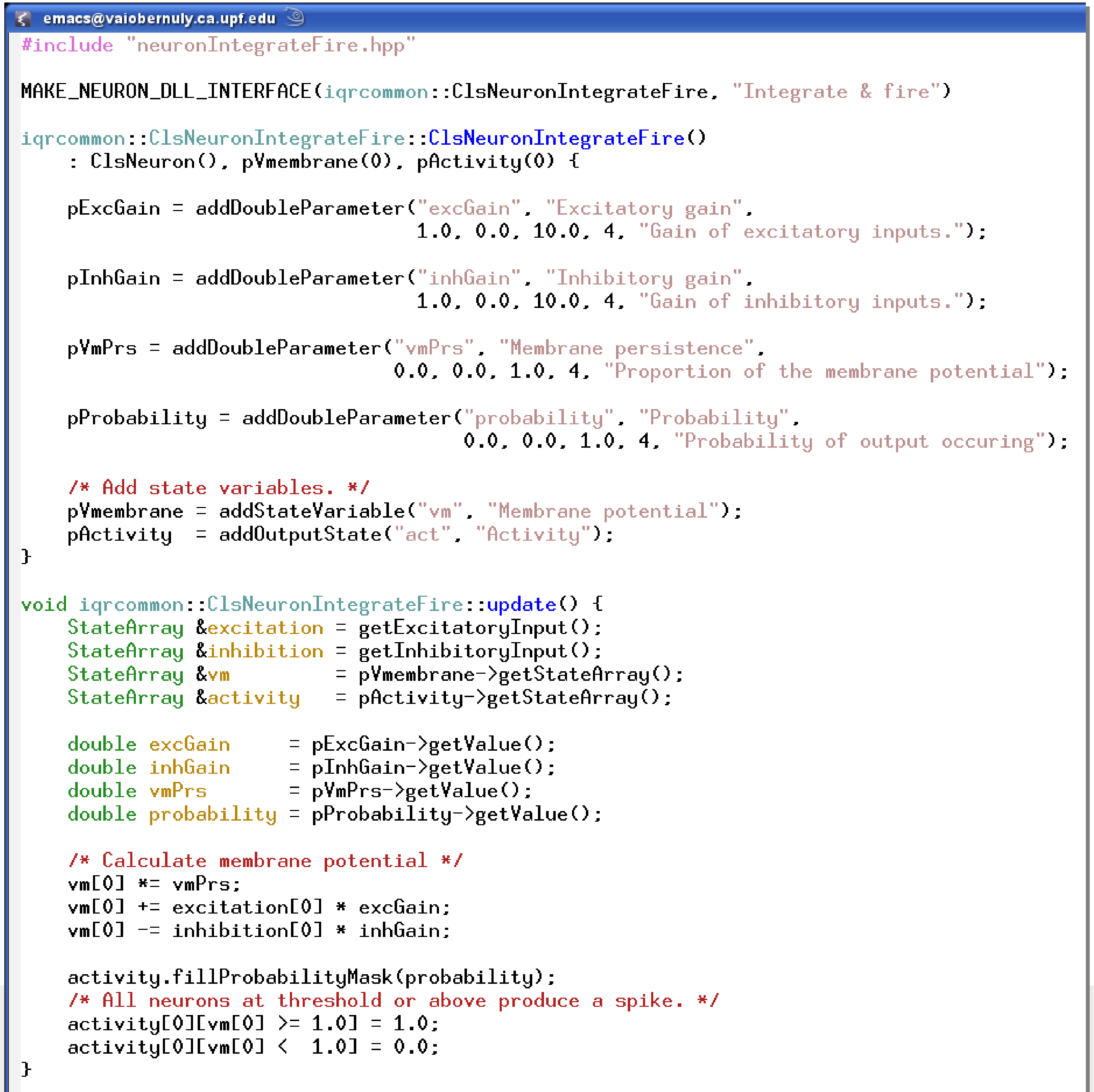
iqrUserdefinedTypes.pdf

Ulysses Bernardet
<u.bernardet@aston.ac.uk>

Neuron example



```
#ifndef NEURONINTEGRATEFIRE_HPP
#define NEURONINTEGRATEFIRE_HPP

#include <Common/Item/neuron.hpp>

namespace iqrcommon {

    class ClsNeuronIntegrateFire : public ClsNeuron {
    public:
        ClsNeuronIntegrateFire();

        void update();

    private:
        /* Hide copy constructor. */
        ClsNeuronIntegrateFire(const ClsNeuronIntegrateFire&);

        /* Pointers to parameter objects */
        ClsDoubleParameter *pVmPrs;
        ClsDoubleParameter *pProbability, *pThreshold;

        /* Pointers to state variables. */
        ClsStateVariable *pVmembrane, *pActivity;
    };
}

#endif
```

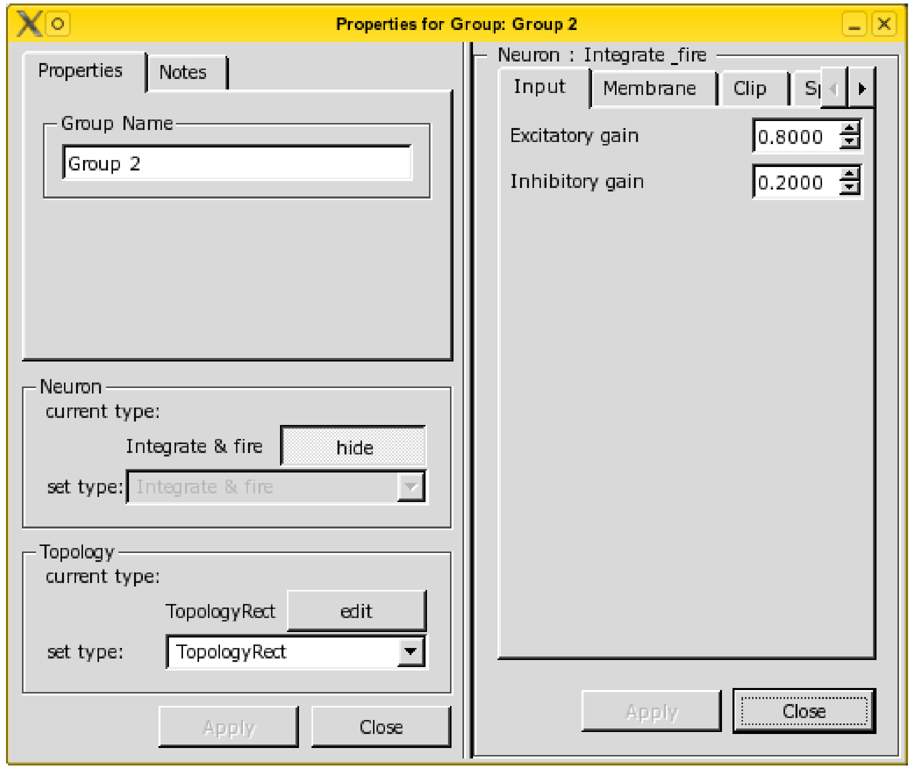


```
#include "neuronIntegrateFire.hpp"

MAKE_NEURON_DLL_INTERFACE(iqrcommon::ClsNeuronIntegrateFire, "Integrate & fire")

iqrcommon::ClsNeuronIntegrateFire::ClsNeuronIntegrateFire()
    : ClsNeuron(), pVmembrane(0), pActivity(0) {

    pExcGain = addDoubleParameter("excGain", "Excitatory gain",
                                  1.0, 0.0, 10.0, 4, "Gain of excitatory inputs.");

    pInhGain = addDoubleParameter("inhGain", "Inhibitory gain",
                                  1.0, 0.0, 10.0, 4, "Gain of inhibitory inputs.");

    pVmPrs = addDoubleParameter("vmPrs", "Membrane persistence",
                                0.0, 0.0, 1.0, 4, "Proportion of the membrane potential");

    pProbability = addDoubleParameter("probability", "Probability",
                                      0.0, 0.0, 1.0, 4, "Probability of output occuring");

    /* Add state variables. */
    pVmembrane = addStateVariable("vm", "Membrane potential");
    pActivity  = addOutputState("act", "Activity");
}

void iqrcommon::ClsNeuronIntegrateFire::update() {
    StateArray &excitation = getExcitatoryInput();
    StateArray &inhibition = getInhibitoryInput();
    StateArray &vm         = pVmembrane->getStateArray();
    StateArray &activity   = pActivity->getStateArray();

    double excGain     = pExcGain->getValue();
    double inhGain     = pInhGain->getValue();
    double vmPrs       = pVmPrs->getValue();
    double probability = pProbability->getValue();

    /* Calculate membrane potential */
    vm[0] *= vmPrs;
    vm[0] += excitation[0] * excGain;
    vm[0] -= inhibition[0] * inhGain;

    activity.fillProbabilityMask(probability);
    /* All neurons at threshold or above produce a spike. */
    activity[0][vm[0] >= 1.0] = 1.0;
    activity[0][vm[0] <  1.0] = 0.0;
}
```

Aston University

Neuron example

```
ClsDoubleParameter *pExcGain =
                addDoubleParameter("excGain",
                "Excitatory gain",
                1.0, 0.0, 1.0, 3,
                "Gain of excitatory inputs.\n"
                "The inputs are summed before\n"
                "being multiplied by this gain.",
                "Input");
```


Properties for Group: Group 2

Properties | Notes

Group Name: Group 2

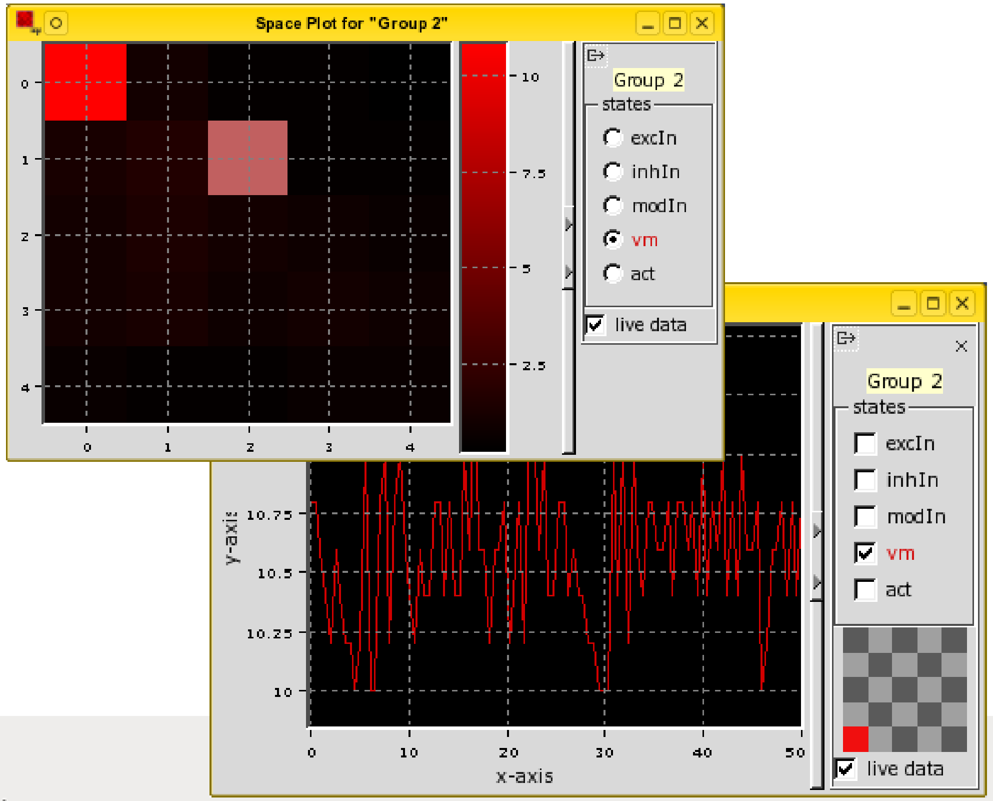
Neuron
current type: Integrate & fire
set type: Integrate & fire

Topology
current type: TopologyRect
set type: TopologyRect

Neuron : Integrate _fire

Input | Membrane | Clip | Sp

Excitatory gain: 0.8000
Inhibitory gain: 0.2000


Space Plot for "Group 2"

Group 2
states
excIn
inhIn
modIn
vm
act
live data

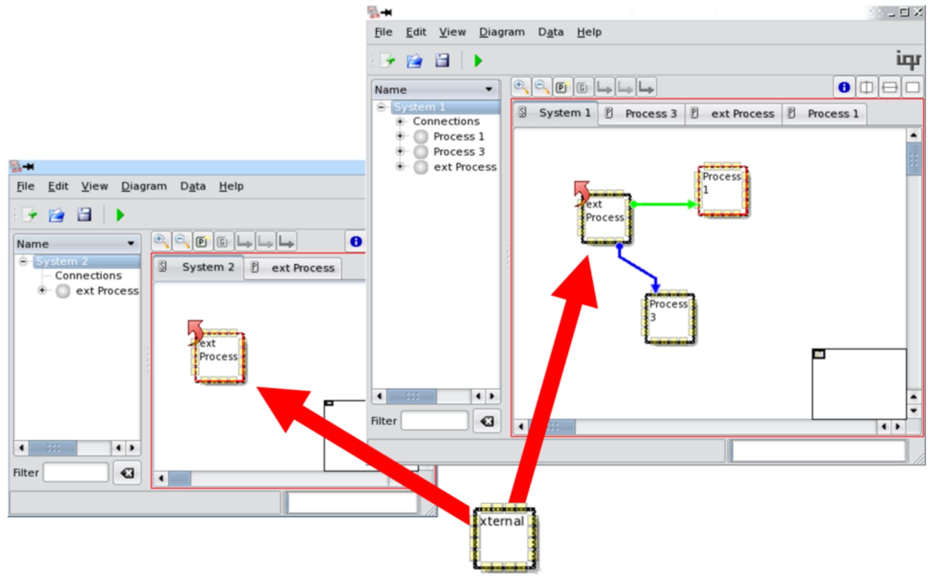
```
ClsStateVariable *pVmembrane =
                addStateVariable("vm",
                "Membrane potential");
```

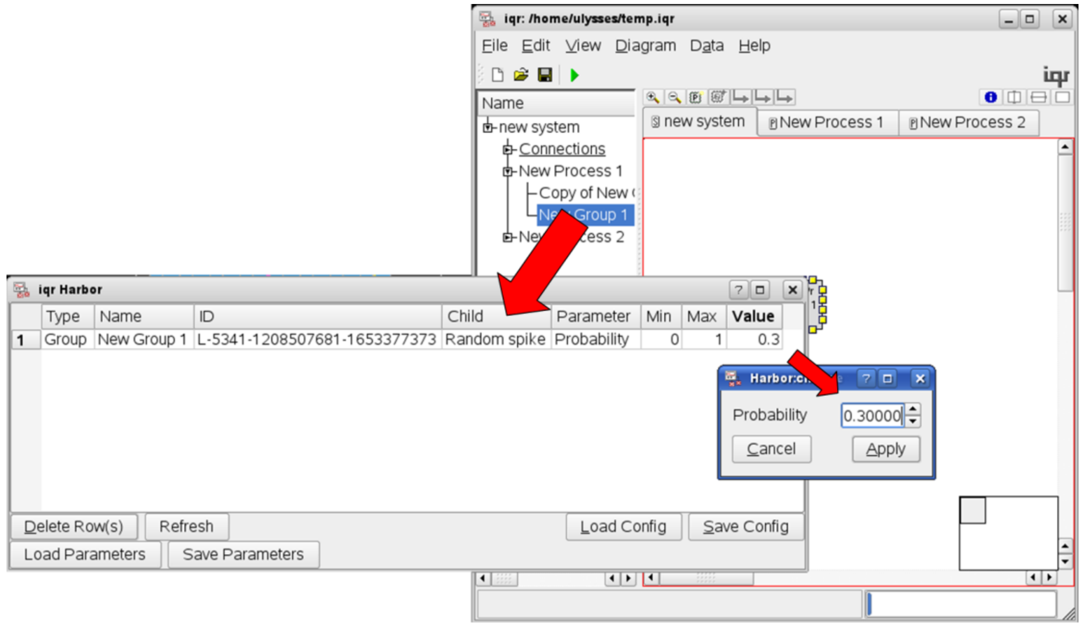
Aston University

Ulysses Bernardet
<u.bernardet@aston.ac.uk>

Advanced features

- External Processes
  - Process + connections defined in separate file
  - Import or link-in Process definition
- Harbor
  - Collect items
  - Change items in single place
  - Make items available to Module
- Optimization
  - Access to system elements from Modules
- Remote control
  - UDP and GUI





	Type	Name	ID	Child	Parameter	Min	Max	Value
1	Group	New Group 1	L-5341-1208507681-1653377373	Random spike	Probability	0	1	0.3

Probability 0.30000

Ulysses Bernardet
<u.bernardet@aston.ac.uk>