COSIVINA

Compose, Simulate, and Visualize Neurodynamic Architectures

Summer School Neural Dynamics for Cognitive Robotics

September 2013

Sebastian Schneegans

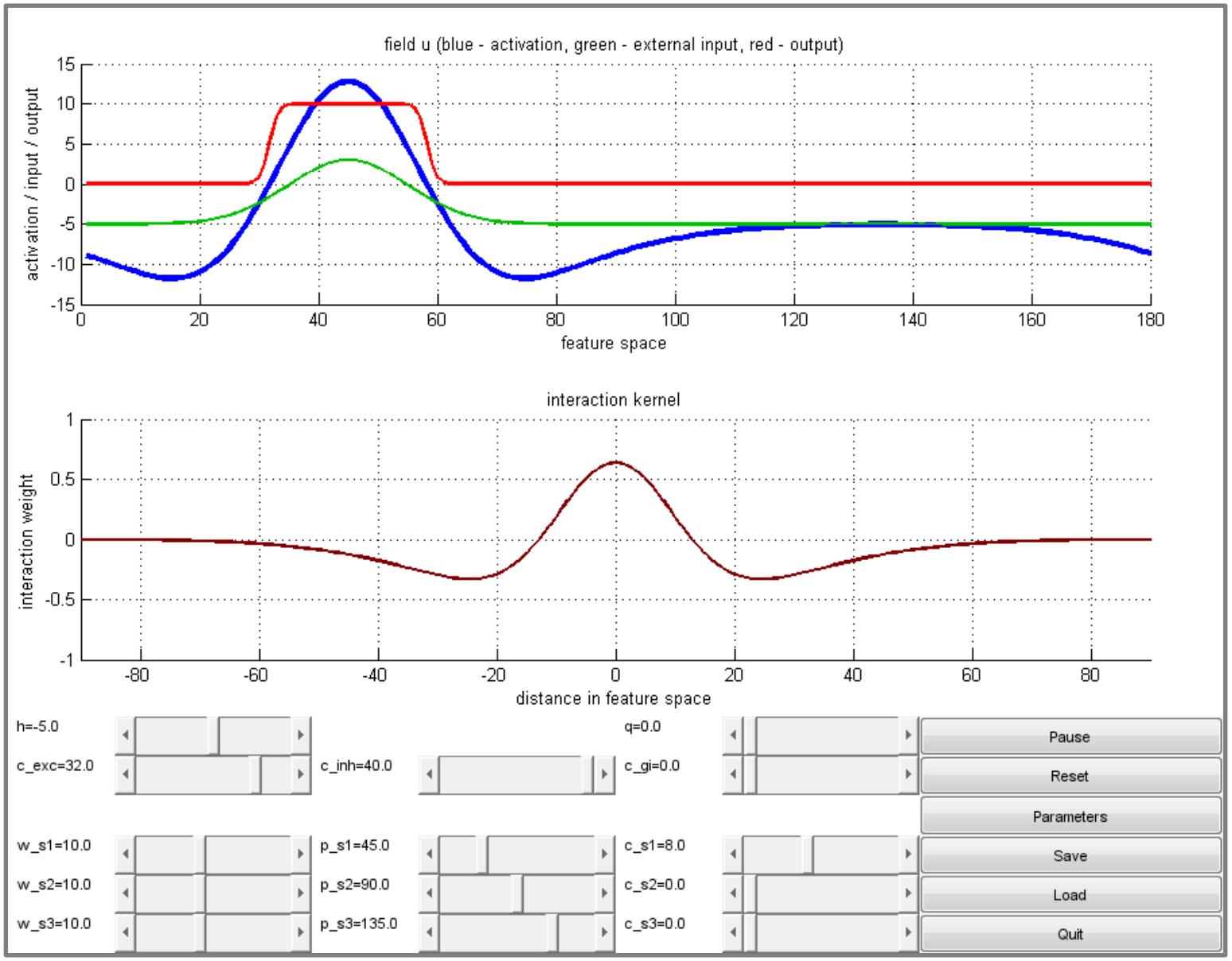
sebastian.schneegans@ini.rub.de

Motivation

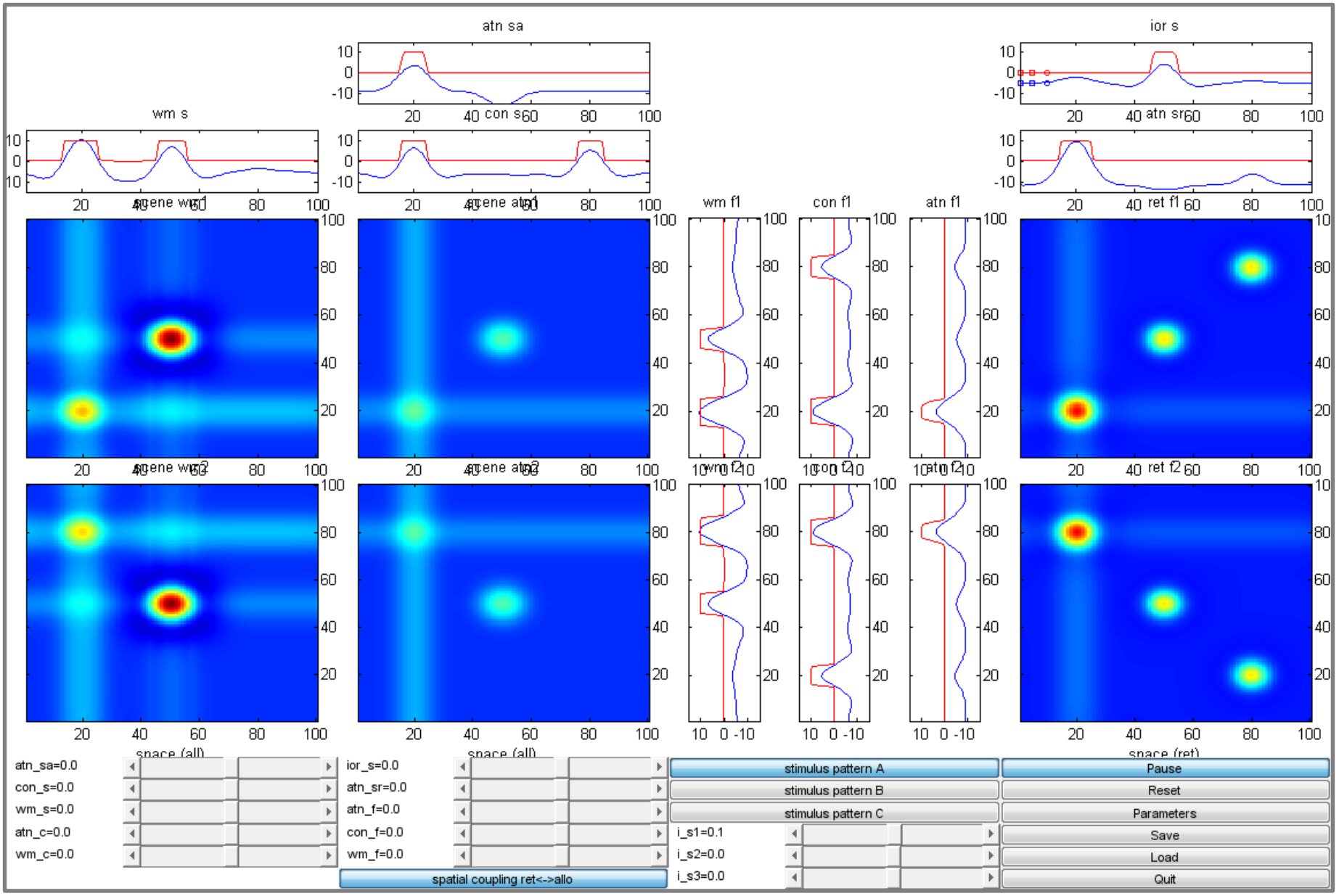
- quickly create and simulate DNF architectures
- create online visualizations and interactive controls of model parameters without hassle
- standardize implementations for different architectures and usages

➢ class library for Matlab

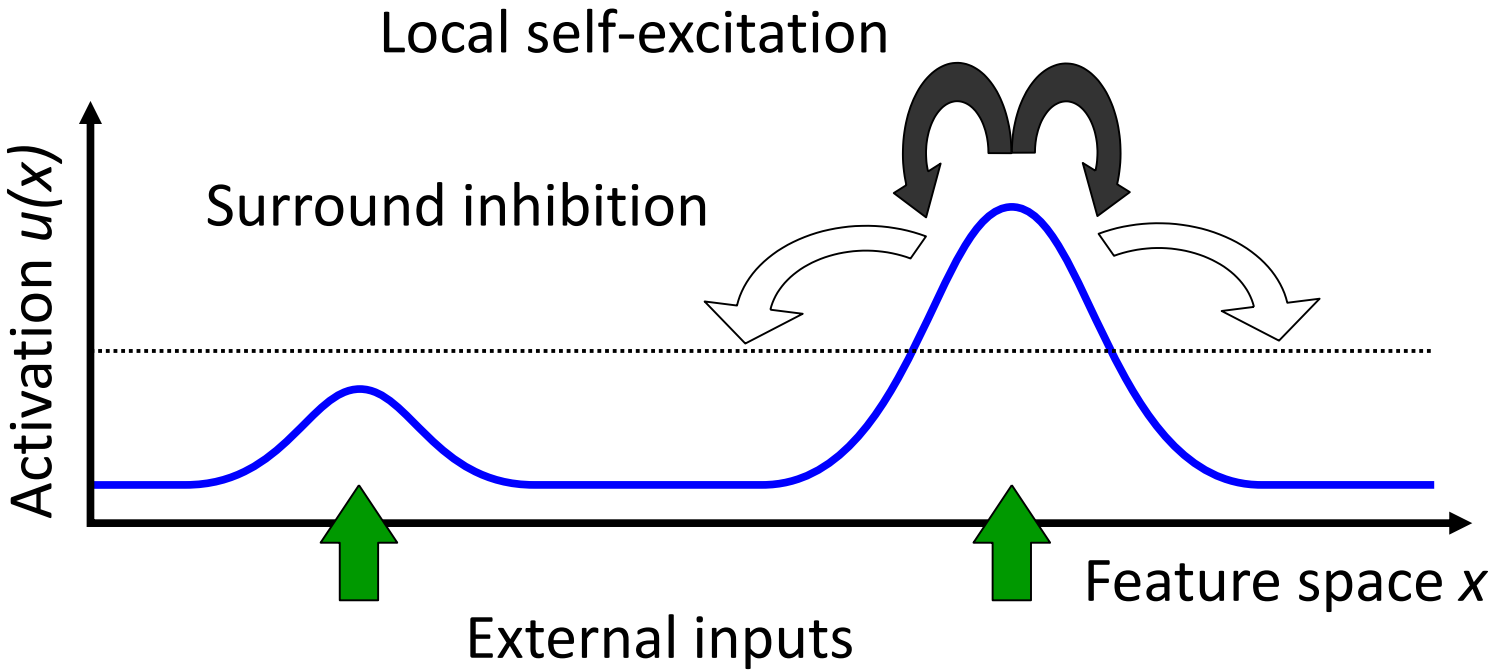
Examples


field u (blue - activation, green - external input, red - output)
activation / input / output
feature space
interaction kernel
interaction weight
distance in feature space
h=-5.0
c_exc=32.0
c_inh=40.0
q=0.0
c_gi=0.0
w_s1=10.0
w_s2=10.0
w_s3=10.0
p_s1=45.0
p_s2=90.0
p_s3=135.0
c_s1=8.0
c_s2=0.0
c_s3=0.0
Pause
Reset
Parameters
Save
Load
Quit

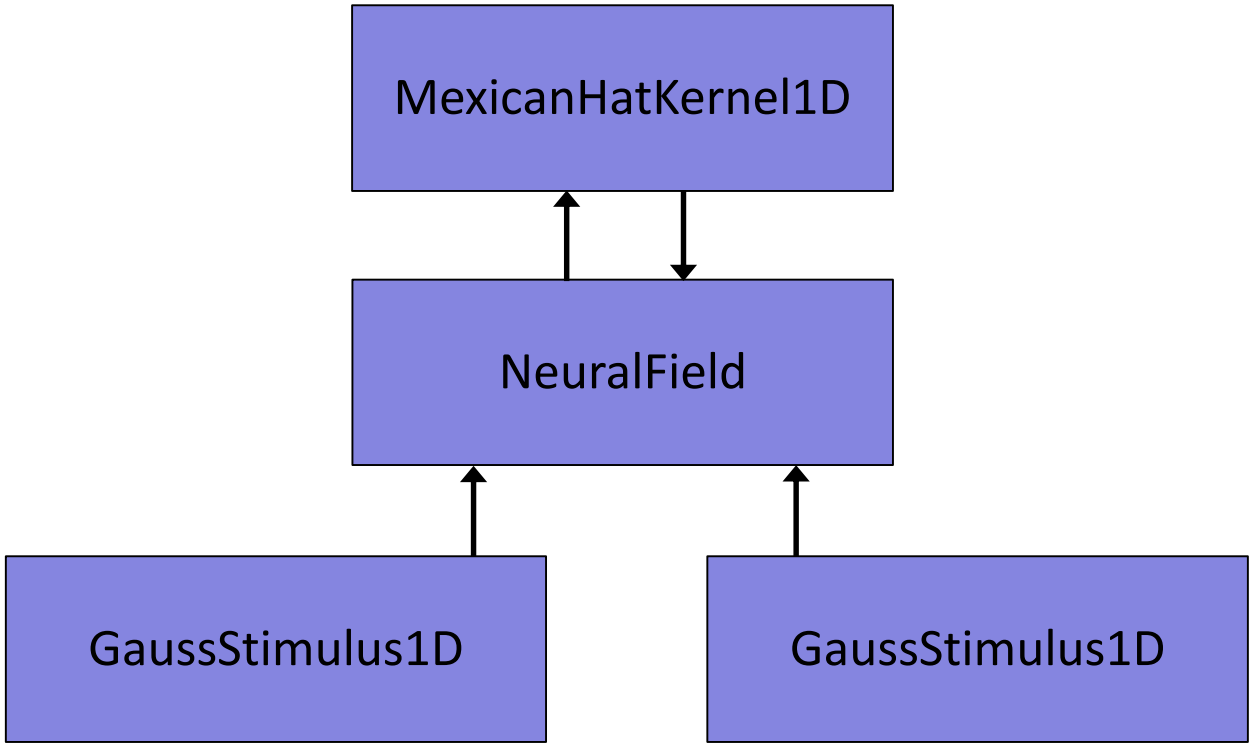
Examples


atn sa
ior s
wm s
con s
atn sr
scene wm f1
scene atn f1
wm f1
con f1
atn f1
ret f1
scene wm f2
scene atn f2
wm f2
con f2
atn f2
ret f2
space (all)
space (all)
space (ret)
atn_sa=0.0
con_s=0.0
wm_s=0.0
atn_c=0.0
wm_c=0.0
ior_s=0.0
atn_sr=0.0
atn_f=0.0
con_f=0.0
wm_f=0.0
spatial coupling ret<->allo
stimulus pattern A
stimulus pattern B
stimulus pattern C
i_s1=0.1
i_s2=0.0
i_s3=0.0
Pause
Reset
Parameters
Save
Load
Quit

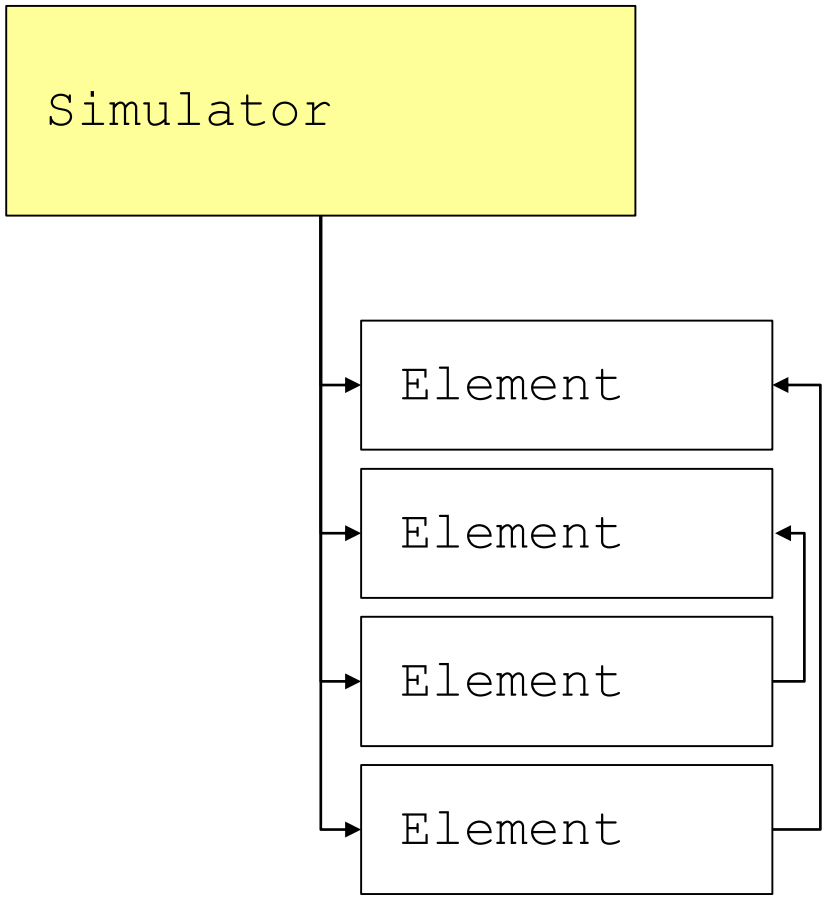
Structure for DNF Architectures



Structure for DNF Architectures



Structure for DNF Architectures



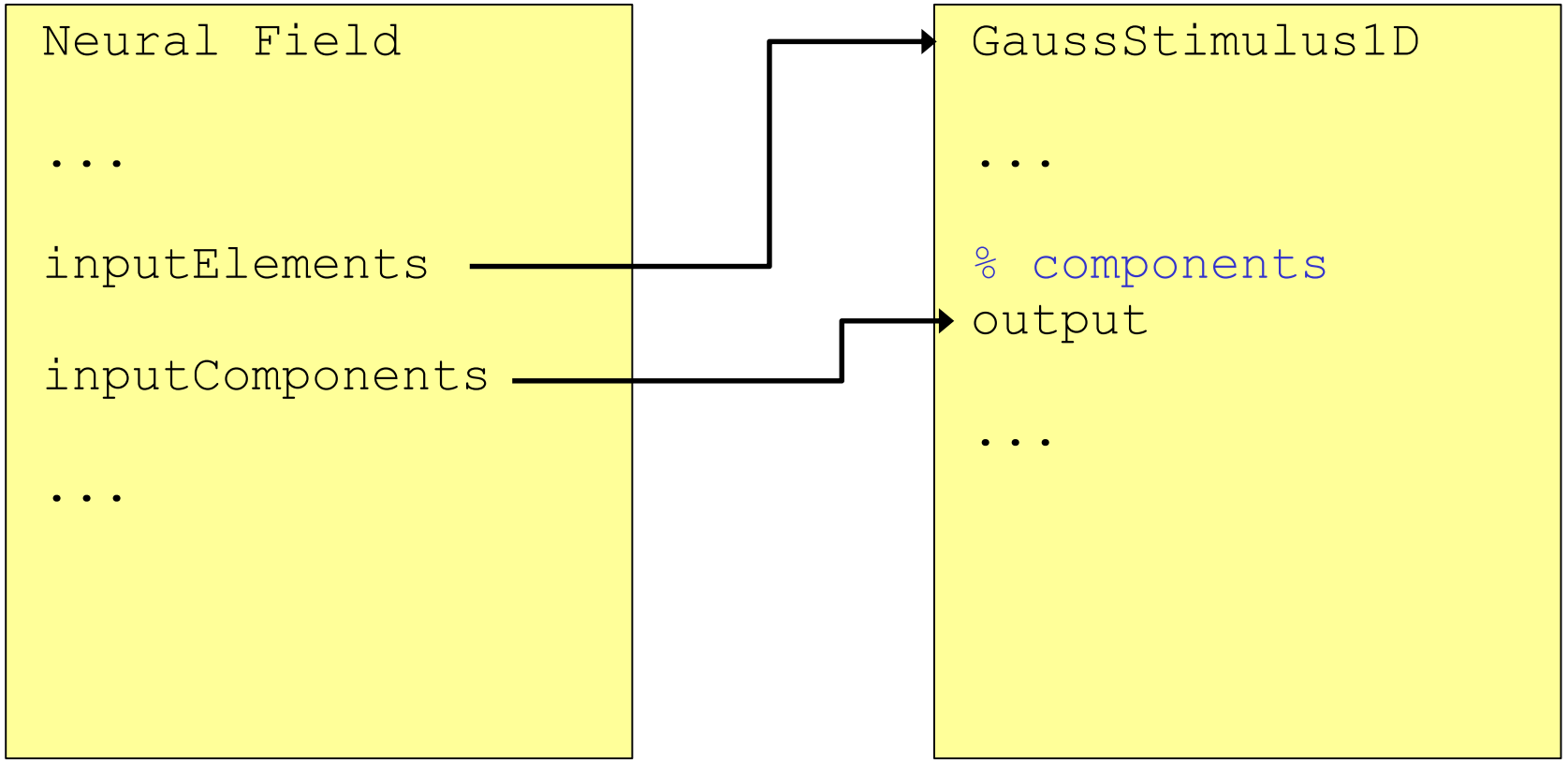
Simulator class manages elements, provides functions to initialize and simulate whole architecture

Elements implemented as handle classes, can have handles to other elements that provide input to them

Element Classes

```
NeuralField < Element

properties

label = 'field u'

% meta-data
parameters
components

% inputs
nInputs
inputElements % handles to other elements
inputComponents

% parameters
size = [1, 100]
tau = 20
...

% components
activation
output
```

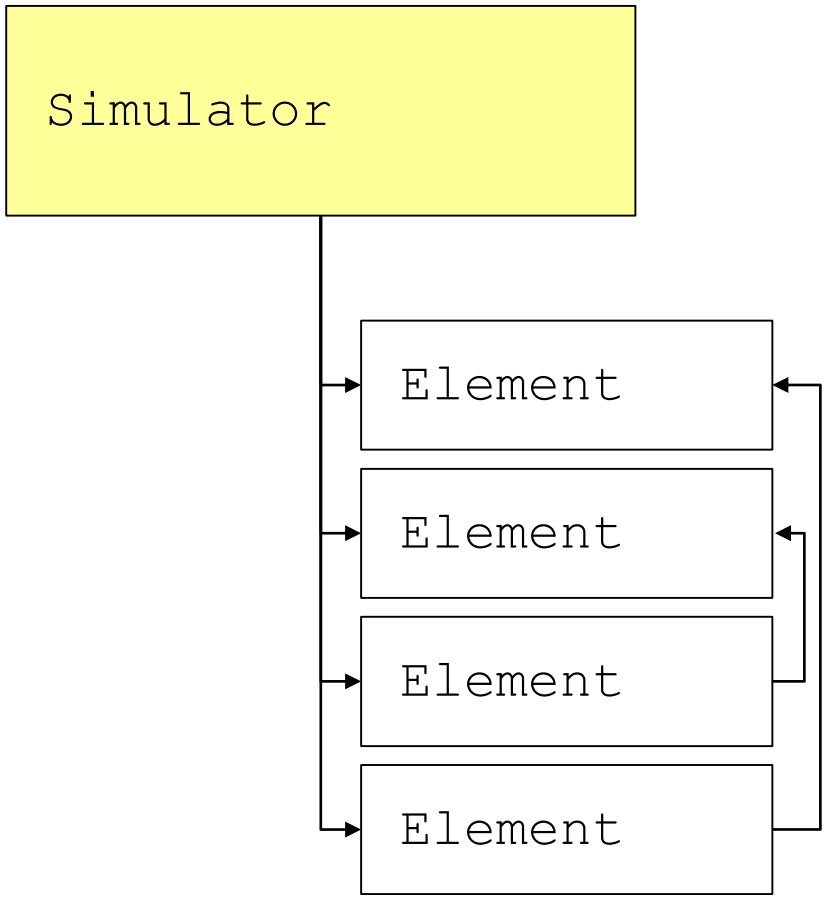
Connections between Elements



Element Classes

```
NeuralField < Element

methods

NeuralField(...) % constructor, creates element object
with given parameters

init() % initializes the element

step(time, deltaT) % updates element for one Euler step

close() % for classes that connect to hardware

... % some auxiliary functions to manage inputs etc.
```

Structure for DNF Architectures



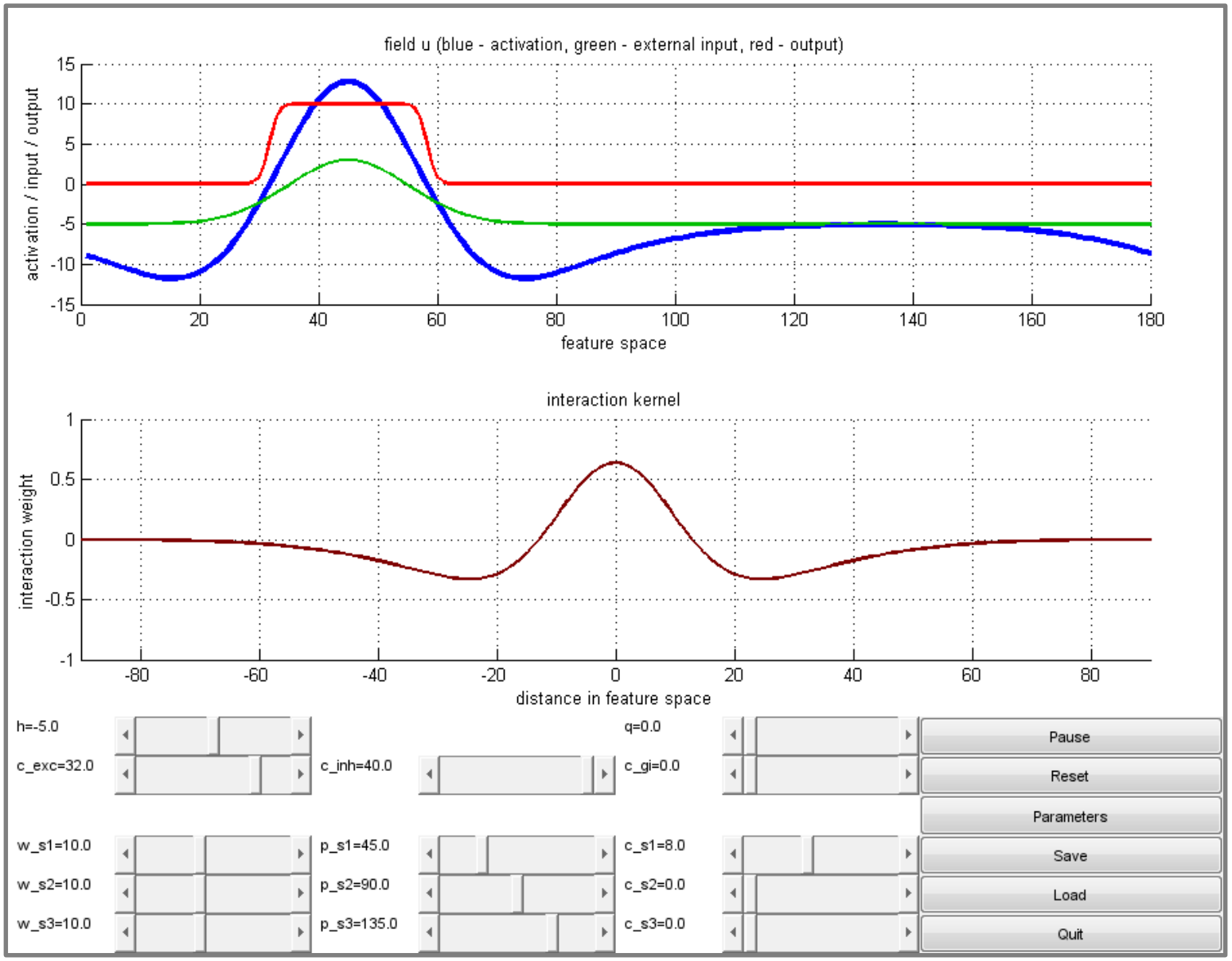
Simulator class manages elements, provides functions to initialize and simulate whole architecture

Elements implemented as handle classes, can have handles to other elements that provide input to them

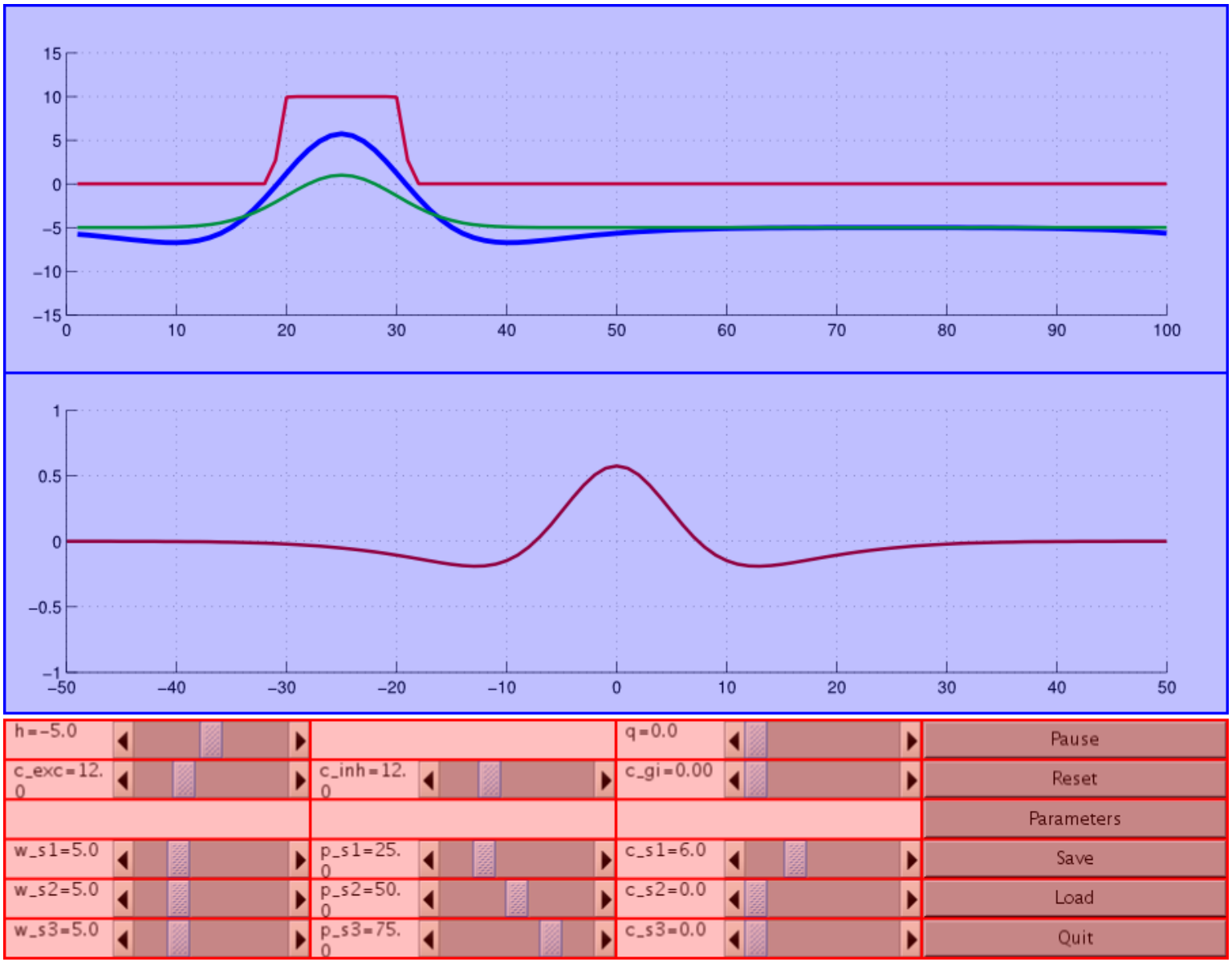
Simulator Class

```
Simulator

% general parameters
t
tZero
deltaT


methods

init() % initializes all element

step() % performs one Euler step

close() % closes all elements

run(tMax, ...) % runs simulation until specified time

... % a lot of methods to access elements and parameters
```

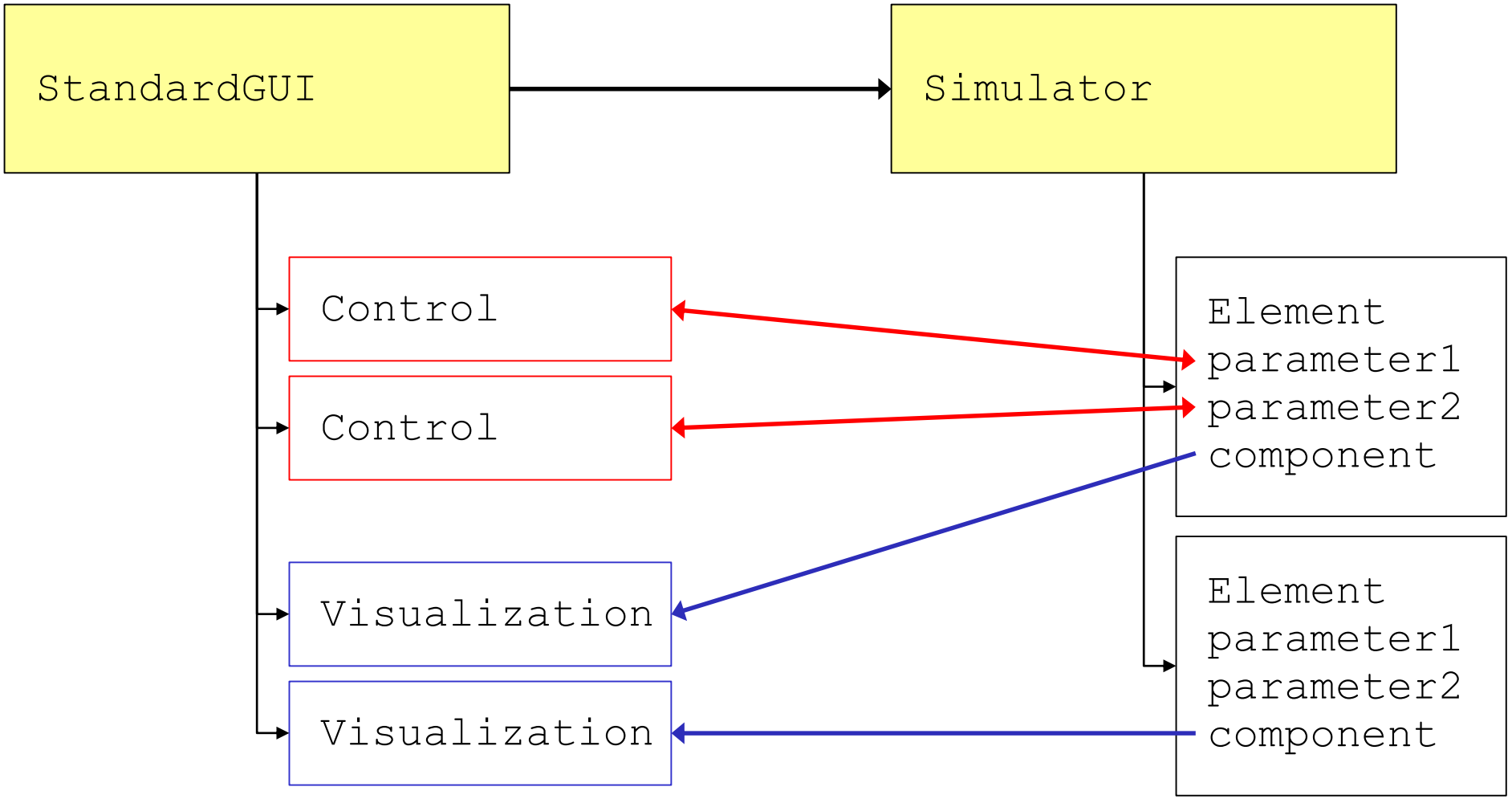
Graphical User Interfaces



Arrangement of Controls and Visualizations



Connection between GUI and Simulator



How to Use the Framework

- “offline” mode without the GUI: run simulator (perhaps for many trials), plot or analyze resulting activities manually
- “offline” mode with GUI: run and view a simulation with specified time course of stimuli etc.
- “online” mode: GUI controls the simulation, runs continuously, change parameters through control elements
- and change back and forth between these modes of operation…

Resources

- get the cosivina toolbox at https://bitbucket.org/sschneegans/cosivina/
- documentation available as pdf or as wiki on bitbucket (the wiki is a bit more up to date)
- Matlab help for all elements, controls and visualization