



NeuroTools

Set of tools to organizer and analyze simulations

Jens Kremkow

DyVA, Institut de Neurosciences Cognitives de la Méditerranée
CNRS & Aix-Marseille Université

Neurobiologie und Biophysik, Institut für Biologie III
Albert-Ludwigs-Universität Freiburg



Where to get it



Where to find:

<http://neuralensemble.org/trac/NeuroTools>

Download:

```
svn co https://neuralensemble.kip.uni-heidelberg.de/svn/NeuroTools/trunk  
NeuroTools
```

Install:

```
python setup.py install
```

or

```
python setup.py install --prefix=/mypath
```

Overview



① Organization of model parameters and scanning parameter spaces

- ParameterSet
- ParameterRange

② Data analysis/plotting:

- spikes

Organization of model parameters and scanning parameter spaces

ParameterSet, ParameterRange and some other useful functions

ParameterSet

defining the parameters of the LGN model



```
from NeuroTools.parameters import ParameterSet
# defining a parameter set of for your model
```

```
# option 1
p = ParameterSet({'size':10.})
p.Ac = 1.
p.As = 1./3.
p['K1'] = 1.05
p.update({'Ac':902,'kl':'89'})
```

```
p.b = ParameterSet({'c':34})
```

```
# option 2
p_dict = arbitrarily nested dict
p = ParameterSet(p_dict)
```

LGN RF parameters. Spatiotemporal LGN RFs are modeled as in Cai et al. (1997). Spatial profiles are described with a DOG, and temporal profiles are described as a difference of gamma functions, with distinct center and surround components. The full expression is $RF(x,t) = F_c(x)G_c(t) - F_s(x)G_s(t)$, where

$$F_c(x) = A_c e^{-x^2/2\sigma_c^2},$$

and $F_s(x)$ is defined analogously. The temporal filter for the center is as follows:

$$G_c(t) = K_1 \frac{(c_1(t - t_1))^{n_1} e^{-c_1(t-t_1)}}{n_1^{n_1} e^{-n_1}} - K_2 \frac{(c_2(t - t_2))^{n_2} e^{-c_2(t-t_2)}}{n_2^{n_2} e^{-n_2}},$$

and $G_s(t) = G_c(t - td)$.

Most parameters are fixed to the geometric means of their distributions as reported in Cai et al. (1997): $A_s/A_c = 0.3$; $K_1 = 1.05$; $c_1 = 0.14$; $n_1 = 7$; $K_2 = 0.7$; $c_2 = 0.12$; $n_2 = 8$.

Allen & Freeman 2006

ParameterSet

addressing/iterating the content



addressing

p['Ac']

p.Ac

p['b']['c']

p.b['c']

p.b.c

p['b.c']

iterating

for key in p:

....

for value in p.values():

....

Scanning parameter spaces



```
p = ParameterSet({})
```

```
p.Ac = 1.
```

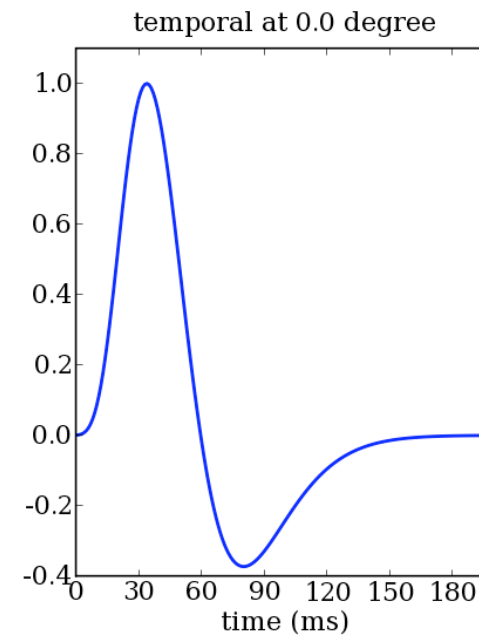
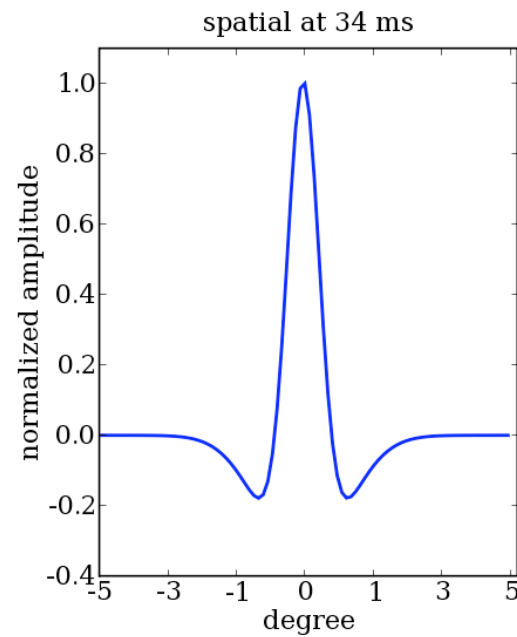
```
p.As = 1./3.
```

```
p.K1 = 1.05
```

```
...
```

```
p.td = 6.0 # time difference between ON-OFF
```

```
p.sigma_c = 0.4 # sigma of center
```



Creating parameter spaces by for loops



p # ParameterSet from the LGN model

you want to change certain parameters e.g.:
p.td # time difference between ON-OFF

td = [1,2,3,4]

for n in td:

 p.td = n

 x,t = space, time

 kernel = RF(x,t,p)

Creating parameter spaces by for loops



p # ParameterSet from the LGN model

you want to change certain parameters e.g.:

p.td # time difference between ON-OFF

p.sigma_c # sigma of center

```
td = [1,2,3,4]
```

```
sigma = [2,5,6,8]
```

```
for n in td:
```

```
    for m in sigma:
```

```
        p.update({'td':n, 'sigma_c':m})
```

```
        x,t = space, time
```

```
        kernel = RF(x,t,p)
```

Creating parameter spaces by for loops



p # ParameterSet from the LGN model

you want to change certain parameters e.g.:

p.td # time difference between ON-OFF

p.sigma_c # sigma of center

...

td = [1,2,3,4]

sigma_c = [2,5,6,8]

sigma_s = [29,387,4]

for n in td:

 for m in sigma_c:

 for o in sigma_s:

 p.update({'td':n, 'sigma_c':m, 'sigma_s':o})

 x,t = space, time

 kernel = RF(x,t,p)

Creating parameter spaces by for loops



p # ParameterSet from the LGN model

you want to change certain parameters e.g.:

p.td # time difference between ON-OFF

p.sigma_c # sigma of center

```
td = [1,2,3,4]
```

```
sigma_c = [2,5,6,8]
```

```
sigma_s = [29,387,4]
```

```
Ac = [3,4,5]
```

```
for n in td:
```

```
    for m in sigma_c:
```

```
        for o in sigma_s:
```

```
            for q in Ac:
```

```
                p.update({'td':n,'sigma_c':m,'sigma_s':o,'Ac':q})
```

```
                x,t = space, time
```

Creating parameter spaces by ParameterRange



```
p # ParameterSet from the LGN model  
from NeuroTools.parameters import ParameterRange
```

you want to change certain parameters e.g.:

p.td # time difference between ON-OFF

p.sigma_c # sigma of center

```
p.td = ParameterRange([1,2,3,4])
```

```
p.range_keys() # list of parameters that contain a range
```

```
for experiment in p.iter_inner():
```

```
    x,t = space, time
```

```
    kernel = RF(x,t,experiment)
```

Creating parameter spaces by ParameterRange



```
p # ParameterSet from the LGN model  
from NeuroTools.parameters import ParameterRange
```

you want to change certain parameters e.g.:

```
p.td # time difference between ON-OFF  
p.sigma_c # sigma of center
```

```
p.td = ParameterRange([1,2,3,4])  
p.sigma_c = ParameterRange([2,5,6,8])
```

```
for experiment in p.iter_inner():  
    x,t = space, time  
    kernel = RF(x,t,experiment)
```

Creating parameter spaces by ParameterRange



```
p # ParameterSet from the LGN model  
from NeuroTools.parameters import ParameterRange
```

you want to change certain parameters e.g.:

p.td # time difference between ON-OFF

p.sigma_c # sigma of center

```
p.td = ParameterRange([1,2,3,4])
```

```
p.sigma_c = ParameterRange([2,5,6,8])
```

```
p.sigma_s = ParameterRange([29,387,4])
```

```
for experiment in p.iter_inner():
```

```
    x,t = space, time
```

```
    kernel = RF(x,t,experiment)
```

Creating parameter spaces by ParameterRange



```
p # ParameterSet from the LGN model  
from NeuroTools.parameters import ParameterRange
```

you want to change certain parameters e.g.:

p.td # time difference between ON-OFF

p.sigma_c # sigma of center

```
p.td = ParameterRange([1,2,3,4])
```

```
p.sigma_c = ParameterRange([2,5,6,8])
```

```
p.sigma_s = ParameterRange([29,387,4])
```

```
p.Ac = ParameterRange([3,4,5])
```

```
for experiment in p.iter_inner():
```

```
    x,t = space, time
```

```
    kernel = RF(x,t,experiment)
```

Creating parameter spaces by ParameterRange (even nested parameters)



```
p # ParameterSet from the LGN model  
from NeuroTools.parameters import ParameterRange
```

you want to change certain parameters e.g.:

p.td # time difference between ON-OFF

p.sigma_c # sigma of center

```
p.td = ParameterRange([1,2,3,4])
```

```
p.sigma_c = ParameterRange([2,5,6,8])
```

```
p.sigma_s = ParameterRange([29,387,4])
```

```
p.Ac = ParameterRange([3,4,5])
```

```
p.weight.LGN.Cortex = ParameterRange([1,2])
```

for experiment in p.iter_inner():

 x,t = space, time

 kernel = RF(x,t,experiment)

demo



```
from LGN_receptive_field import *
```

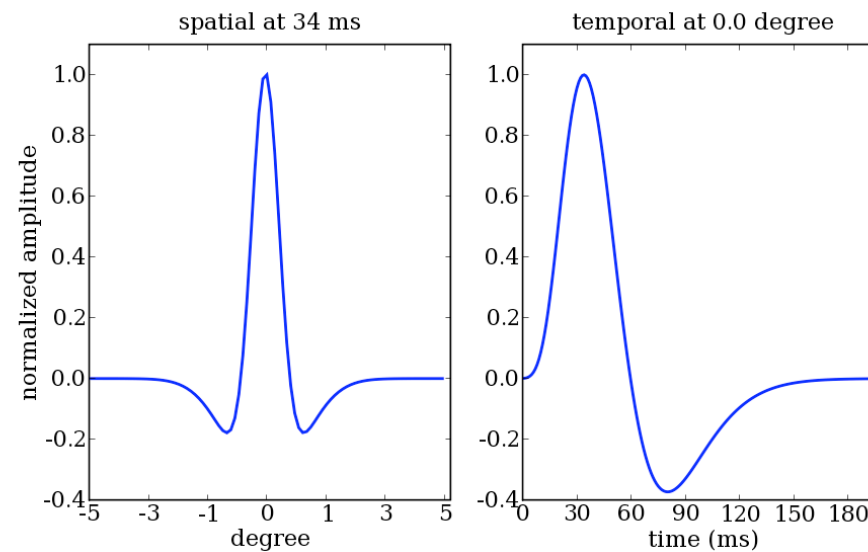
```
p = default_parameters()
```

```
p.td = ParameterRange([0.,6.,20.])
```

```
p.sigma_c = ParameterRange([0.1,0.4,0.8])
```

```
for experiment in p.iter_inner():
```

```
    plot_receptive_field(p=experiment)
```



NeuroTools.sandbox

make_name



make_name

“creates a string of the parameters in an experiment that have a range
it orders the parameters alphabetically
and attaches the actual value behind the parameter name“

```
from NeuroTools.sandbox import make_name
```

```
...
```

```
for experiment in p.iter_inner():
```

```
    name = make_name(experiment,p.range_keys())
```

```
    plot_receptive_field(p=experiment,label=name)
```

NeuroTools.sandbox

check_name



check_name

“Takes a name and checks if name+'_running' exists on the filesystem.
If it does not it touches the name+'_running' and returns True.
If it exists it returns False”

```
from NeuroTools.sandbox import check_name
```

```
...
```

```
for experiment in p.iter_inner():  
    name = make_name(experiment,p.range_keys())  
    still_to_simulate = check_name(name)  
    if still_to_simulate:  
        plot_receptive_field(p=experiment,label=name)  
    else:  
        load data, analyze data ...
```

ParameterRange

dimension, labels of the parameter space



```
p = default_parameters()
p.td = ParameterRange([0.,6.,20.])
p.sigma_c = ParameterRange([0.1,0.4,0.8])
```

```
dim, labels = p.parameter_space_dimension_labels()
```

```
results = numpy.empty(dim)
```

ParameterRange

index in the parameter space



```
p = default_parameters()
p.td = ParameterRange([0.,6.,20.])
p.sigma_c = ParameterRange([0.1,0.4,0.8])
```

```
dim, labels = p.parameter_space_dimension_labels()
```

```
results = numpy.empty(dim)
```

```
for experiment in p.iter_inner():
    index = p.parameter_space_index(experiment)
    simulate
    data = load data
    results[index] = data
```

Complete setup very simple



```
p = default_parameters()
p.td = ParameterRange([0.,6.,20.])
p.sigma_c = ParameterRange([0.1,0.4,0.8])
```

variable

```
dim, labels = p.parameter_space_dimension_labels()
results = numpy.empty(dim)
for experiment in p.iter_inner():
    name = make_name(experiment, p.range_keys())
    still_to_simulate = check_name(name)
    if still_to_simulate:
        index = parameters.parameter_space_index(experiment)
        simulate
        data = load_data(name)
        results[index] = data
    else:
        data = load_data(name)
        results[index] = data
```

fixed

Data analysis/plotting

Using the SpikeTrain and SpikeLists for easy data visualization

NeuroTools.spikes



Tool for analyzing and plotting

- spikes
- membrane traces
- current traces
- conductance traces

Get the spikes into the SpikeList



```
import numpy
import NeuroTools.spikes as spikes
```

```
neurons = numpy.arange(1,160)
```

```
start = 0. # ms
```

```
stop = 10000. # ms
```

```
dt = 0.1 # ms
```

```
filename = 'example_spikes'
```

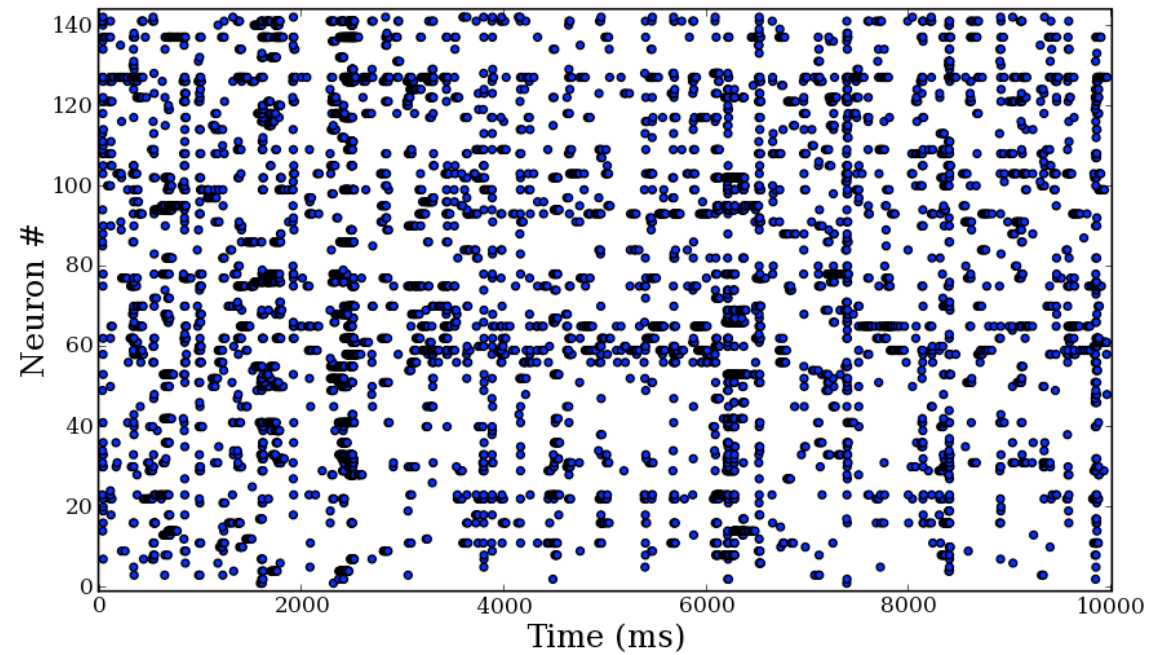
```
spike_data = spikes.loadSpikeList('example_spikes',neurons,dt,start,stop)
```

from here on it gets even more simple

Raster Plots



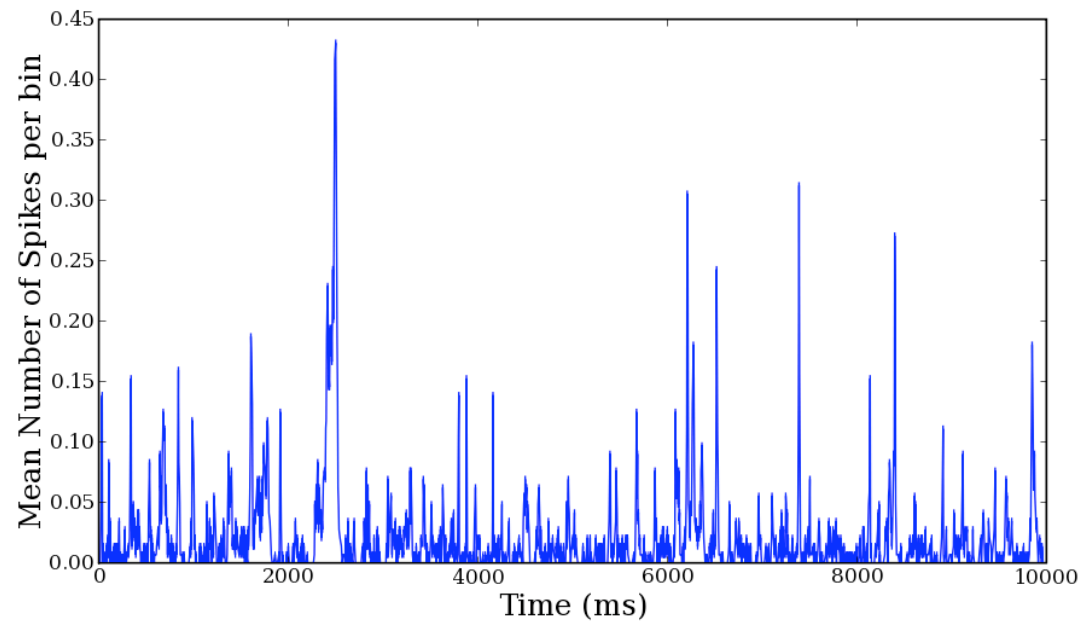
```
# raster plot  
spike_data.raster_plot()
```



PSTH



```
# plotting psth  
spike_data.spike_histogram(time_bin=5.,display=True)  
# just analyze  
psth = spike_data.spike_histogram(time_bin=5.,display=False)
```



Mean rate, fano factor ...

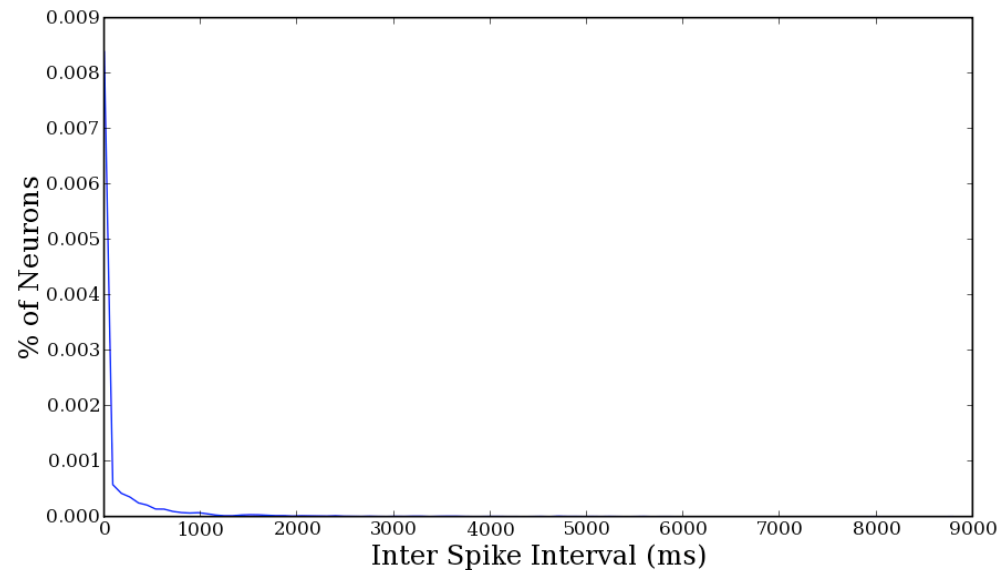


```
# mean firing rate  
spike_data.mean_rate()
```

```
# fano factor  
spike_data.fano_factor(2.)
```

```
# isi distribution  
spike_data.isi(display=True)
```

...



you miss a function??? feel free to contribute and share your code

NeuroTools.spikes membrane, conductance traces



spikes.loadMembraneTraceList

spikes.loadConductanceTraceList

spikes.loadCurrentTraceList

still under development, but feel free to contribute

Summary



- NeuroTools.parameters:
 - organize your parameters
 - easy scanning of parameter spaces
 - some functions to avoid re-simulating data
- NeuroTools.spikes:
 - analyzing/plotting
 - of spike trains, membrane/conductances/current traces
 - still under development, feel free to contribute and share code