QUALITY OF LIFE IN NEURON
Weak points of NEURON’s Python interface

• Thin wrapper
• Exposes quirks of C to Python
• Not an expressive or convenient API
• Many peculiar need-to-know things
Example

```python
s = h.Section()
s2 = h.Section()
syn = h.ExpSyn(s2(0.5))
syn.tau = 2
nc = h.NetCon(s(0.5)._ref_v, syn, sec=s)
nc.weight[0] = 0.1
```

```python
s = h.Section()
s2 = h.Section()
syn = h.ExpSyn(s2, tau=2)
nc = s.connect(syn, weight=0.1)
```
QUALITY OF LIFE IMPROVEMENTS

FLATTEN THE (LEARNING) CURVE

SANE DEFAULTS

HIDE BACKGROUND CLUTTER

INCREASE EXPRESSIVENESS
nrn-patch

• Drop-in replacement of NEURON’s Python module

• Fixes bugs, gotchas and improves default behaviors

• Wraps C-like NEURON objects in a Python object

• Adds expressiveness & convenience methods
Object referencing

In NEURON objects connected to each other can disappear without throwing an error when their Python reference disappears:

```python
def connect(s1, syn):
    h.NetCon(s1(0.5)._ref_v, syn)
```
QoL convenience layer in action

Detecting the spikes of a Section and connecting them to GID 1:

```python
from neuron import h
gid = 1
h.nrnmpi_init()
pc = h.ParallelContext()
s = h.Section()
nc = h.NetCon(s(0.5)._ref_v, None)
p.set_gid2node(gid, pc.id())
pc.cell(gid, nc)
pc.outputcell(gid)
```

Connecting the spikes of GID 1 to a synapse:

```python
from neuron import h
 gid = 1
 h.nrnmpi_init()
 pc = h.ParallelContext()
 s = h.Section()
 syn = h.SynExp(s)
 pc.gid_connect(gid, syn)
```

Detecting the spikes of a Section and connecting them to GID 1:

```python
from patch import p
gid = 1
s = p.Section()
nc = p.ParallelCon(s, gid)
```

Connecting the spikes of GID 1 to a synapse:

```python
from patch import p
gid = 1
syn = p.Section().synapse(p.SynExp)
nc = p.ParallelCon(gid, syn)
```
nrn-glia

Library  Packages  Versioning
01 Load/build morphologies
02 Define section and synapse types
03 Label sections
bc = BasketCell()
bc.record_soma()
p.run(1000)
go.Figure(go.Scatter(x=p.time, y=bc.Vm)).show()
Contact

robingilbert.deschepper@unipv.it