

The NeuroML ecosystem for standardised multi-scale modelling in neuroscience

Ankur Sinha
on behalf of NeuroML contributors
28/02/2026

What we've been up to on the NeuroML front

- Google summer of code (GSoC)
- Paper and general improvements

Google Summer of Code

- Under the INCF organisation
 - 40 accepted this year, in total
 - No longer limited to students

- Under the INCF organisation
 - 40 accepted this year, in total
 - No longer limited to students
- Us: 4 candidates
 - 3 working on NeuroML/modelling
 - 1 on NWB
 - 175 hours each, over 3 months (medium projects)

- Under the INCF organisation
 - 40 accepted this year, in total
 - No longer limited to students
- Us: 4 candidates
 - 3 working on NeuroML/modelling
 - 1 on NWB
 - 175 hours each, over 3 months (medium projects)
- Goals:
 - Conversions of models to NeuroML (to allow them to be re-used and featured on OSB)
 - Get folks looking into and using NeuroML in different use cases

- Under the INCF organisation
 - 40 accepted this year, in total
 - No longer limited to students
- Us: 4 candidates
 - 3 working on NeuroML/modelling
 - 1 on NWB
 - 175 hours each, over 3 months (medium projects)
- Goals:
 - Conversions of models to NeuroML (to allow them to be re-used and featured on OSB)
 - Get folks looking into and using NeuroML in different use cases
 - Get folks to improve NeuroML where possible

- Masters student at the Bernstein Centre for Computational Neuroscience, Berlin.

GSoC: Anuja: convert Allen Institute models to NeuroML

- Masters student at the Bernstein Centre for Computational Neuroscience, Berlin.
- Source models: [Allen institutes cell type database](#) (example)

GSoC: Anuja: convert Allen Institute models to NeuroML

- Masters student at the Bernstein Centre for Computational Neuroscience, Berlin.
- Source models: [Allen institutes cell type database](#) ([example](#))
- Steps:
 - Automate download of models using the Allen SDK
 - Automate conversion to NeuroML
 - Plot comparison graphs to validate conversions ([LIF models, Detailed](#))
 - Document comparison, usage, update OMV tests for CI

GSoC: Anuja: example figure 1

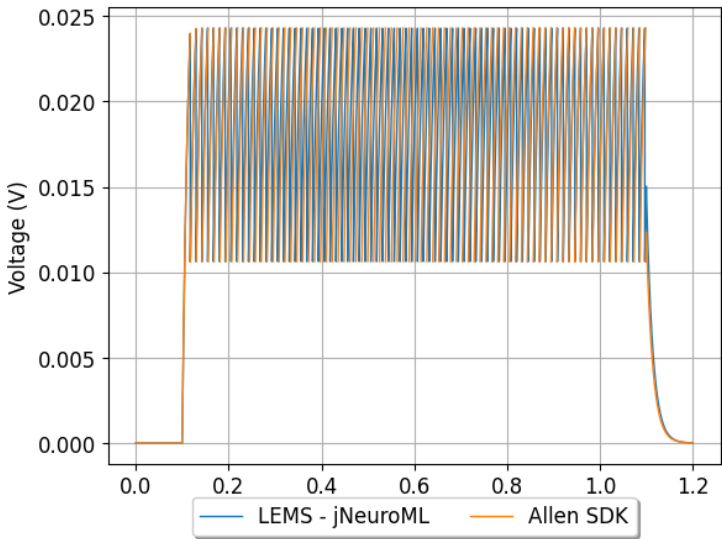


Figure 1: Membrane potentials of Allen and NeuroML models

GSoC: Anuja: example figure II

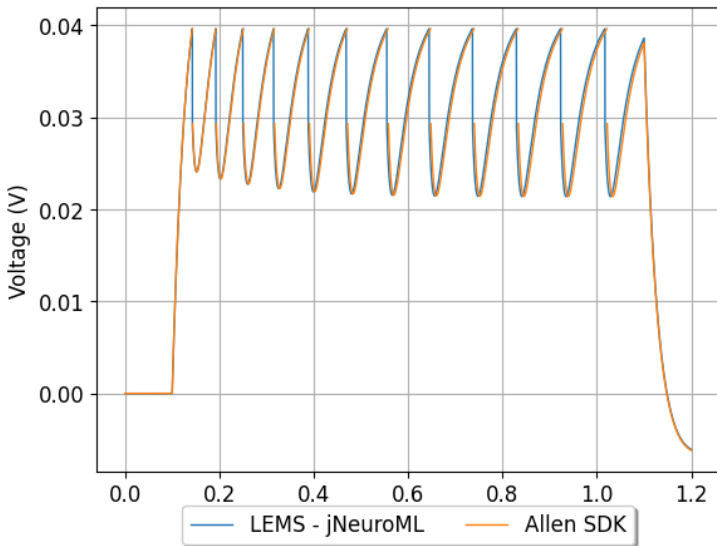


Figure 2: Membrane potentials of Allen and NeuroML models

GSoC: Anuja: example figure III

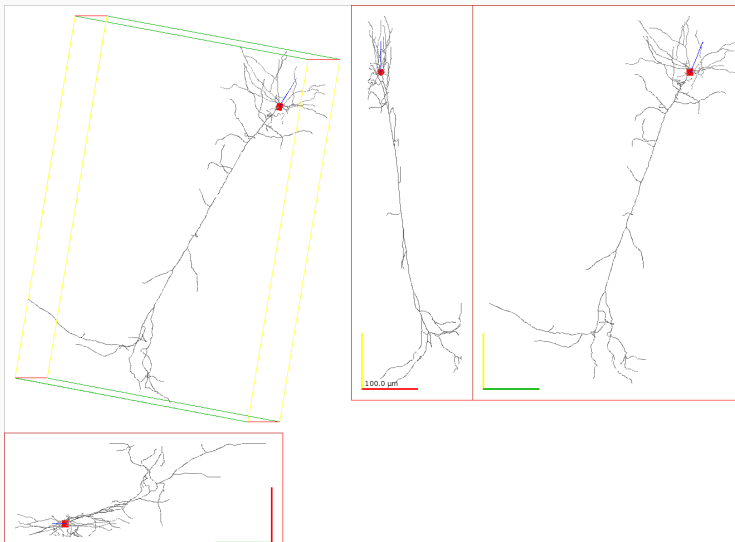


Figure 3: Morphology of example cell

GSoC: Anuja: example figure IV

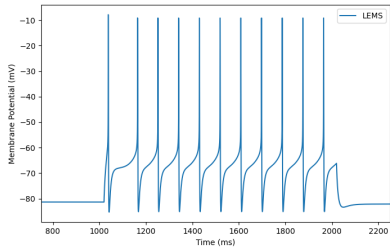
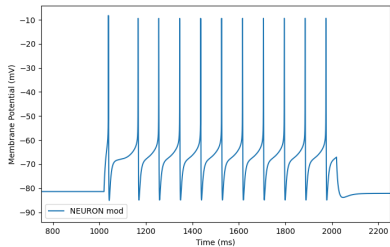


Figure 4: NEURON vs NeuroML output

GSoC: Shayan: convert BahlEtAl2012 (Reduced L5 Pyr Cell) to NeuroML

- MSc/BSc from IIT Kharagpur (Maths and Computing)

GSoC: Shayan: convert BahlEtAl2012 (Reduced L5 Pyr Cell) to NeuroML

- MSc/BSc from IIT Kharagpur (Maths and Computing)
- Source models: [OSB/BahlEtAl2012](#)

GSoC: Shayan: convert BahlEtAl2012 (Reduced L5 Pyr Cell) to NeuroML

- MSc/BSc from IIT Kharagpur (Maths and Computing)
- Source models: [OSB/BahlEtAl2012](#)
- Steps:
 - Convert ion channels to NeuroML, validate, [compare with NEURON mod files](#)
 - Implement single compartment cell model with passive channels
 - incrementally add ion channels
 - compare with NEURON model
 - Implement multi-compartmental cell, repeat
 - [Document comparison, usage](#), update OMV tests for CI
 - [Interactive notebook](#) to reproduce figures from paper using NeuroML models

GSoC: Shayan: example figures

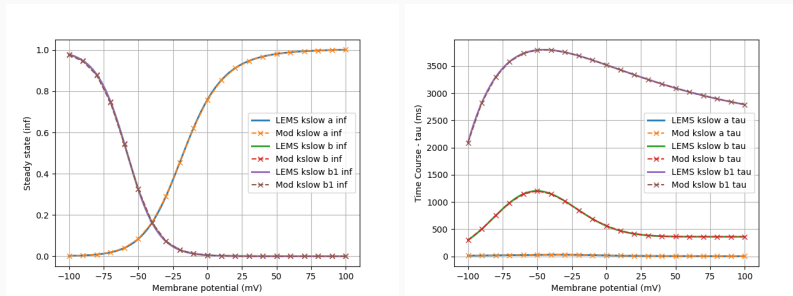


Figure 5: Comparing ion channels: steady state, time constant

GSoC: Shayan: example figures II

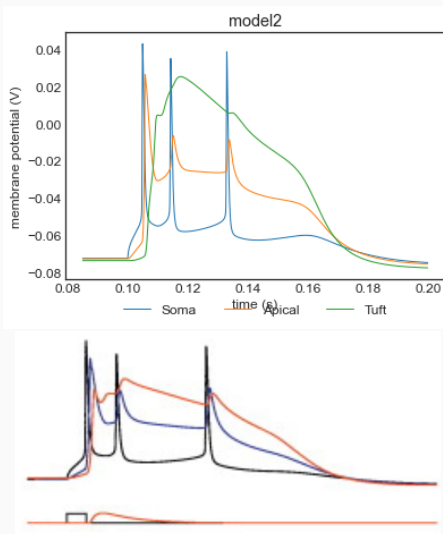


Figure 6: Replicating figures from the paper (8d)

- Aerospace engineer (MTech, IIT Kharagpur), working in South Korea

GSoC: Rahul: convert HH tutorial to Jupyter notebook

- Aerospace engineer (MTech, IIT Kharagpur), working in South Korea
- Source models: [OpenWorm/HH tutorial](#)

GSoC: Rahul: convert HH tutorial to Jupyter notebook

- Aerospace engineer (MTech, IIT Kharagpur), working in South Korea
- Source models: [OpenWorm/HH tutorial](#)
- Steps:
 - Investigate Jupyter widgets
 - Convert pure Python HH tutorial to use Jupyter widgets
 - Investigate NeuroML based tutorial
 - Investigate conversion of NeuroML to Jupyter widgets
 - Update sphinx documentation for [ReadTheDocs site](#)
 - [Document usage](#).

GSoC: Rahul: example I

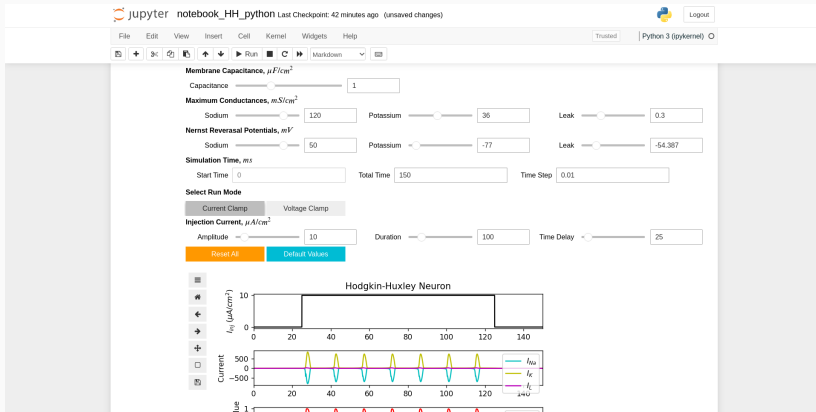


Figure 7: Pure python tutorial converted to use Jupyter Widgets

GSoC: Rahul: example II

The screenshot displays a Jupyter Notebook interface for a NeuroML simulation. The notebook title is "notebook_NML" with "(unsaved changes)" in parentheses. The top bar includes a "Logout" button and a "Python 3 (ipykernel)" indicator. The menu bar contains "File", "Edit", "View", "Insert", "Cell", "Kernel", "Widgets", and "Help". The toolbar shows icons for file operations and a "Run" button. The main content area features a series of sliders and input fields for simulation parameters:

- Membrane Capacitance, $\mu F/cm^2$** : A slider set to 1.
- Maximum Conductances, mS/cm^2** :
 - Sodium**: A slider set to 120.
 - Potassium**: A slider set to 36.
 - Leak**: A slider set to 0.3.
- Nernst Reversal Potentials, mV** :
 - Sodium**: A slider set to 50.
 - Potassium**: A slider set to -77.
 - Leak**: A slider set to -64.3.
- Simulation Time, ms** :
 - Start Time**: A text input set to 0.
 - Total Time**: A text input set to 150.
 - Time Step**: A text input set to 0.01.
- Injection Current, nA** :
 - Amplitude**: A slider set to 0.1.
 - Duration**: A slider set to 100.
 - Time Delay**: A slider set to 25.

Below the sliders are two buttons: "Reset All" (orange) and "Default Values" (blue). A green "Run NeuroML" button is also present. The bottom section of the notebook shows the terminal output of the command `pyNeuroML`:

```
pyNeuroML >>> INFO - Executing: (java -Xmx400M -Djava.awt.headless=true -jar "/home/asinha/.local/share/virtuale
nvs/neuroml-dev/lib/python3.10/site-packages/pyneuroml/lib/jNeuroML-0.12.0-jar-with-dependencies.jar" ".../Tuto
rial/Source/LEMS_HH_Simulation.xml/" -nogui) in directory: .
pyNeuroML >>> INFO - Command completed. Output:
jNeuroML >>> jNeuroML v0.12.0
jNeuroML >>> Loading: /home/asinha/Documents/02_Code/00_mine/models/hodgkin_huxley_tutorial/notebooks/NeuroML_HH
_version/.../Tutorial/Source/LEMS_HH_Simulation.xml with jLEMS, NO GUI mode...
jNeuroML >>> INFO Sep 21,2022 16:26 (INFO) Loading LEMS file from: /home/asinha/Documents/02_Code/00_mine/model
s/hodgkin_huxley_tutorial/notebooks/NeuroML_HH_version/.../Tutorial/Source/LEMS_HH_Simulation.xml
jNeuroML >>> INFO Sep 21,2022 16:26 (INFO) Reading from: /home/asinha/Documents/02_Code/00_mine/models/hodgkin_
huxley_tutorial/notebooks/NeuroML_HH_version/.../Tutorial/Source/LEMS_HH_Simulation.xml
jNeuroML >>> INFO Sep 21,2022 16:26 (INFO) Finished 15000 steps in 0.553 seconds (sim duration: 150.0ms; dt: 0.
01ms)
jNeuroML >>> INFO Sep 21,2022 16:26 (INFO) Written to the file /home/asinha/Documents/02_Code/00_mine/models/ho
dgin_huxley_tutorial/notebooks/NeuroML_HH_version/.../Tutorial/Source/LEMS_HH_Simulation.xml
```

Figure 8: NeuroML tutorial converted to use Jupyter Widgets

Paper and general updates

Paper: NeuroML is the best thing since sliced bread

Convince readers (research community) to use NeuroML for their modelling work*.

Paper: NeuroML is the best thing since sliced bread

Convince readers (research community) to use NeuroML for their modelling work*.

- *instead/ahead of other tools

Paper: NeuroML is the best thing since sliced bread

Convince readers (research community) to use NeuroML for their modelling work*.

- *instead/ahead of other tools
- *on a daily basis
 - not as an afterthought for standardisation—once the paper has been published no one has time to re-write model (or re-process data!) to standardise

Paper: NeuroML is the best thing since sliced bread

Convince readers (research community) to use NeuroML for their modelling work*.

- *instead/ahead of other tools
- *on a daily basis
 - not as an afterthought for standardisation—once the paper has been published no one has time to re-write model (or re-process data!) to standardise
 - the carrot, “standardisation is good for science”, isn’t enough in a mostly resource limited academic/research system
 - requires stick: “for this grant, you must ...”; “for this journal, you must ...”

You should use NeuroML because

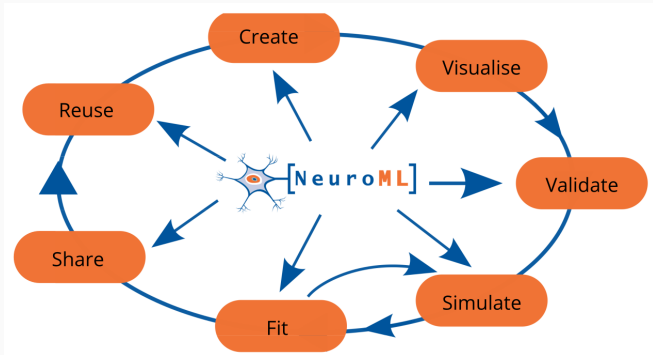


Figure 9: NeuroML overview figure from paper

You should use NeuroML because

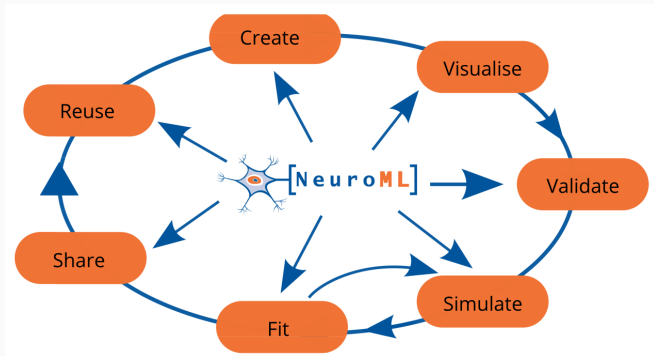


Figure 9: NeuroML overview figure from paper

These claims are all true, and have been for quite a while.

So many advantages: why isn't everyone using it?

So many advantages: why isn't everyone using it?

Irrespective of all its features, NeuroML **needs to be easier to use than other tools.**

So many advantages: why isn't everyone using it?

Irrespective of all its features, NeuroML **needs to be easier to use than other tools.**

- information on NeuroML needs to be easy to find:
 - website
 - point of entry for completely new users: first impression
 - replaced with modern looking static page that redirects to individual pages in docs
 - all information migrated to docs

So many advantages: why isn't everyone using it?

Irrespective of all its features, NeuroML **needs to be easier to use than other tools.**

- information on NeuroML needs to be easy to find:
 - website
 - point of entry for completely new users: first impression
 - replaced with modern looking static page that redirects to individual pages in docs
 - all information migrated to docs
 - docs
 - re-organised, modernised
 - include tutorials, interactive tutorials via Jupyter note books, how-tos
 - complete searchable schema docs
 - still not yet fully complete

So many advantages: why isn't everyone using it?

Irrespective of all its features, NeuroML **needs to be easier to use than other tools.**

- usability: so much can be done, but can it be done **easily**?
 - has not been clearly easier to use than other tools
 - Python API exists, but we haven't taken advantage of it enough to **make life easier for users** yet

Example: create: single neuron Izhikevich network (from docs)

```
1  nml_doc = NeuroMLDocument(id="IzhSingleNeuron")
2
3  izh0 = Izhikevich2007Cell(
4      id="izh2007RS0", v0="-60mV", C="100pF", k="0.7nS_per_mV", vr="-60mV",
5      vt="-40mV", vpeak="35mV", a="0.03per_ms", b="-2nS", c="-50.0mV", d="100pA")
6  nml_doc.izhikevich2007_cells.append(izh0)
7
8  net = Network(id="IzhNet")
9  nml_doc.networks.append(net)
10
11  size0 = 1
12  pop0 = Population(id="IzhPop0", component=izh0.id, size=size0)
13  net.populations.append(pop0)
14
15  pg = PulseGenerator(
16      id="pulseGen_%i" % 0, delay="0ms", duration="1000ms",
17      amplitude="0.07 nA"
18  )
19  nml_doc.pulse_generators.append(pg)
20  exp_input = ExplicitInput(target="%s[%i]" % (pop0.id, 0), input=pg.id)
21  net.explicit_inputs.append(exp_input)
22
23  nml_file = 'izhikevich2007_single_cell_network.nml'
24  writers.NeuroMLWriter.write(nml_doc, nml_file)
25  print("Written network file to: " + nml_file)
26
27  validate_neuroml2(nml_file)
```

Inspect/visualise network

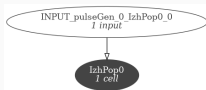


Figure 10: Generated network graph

```
>>> nml_doc.summary()
*****
* NeuroMLDocument: IzhSingleNeuron
*
* Izhikevich2007Cell: ['izh2007RS0']
* PulseGenerator: ['pulseGen_0']
*
* Network: IzNet
*
* 1 cells in 1 populations
*   Population: IzhPop0 with 1 components of type izh2007RS0
*
* 0 connections in 0 projections
*
* 0 inputs in 0 input lists
*
* 1 explicit inputs (outside of input lists)
*   Explicit Input of type pulseGen_0 to IzhPop0(cell 0), destination: unspecified
*
*****
```

Strengths

- extremely **declarative**
 - components (Izhikevich2007Cell, Network, Population, PulseGenerator, ExplicitInput) clearly visible
 - components and dynamics fully, formally documented in schema docs
 - component parameters clearly visible
 - units/dimensions explicitly mentioned

izhikevich2007Cell

extends [baseCellMembPotCap](#)

Cell based on the modified Izhikevich model in Izhikevich 2007, *Dynamical systems in neuroscience*, MIT Press.

Parameters Exposures Event Ports Attachments Dynamics

Usage: Python Usage: XML

State Variables

v: [voltage](#) (exposed as v)

u: [current](#) (exposed as u)

On Start

v = v0

u = 0

On Conditions

IF v > vpeak THEN

v = c

u = u + d

EVENT OUT on port: [spike](#)

Derived Variables

iSyn = synapses[*]->i(reduce method: add) (exposed as iSyn)

iMemb = k * (v-vr) * (v-vr) + iSyn - u (exposed as iMemb)

Time Derivatives

d v /dt = iMemb / C

d u /dt = a * (b * (v-vr) - u)

Figure 11: Schema docs on Izhikevich2007Cell

Strengths

- **validation** without simulation (no simulator has this)
 - Level 1 validation: units/dimensions, structure of model checked against schema
 - Level 2 validation: extra “logical” checks

Strengths

- **validation** without simulation (no simulator has this)
 - Level 1 validation: units/dimensions, structure of model checked against schema
 - Level 2 validation: extra “logical” checks
- easy **simulation** with different simulators
- easy **visualisation** and inspection of model: network structure, connectivity matrices, LEMS simulation graph, morphology figures, model summary

Weaknesses

- information on components: must switch back and forth between docs and code
 - Not too bad: required for most simulators/programming languages

Weaknesses

- information on components: must switch back and forth between docs and code
 - Not too bad: required for most simulators/programming languages
- unclear how components fit together
 - how do I know that Network → Populations →?

Weaknesses

- information on components: must switch back and forth between docs and code
 - Not too bad: required for most simulators/programming languages
- unclear how components fit together
 - how do I know that Network → Populations →?
- Too many “under the hood” bits that users are expected to know:
 - `nml_doc.izhikevich2007_cells.append(izh0)`
 - how do users know this?
 - read the schema docs?
 - read the [NeuroML Python API documentation?](#)
 - read the [NeuroML Python API source?](#) (currently 64,000 lines of code)

Weaknesses

- validation upon model completion
 - using `jnml`, so requires Java
 - better than errors on run, but still quite late

Weaknesses

- validation upon model completion
 - using `jnml`, so requires Java
 - better than errors on run, but still quite late
- NeuroML generated NEURON code is less performant than native NEURON code
 - because hand-written mod files can be optimised, while we rely on a template to generate them
 - TODO: optimisation of template

Example: create: single neuron Izhikevich network (devel)

```
1  nml_doc = component_factory("NeuroMLDocument", id="IzhSingleNeuron")
2  nml_doc.info(show_contents=True)
3
4  izh0 = nml_doc.add(
5      "Izhikevich2007Cell",
6      id="izh2007RS0", v0="-60mV", C="100pF", k="0.7nS_per_mV", vr="-60mV",
7      vt="-40mV", vpeak="35mV", a="0.03per_ms", b="-2nS", c="-50.0mV", d="100pA")
8  izh0.info(show_contents=True)
9
10 net = nml_doc.add("Network", id="IzNet", validate=False)
11
12 size0 = 1
13 pop0 = net.add("Population", id="IzhPop0", component=izh0.id, size=size0)
14
15 pg = nml_doc.add(
16     "PulseGenerator",
17     id="pulseGen_%i" % 0, delay="0ms", duration="1000ms",
18     amplitude="0.07 nA"
19 )
20 exp_input = net.add("ExplicitInput", target="%s[%i]" % (pop0.id, 0), input=pg.id)
21
22 nml_doc.validate(recursive=True)
23
24 nml_file = 'izhikevich2007_single_cell_network.nml'
25 writers.NeuroMLWriter.write(nml_doc, nml_file)
26 print("Written network file to: " + nml_file)
```


The component factory

- single factory function to create new components
- runs extra checks
 - are all arguments (parameters) correct?
 - is the component valid (level 1 validation for each component at build-time)?

```
>>> izh0 = component_factory("Izhikevich2007Cell")
ValueError: Validation failed:
- Izhikevich2007Cell (None): Required value v0 is missing
- Izhikevich2007Cell (None): Required value k is missing
- Izhikevich2007Cell (None): Required value vr is missing
- Izhikevich2007Cell (None): Required value vt is missing
- Izhikevich2007Cell (None): Required value vpeak is missing
- Izhikevich2007Cell (None): Required value a is missing
- Izhikevich2007Cell (None): Required value b is missing
- Izhikevich2007Cell (None): Required value c is missing
- Izhikevich2007Cell (None): Required value d is missing
- Izhikevich2007Cell (None): Required value C is missing
- Izhikevich2007Cell (None): Required value id is missing
```

The component factory

```
>>> izh0 = component_factory(
    "Izhikevich2007Cell",
    id="izh2007RS0", v0="-60mV", C="100pF", k="0.7nS_per_mV", vr="-60mV",
    vt="-40ms", vpeak="35mV", a="0.03per_ms", b="-2nS", c="-50.0mV", d="100pA")
```

ValueError: Validation failed:

- Izhikevich2007Cell (izh2007RS0): Value "-40mS" does not match xsd pattern restrictions:

↪ `[['^(-(?[0-9]*(\\.?[0-9]+)?)([eE]-?[0-9]+)?[\\s]*(V|mV))$']]]`

New add method

- uses component factory to create a new component and run checks
- smart enough to know where the new element needs to go in the parent

```
1  izh0 = Izhikevich2007Cell(  
2      id="izh2007RS0", v0="-60mV", C="100pF", k="0.7nS_per_mV", vr="-60mV",  
3      vt="-40mV", vpeak="35mV", a="0.03per_ms", b="-2nS", c="-50.0mV", d="100pA")  
4  nml_doc.izhikevich2007_cells.append(izh0)
```

VS

```
1  izh0 = nml_doc.add(  
2      "Izhikevich2007Cell",  
3      id="izh2007RS0", v0="-60mV", C="100pF", k="0.7nS_per_mV", vr="-60mV",  
4      vt="-40mV", vpeak="35mV", a="0.03per_ms", b="-2nS", c="-50.0mV", d="100pA")
```

Inspect each component individually

```
izh0.info(show_contents=True)
Izhikevich2007Cell -- Cell based on the modified Izhikevich model in Izhikevich 2007, Dynamical
↳ systems in neuroscience, MIT Press
Please see the NeuroML standard schema documentation at
↳ https://docs.neuroml.org/Userdocs/NeuroMLv2.html for more information.
```

Valid members for Izhikevich2007Cell are:

```
* b (class: Nml2Quantity_conductance, Required)
    * Contents ('ids'/<objects>): -2nS

* C (class: Nml2Quantity_capacitance, Required)
    * Contents ('ids'/<objects>): 100pF

* c (class: Nml2Quantity_voltage, Required)
    * Contents ('ids'/<objects>): -50.0mV

* d (class: Nml2Quantity_current, Required)
    * Contents ('ids'/<objects>): 100pA

* neuro_lex_id (class: NeuroLexId, Optional)
* metaid (class: MetaId, Optional)
* v0 (class: Nml2Quantity_voltage, Required)
    * Contents ('ids'/<objects>): -60mV

* id (class: NmlId, Required)
    * Contents ('ids'/<objects>): izh2007RS0

...
```

Component Type info without creating a new component: ctinfo

```
>>> ctinfo("ExpOneSynapse")
```

```
ExpOneSynapse -- Ohmic synapse model whose conductance rises instantaneously by ( **gbase** *  
↳ **weight** ) on receiving an event, and which decays exponentially to zero with time course  
↳ **tauDecay**
```

Please see the NeuroML standard schema documentation at

↳ <https://docs.neuroml.org/Userdocs/NeuroMLv2.html> for more information.

Valid members for ExpOneSynapse are:

```
* neuro_lex_id (class: NeuroLexId, Optional)  
* gbase (class: Nml2Quantity_conductance, Required)  
* metaid (class: MetaId, Optional)  
* erev (class: Nml2Quantity_voltage, Required)  
* notes (class: xs:string, Optional)  
* id (class: NmlId, Required)  
* properties (class: Property, Optional)  
* annotation (class: Annotation, Optional)  
* tau_decay (class: Nml2Quantity_time, Required)
```

Where does this component type fit?

```
>>> ctparentinfo("HHRate")
```

Please see the NeuroML standard schema documentation at

↪ <https://docs.neuroml.org/Userdocs/NeuroMLv2.html> for more information.

Valid parents for HHRate are:

- * GateHHRates

- * forward_rate (class: HHRate, Required)

- * reverse_rate (class: HHRate, Required)

- * GateHHRatesInf

- * forward_rate (class: HHRate, Required)

- * reverse_rate (class: HHRate, Required)

- * GateHHRatesTau

- * forward_rate (class: HHRate, Required)

- * reverse_rate (class: HHRate, Required)

- * GateHHRatesTauInf

- * forward_rate (class: HHRate, Required)

- * reverse_rate (class: HHRate, Required)

- * GateHHUndetermined

- * forward_rate (class: HHRate, Optional)

- * reverse_rate (class: HHRate, Optional)

Additions to make multi-compartment cell building easier

- `set_init_memb_potential()` instead of
`cell.biophysical_properties.membrane_properties.init_memb_potential`
- `add_channel_density()`...
- `add_segment, add_unbranched_segment_list...`
 - also takes care of **NeuroLex** (now InterLex) ids, used by NEURON to create “sections”.
 - another hidden feature of NeuroML

On-going/future work: not all to be done before paper

- more utils for other common, repeated tasks (model factory/templates?)
- update docs, add more tutorials/recipes

On-going/future work: not all to be done before paper

- more utils for other common, repeated tasks (model factory/templates?)
- update docs, add more tutorials/recipes
- GUI model building
 - Jupyter widgets based model inspection/modification/creation
 - NeuroMLLite GUI (NeuroML v3?)
 - Will absorb all neuroConstruct functionality

On-going/future work: not all to be done before paper

- more utils for other common, repeated tasks (model factory/templates?)
- update docs, add more tutorials/recipes
- GUI model building
 - Jupyter widgets based model inspection/modification/creation
 - NeuroMLLite GUI (NeuroML v3?)
 - Will absorb all neuroConstruct functionality
- Complete NetPyNE UI interactions with NeuroML
 - import works, but cannot modify model
 - export needs to be implemented

On-going/future work: not all to be done before paper

- more utils for other common, repeated tasks (model factory/templates?)
- update docs, add more tutorials/recipes
- GUI model building
 - Jupyter widgets based model inspection/modification/creation
 - NeuroMLLite GUI (NeuroML v3?)
 - Will absorb all neuroConstruct functionality
- Complete NetPyNE UI interactions with NeuroML
 - import works, but cannot modify model
 - export needs to be implemented
- Migrate all code to Python (so no more Java required)
 - long term, multi-year project, requires another grant