CSCS
Centro Svizzero di Calcolo Scientifico
Swiss National Supercomputing Centre

Human Brain Project

JÜLICH
Forschungszentrum

arbor

A morphologically detailed neural network simulation library for modern high performance computer architectures.

Nora Abi Akar, Ben Cumming, Stuart Yates (CSCS); Thorsten Hater, Brent Huisman, Anne Küsters (Forschungszentrum Jülich)

We present recent developments in Arbor, a library for the simulation of morphologically detailed neurons and networks thereof [1]. Arbor places strong emphasis on performance, portability, and usability. It can exploit modern architectures based on super-scalar multi-core processors and GPU accelerators. We showcase some of the features added to Arbor since the last release and how they can used to almost directly import and run single cell models from the Allen Brain Atlas database. These are interesting, non-trivial models on their own, but more importantly form the basis for a set of models for the mouse primary visual cortex network [2].

Where to find us

Website arbor-sim.github.io
Source code github.com/arbor-sim/arbor
Documentation arbor.readthedocs.io
Contact arbor-sim@fz-juelich.de

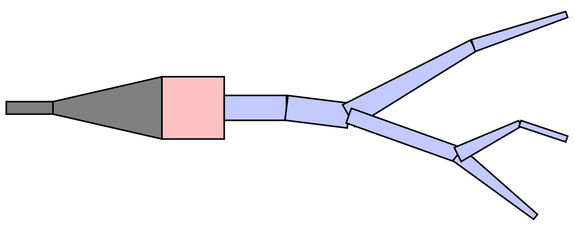
New Features

Morphology

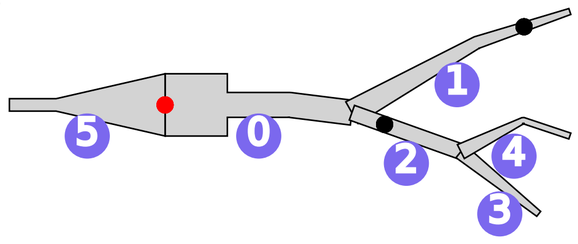
Arbor provides a domain specific language (DSL) for describing regions and locations on morphologies, and a dictionary for associating these descriptions with a string label. The labels are used to refer to regions and locations when setting cell properties and attributes. For example, the membrane capacitance on a region of the cell membrane, or the location of synapse instances.

❶ Consider a 10 segment cell stored in an SWC file. The SWC format defines tags 1, 2, and, 3; corresponding to the soma ■, an axon ■ and a dendrite ■ respectively. We label these regions as follows:

```
labels = arb.label_dict({'soma': '(tag 1)',
                         'axon': '(tag 2)',
                         'dend': '(tag 3)'})
```



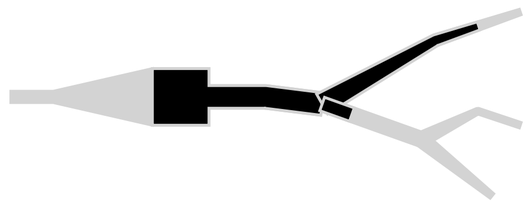
❷ Arbor numbers branches of cells automatically, where a branch is a region between branching points, starting at the proximal point ● of the first segment (usually the soma). For this cell, Arbor generates ❻ branches.



As an example, we define a region to which we want to add certain properties, taking two points ● on the dendrite: one 80% along branch ❶ and the other 30% along branch ❷.

❸ The two points and the proximal point of the soma define an interval where we want to place specific dynamics. By adding a new label we can access this region later.

```
labels['roi'] = '(proximal_interval (sum
↪  (location 1 0.8) (location 2 0.3)))'
```



Now we can 'paint' Hodgkin–Huxley dynamics on the 'roi' region:

```
cell.paint('roi', 'hh')
```

Mechanism Catalogues

We provide an interface to collections of *mechanisms* which describe processes both associated with an area, like ion channels and localised, like synapses. These mechanisms are described in the NMODL DSL and translated into plain or vectorised C++ and CUDA for execution on GPUs. In particular, we currently offer two catalogues. The first, *default*, comprises basic functionality, such as a passive leak current. The second, *allen*, is a collection of mechanisms obtained from the Allen Brain Atlas [4]. We have carefully optimised these, as they form the basis for the V1 network model, which is quite resource intensive. Catalogues are built at compile time and can be accessed at runtime, during the simulation. Furthermore, they can be composed into larger catalogues, where name clashes are avoided via an optional prefix.

Spherical Somata

A common pattern in SWC files is a soma consisting of a single point only. Until recently, Arbor treated these as sphere with the given radius. However, it is not possible to resolve various ambiguities, e.g. where apical dendrites should be attached. Consequently, the interpretation of single point somata has been changed to a cylinder with the same surface area — excluding caps — as the sphere with the given radius. As cable segments might no longer attach directly to the surface, support for these detached segments was added. As a by-product, this change significantly simplified the associated program code. We offer specialised loaders for different flavours of SWC with clear semantics for handling the soma.

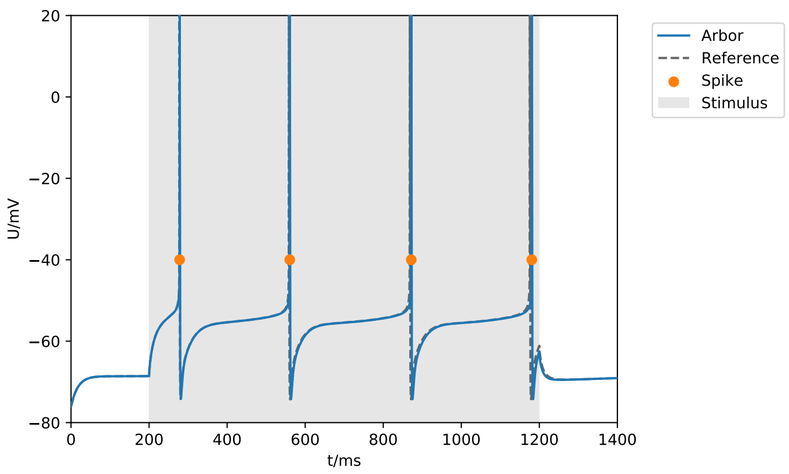
Running an Allen Brain Atlas Model

The code snippet on the right is complete except the parsing and plotting steps; it is available in full together with this poster [5]. While the electro-physiological data is supplied as a JSON file for all models, the lack of a standard schema precludes defining a single procedure for loading.

❶ load SWC structure from a file
❷ assign labels to geometry
❸ build a cell description from labels and geometry
❹ parse the electro-physiological properties supplied in the download and assign to regions
- set physical properties: T, V_m, R_a, C_m
- define ion dynamics and reversal potentials

❺ attach to the soma's center
- current clamp; rectangular stimulus of 150 pA from 200 ms to 1200 ms
- spike detector; triggering at $V = -40\,mV$
- voltage probe; sampling with 200 kHz

❻ convert the cell description into a runnable simulation
❼ set mechanism catalogue comprising the defaults and Allen DB mechanisms.
❽ run the simulation for 1400 ms with time step $\Delta t = 0.005\,ms$



The reference solution was obtained using the allensdk Python package with the default Neuron backend [3]. A minor modification was made to suppress editing of the axon at load time of the geometric information. For comparable simulations, we instead performed this manipulation by hand once and stored the result in the SWC input. We observe a minor deviation from the reference run, which is explained by different discretisations.
The elapsed wall clock times for only the simulation steps are 14.7 s with the code on the right and 121.8 s for the reference solution; yielding a speed-up of 8.25× when using Arbor.

```
import utils, arbor as arb

# ❶ read in geometry
segment_tree = arb.load_swc_allen('cell.swc', no_gaps=False)
morphology = arb.morphology(segment_tree)
# ❷ assign names to regions defined by SWC and center of soma
labels = arb.label_dict({'soma': '(tag 1)', 'axon': '(tag 2)',
                         'dend': '(tag 3)', 'apic': '(tag 4)',
                         'center': '(location 0 0.5)'})
cell = arb.cable_cell(morphology, labels) # see ❸
cell.compartments_length(20) # discretisation strategy: max compartment length
# ❹ load and assign electro-physical parameters
defaults, regions, ions, mechanisms = utils.load_allen_fit('fit.json')
# set defaults and override by region
cell.set_properties(tempK=defaults.tempK, Vm=defaults.Vm,
                    cm=defaults.cm, rL=defaults.rL)
for region, vs in regions:
    cell.paint('"'+region+'"', tempK=vs.tempK, Vm=vs.Vm, cm=vs.cm, rL=vs.rL)
# set reversal potentials
for region, ion, e in ions:
    cell.paint('"'+region+'"', ion, rev_pot=e)
cell.set_ion('ca', int_con=5e-5, ext_con=2.0, method=arb.mechanism('nernst/x=ca'))
# assign ion dynamics
for region, mech, values in mechanisms:
    cell.paint('"'+region+'"', arb.mechanism(mech, values))
# ❺ attach stimulus and spike detector
cell.place('"center"', arb.iclamp(200, 1000, 0.15))
cell.place('"center"', arb.spike_detector(-40))
# ❻ set up runnable simulation
model = arb.single_cell_model(cell)
model.probe('voltage', '"center"', frequency=200000) # see ❺
# ❼ assign catalogues
model.properties.catalogue = arb.allen_catalogue()
model.properties.catalogue.extend(arb.default_catalogue(), '')
# ❽ run simulation and plot results
model.run(tfinal=1400, dt=0.005)
utils.plot_results(model)
```

Conclusion and Outlook

We show that Arbor is a capable, ergonomic and highly performant tool for building simulations of bio-physically detailed neurons. We are working actively to further improve Arbor:

- support for computation of local field potentials (LFP)
- coupling with other simulators such as Nest
- support for NeuroML and Neurolucida cell descriptions
- streamlined NMODL mechanism integration without re-compilation

References

- N. Abi Akar et al. *Arbor — A Morphologically-Detailed Neural Network Simulation Library for Contemporary High-Performance Computing Architectures*. IEEE, 2019.
- Y. N. Billeh et al. *Systematic integration of structural and functional data into multi-scale models of mouse primary visual cortex*. Neuron, 2020.
- N. T. Carnevale and M. L. Hines. *The NEURON book*. Cambridge University Press, 2006.
- E. S. Lein et al. *Genome-wide atlas of gene expression in the adult mouse brain*. Nature, 2007.
- "This Poster". 2020. URL: https://github.com/thorstenhater/bernstein-2020.

Acknowledgements

This research has received funding from the European Union's Horizon 2020 Framework Programme for Research and Innovation under the Specific Grant Agreements No. 720270 (HBP SGA1), No. 785907 (HBP SGA2), and No. 945539 (HBP SGA3).