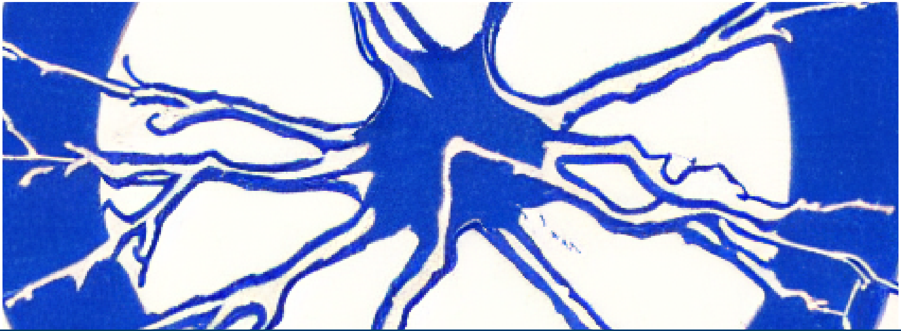


GET YOURSELF (RE)CONNECTED STRUCTURAL PLASTICITY IN ARBOR

TECHNICAL BACKGROUND, FOUNDATIONS, AND FUTURE PLANS

September 2022 | T. Hater | Forschungszentrum Jülich

JÜLICH SUPERCOMPUTING CENTRE

Topics

abc

- Technical background of Arbor’s connectivity.
- Structural Plasticity in Arbor.
- Demo!
- Calcium tracking.
- Diffusion.
- Future Plans
  - A DSL for connectivity.
  - Dynamic matching.
  - ...

From Recipe to Network

…and beyond

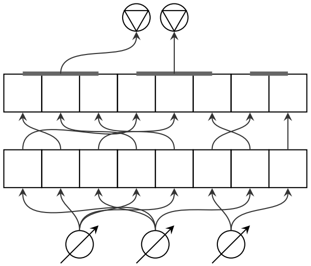
```
import arbor as A

class Recipe(A.recipe):
    def connections_on(self, target_gid):
        return [A.connection((source_gid,   # cell from which spikes are coming in
                               "source"),   # label of the detector on that cell
                              "synapse",    # label of the synapse on our cell
                              weight,       # connection weight
                              delay),       # axonal + synaptic delay
                # ...
                ]
```

- gids are unsigned numbers to label a cell
- "label"s are used to designate items on cells
- each cell may return different sources based on its own gid
- there are similar mechanisms for local generators and gap junctions

Transmission of Spikes

- Communication pattern: collect all spikes on all receiving processes
  - Spikes are sent as (`source, weight, time`)
  - These are locally sorted, `AllGathered`, and thus globally sorted!
  - Receivers filter by 'their' sources (`bsearch`)



- Conncurrent communication and computation based on decoupling
  - Pick the global minimum τ over `delay`; half for double-buffering.
  - No event at t can influence anything before $t + \tau$

Local Filtering

…and Plasticity

- Each process holds a list of sources it's connected to; plus their associated targets.
- This table is built using the `connections_on` data
- Thus, to modify the connections, we just need to tweak this table.
- However, it's simpler, safer, and faster to completely rebuild.
- This is a *purely local*, independent process.

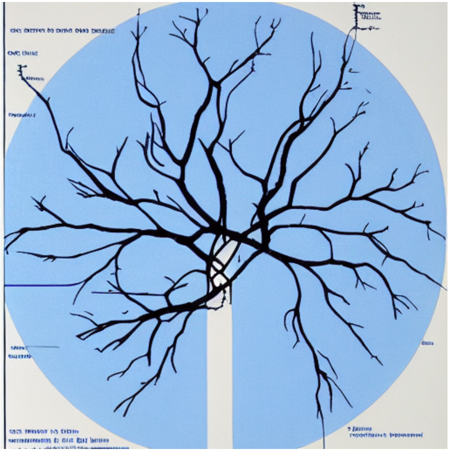
Structural Plasticity

Bare Bones Example

```python
import arbor as A
# contains initial network
rec = my_recipe()
# run the network for 5ms
sim = A.simulation(rec)
sim.run(5)
# here we update the recipe, or create a new one
rec = ...
# rebuild connection table
sim.update(rec)
# ... and continue simulating
sim.run(5)
```

- Add/delete synaptic connections and event generators and adjust their weights.
- Not changed: Gap Junctions, detectors, cells, parameters, morphology, and synapses.

Demo!



Calcium Tracking

```python
class Recipe(A.recipe):
    # ...
    def probes_on(self, gid):
        if gid == 0:
            # Probe ca concentration on a given set of locations (here: all terminals)
            return [A.cable_probe_ion_int_concentration('(terminal)', 'ca')]
        elif gid == 42:
            # Probe ca on every cable in the cell
            return [A.cable_probe_ion_int_concentration_cell('ca')]
        else:
            return []

rec = Recipe()
sim = simulation(rec)
hd0 = sim.sample((0, 0), A.regular_schedule(0.1))  # 1st probe on gid=0: terminals
hd1 = sim.sample((42, 0), A.regular_schedule(0.1)) # 1st probe on gid=42: all cables
sim.run(23)
results = sim.samples(hd0)
# ...
```

Diffusing Molecules

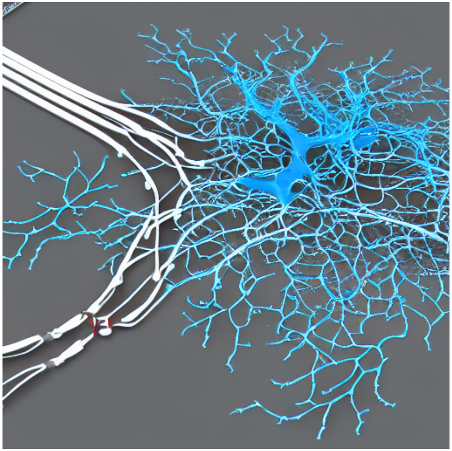
```
tree = load_swc_arbor(...)
labels = A.label_dict()
decor = (A.decor()
      .set_ion("na", int_con=0.0, diff=0.005)                       # Add diffusion to Sodium
      .place("(location 0 0.5)", A.synapse("inject/x=na"), "Zap") # Synapse to produce Na
      .paint("(all)", A.density("decay/x=na")))                     # Na decays exponentially

cell = A.cable_cell(tree, labels, decor)
```

- No longer part of the cable model, but separate!
  - Breaks thin shell approximation,
  - no reset like for internal/external concentration X_i and X_o,
  - $\partial_t X_d$ does not contribute to current (neither do $\partial_t X_i$ and $\partial_t X_o$).
- Can be used as
  - a memory mechanism,
  - transmitter between spatial separated synapses.
- Can be probed just like X_i and X_o.

JÜLICH Forschungszentrum JÜLICH SUPERCOMPUTING CENTRE

Future Plans



Connectivity

- Currently connectivity is defined via lists of indices and labels.
- UX does not scale and is not ergonomic even at small scales.
- Thus: define a high-level connection DSL, loosely inspired by NMLlite.

```
(connect
  (from                          ; sources
    (all)                        ; all gids
    (is-kind spike-detector))    ; source labels
  (to                            ; targets
    (distance-lt 0.42)           ; L2 distance src <-> tgt
    (is-kind 'expsyn')))         ; target labels
```

- Design is work in progress, phase space is huge.
- Probabilistic connectivity, tag based queries, relational algebra, …

Selecting Target/Source Pairs

- (Re)Connecting cells requires spatial queries across multiple ranks in a parallel computation.
- Investigate algorithms for finding partners, starting simple.
- More scalable approaches like Rinke et al ’17 might be interesting.

Possible Explorations

- Modifying parameters during simulation.
- Tweaking the morphology.
- Re-connecting Gap Junctions (when we have implemented a distributed algorithm)

Disclaimer

These are hard(er) problems and might not be feasible to implement.

Summary

- Plasticity is an active driver for Arbor development.
- We already have solid foundations in place.
- From here, your input is needed.

Summary

- Plasticity is an active driver for Arbor development.
- We already have solid foundations in place.
- From here, your input is needed.

Thank you!
Questions?

t.hater@fz-juelich.de