WHAT IS ARBOR?

Arbor is a library for implementing performance portable network simulations of multi-compartment neuron models.

- Simulate large networks of morphologically-detailed, spiking neurons

- Library: you control your program/workflow. Interoperable.

- Portable: scientific description is separate from execution instructions. E.g. run one scientific description on laptop CPU, GPU cluster or future hardware.

- *Performance* portable: add optimized backends for new computer architectures. Currently supported:
  - Distributed parallelism using MPI
  - CUDA backend for NVIDIA and AMD GPUs
  - Vectorized backends for x86-64 (KNL, AVX, AVX2) and Arm64 (NEON, SVE) intrinsics

- Executes on all HPC systems in the HBP (and outside).

arbor

JÜLICH
Forschungszentrum

WHO IS ARBOR?

Open development style through Github

- Issues, PR workflow, Discussions, Slack/Gitter
- Code quality: PR review, unit testing, CI at Github and CSCS

Core contributors

- Ben Cumming
- Nora Abi Akar
- Stuart Yates
- Anne Küsters
- Thorsten Hater
- Brent Huisman

Website: `arbor-sim.org`

CSCS
Centro Svizzero di Calcolo Scientifico
Swiss National Supercomputing Centre

JÜLICH Forschungszentrum | JÜLICH SUPERCOMPUTING CENTRE

arbor

JÜLICH Forschungszentrum

ARBOR STATUS

- Latest release: v0.5.2
- 42 Github forks
- 1300+ commits to main branch
- loc: C++ header: 68k, C++: 68k, Python: 16k, reStructuredText: 8k
- 24 contributors, from 9+ institutions

Ongoing collaborations:

- FIPPA - extend Arbor by key plasticity processes to simulate and analyze the long-term adaptive dynamics of large-scale, morphologically-detailed neuronal networks
- Arborio - large-scale model of the inferior olive of the cerebellar complex as a case study
- LFPy - investigating Arbor as possible backend
- Co-simulation - Nest, Elephant, TVB

arbor

JÜLICH
Forschungszentrum

BUILDING ARBOR
begins with the computational model of neurons

- Arbor simulates networks of multi-compartment neurons

- Neurons: approximated by axonal delay, synaptic functions and a set of cables connected in a tree.

- Cables: characterized as electrical compartments (frustums) composed of ion channels, cable resistance and capacitance.

- Neurons represented as sparse, close-to-band matrices to be solved (e.g. by Hines solver) against known current states due to synaptic conductance.

- Network and spike exchange between neurons at synapses are represented by concatenations of matrices.

arbor

JÜLICH
Forschungszentrum
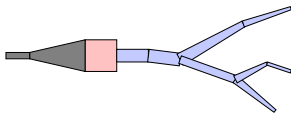
USER DESIGN
Describe the neuroscience first ...

Cells

- Cells represent the smallest model to be simulated
- Cells are the smallest unit of work distributed across processes
- Types:
    - Cable cells
    - Leaky integrate-and-fire cells
    - Spiking cells
    - (Benchmark cells)

Recipes

- Recipes describes models in a cell-oriented manner and supplies methods to:
    - Map global cell identifier gid to cell type
    - Describe cells (Cable cell 123, what is it's morphology?)
    - List all connections from other cells that terminate on a cell
- Advantage: parallel instantiation of cell data

USER DESIGN

**... then translate it into execution.**

Cell groups

- Cell groups represent a collection of cells of the same type together with implementation of their simulation
- Partitioning into cell groups provided by decomposition
- A **simulation** manages instantiation of model and scheduling of spike exchange as well as integration for each cell group

Mechanisms

- In a recipe, mechanisms are specifications of ion channel and synapse dynamics. Advantage: parallel instantiation of cell data.

  Implementations of mechanisms:

- Hand-coded for CPU/GPU execution
- A translator (modcc) compiles a subset of NMODL to architecture-optimized vectorized C++ or CUDA
- Soon: Arblang

INTERNAL DESIGN

Programming interface ensures extensibility
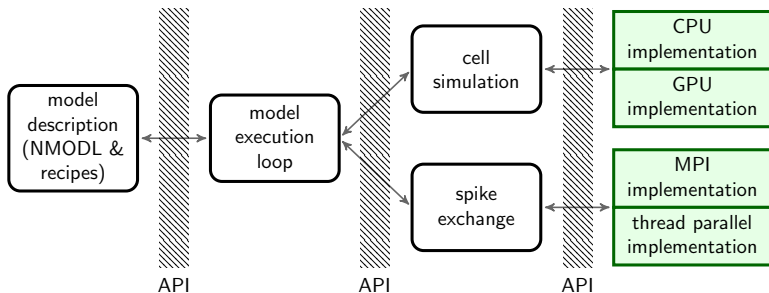
- Components can be substituted according to the internal API.

DESIGN
Summary

Arbor internal model:

- Internal API decouples model description, execution, spike exchange and cell simulation

- Computational work is hidden in pluggable backends, allowing automatic generation for different architectures

User models are composed of:

- Cells representing the smallest unit of computation

- Recipes representing a parallelizable set of neuron construction and connections

- Mechanisms representing ion channel and synapse dynamics

arbor

JÜLICH
Forschungszentrum

INTEROPERABILITY, PORTABILITY AND EXTENSIBILITY

Why are they relevant to a computational neuroscientist?

- Why a library?
    - Makes Arbor interoperable with other tools and your way of working.

- What is portability?
    - Write your science and let Arbor worry about how to efficiently execute it.

- What is performance portability?
    - You can extend Arbor to take advantage of new hardware without having to modify your scientific description. Your scientific description will continue to work.

JÜLICH
Forschungszentrum

WHAT'S NEW?

- Expanded set of tutorials
- CI significantly expanded
  - Automated building of Python and Spack packages
  - Soon: Ebrains CD
- File format compatibility: cell morphologies
  - SWC
  - NeuroML
  - Neurolucida ASCII
  - Arbor Cable Cell
- Arbor GUI
  - Focussed on cell design, decoration
  - Can run single cell model simulations

⩗arbor

JÜLICH
Forschungszentrum

WHAT'S COOKING?

- Arbor mechanism description language (codename ARBLANG)
    - Declarative
    - Simple, extendable and maintainable
    - Capable of performing powerful optimizations on the source code
    - Make support in other simulators easy, make Arblang translatable to/from similar DSLs (eg: NEUROML/LEMS, NMODL, SBML)
- Crack the nut of distributed gap junctions
    - MSc will study Wave Relaxation method this summer
    - Arborio is investigating *multi*-GPU cell groups
- Implement synaptic plasticity, synaptic scaling, and structural plasticity
    - FIPPA
- LFP estimation
    - Arborio

JÜLICH
Forschungszentrum

WRAP UP
Questions?

- Web: `arbor-sim.org`
- Docs: `docs.arbor-sim.org`
- Community: `github.com/arbor-sim/arbor/discussions`

EBRAINS   Human Brain Project

arbor

JÜLICH
Forschungszentrum

DEMO TIME!

- We'll be doing the network ring tutorial:

  `docs.arbor-sim.org/en/latest/tutorial/network_ring.html`

- Connect at `https://lab.ebrains.eu`

- `pip install arbor`

- `wget https://github.com/arbor-sim/arbor/raw/v0.5.2/python/example/network_ring.py`

STEP 0

- Documentation at `docs.arbor-sim.org`
    - 'Concepts' for conceptual explanation
    - Python and C++ sections detail the matching conceptual pages
- Today's plan
    - Make a cell
    - Make a recipe
    - Run
    - Visualize results

arbor

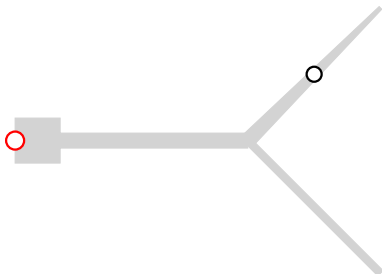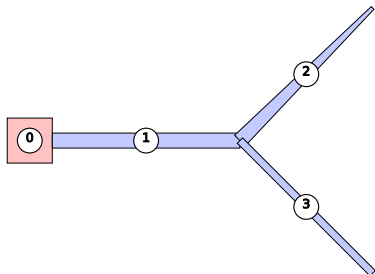**JÜLICH**
Forschungszentrum

STEP 1

Make a cell

Figure: Left: Make a morphology of 4 segments: a soma (pink) and a branched dendrite (light blue). Right: We'll create labels for the root (red) and a synapse_site (black).

STEP 2
Make a network

- Cells are the basic building blocks in Arbor
- Recipes tie them together
- You'll make your own recipe by inheriting from `arbor.recipe`
- Some member functions need to be overridden

arbor

JÜLICH
Forschungszentrum

STEP 3

Make a simulation

- Describe
  1. context (what hardware do you have?)
  2. domain decomposition (how to use the hardware)
  3. simulation
- Start with an `arbor.simulation` object, and see what you need for it.
- (defaults are OK)
- We need to store handles to the samplers, because later we'll use `arbor.simulation.samples()` to retrieve results.

STEP 4
Show your results!

- We can extract results from
  - `arbor.simulation.spikes`
  - `arbor.simulation.samples`

- Print or plot
  - Suggestion: Pandas and Seaborn
  - `pip install pandas seaborn`

⚖️arbor

JÜLICH
Forschungszentrum

THE END
Questions?

- Web: `arbor-sim.org`
- Docs: `docs.arbor-sim.org`
- Contact: `contact@arbor-sim.org`
- Community: `github.com/arbor-sim/arbor/discussions`

EBRAINS   HBP Human Brain Project

arbor

JÜLICH
Forschungszentrum