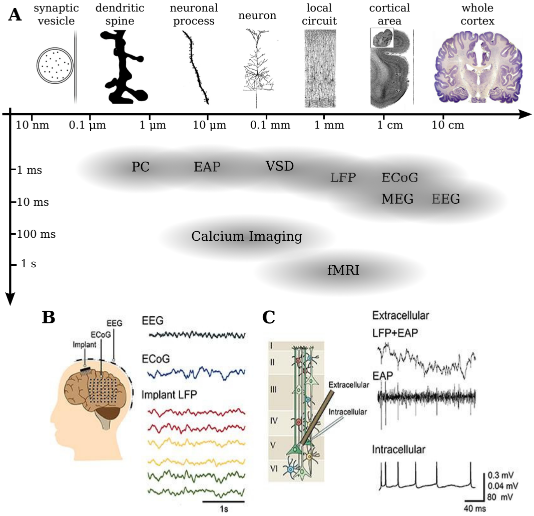
NEST TUTORIAL - EITN FALL SCHOOL 2024

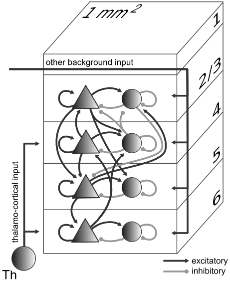
17 October 2024 | Barna Zajzon | IAS-6; Jülich Research Center

Multi-scale brain structure and dynamics



The microcircuit model

- 10^5 identical leaky-integrate and fire neurons
- $3 \cdot 10^8$ exponentially decaying synaptic currents
- Four layers with one excitatory and one inhibitory population each
- Size of populations and connection probabilities deduced from anatomical data

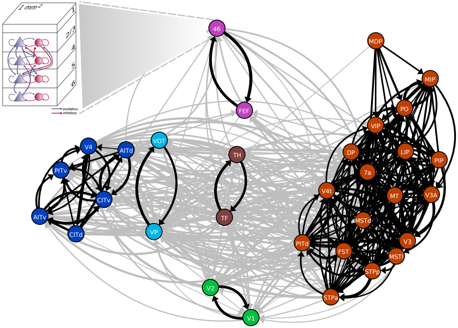


- Asynchronous irregular and cell-type specific firing rates
- Thalamic stimulation elicits flow of activity through cortical layers

Potjans and Diesmann (2014) The Cell-Type Specific Cortical Microcircuit: Relating Structure and Activity in a Full-Scale Spiking Network Model. Cerebral Cortex 24(3):785-806

The multi-area model

- Full-density model of macaque visual cortex
- Axonal tracing data from the CoCoMac database, which are systematically refined using dynamical constraints
- Stable asynchronous irregular ground state

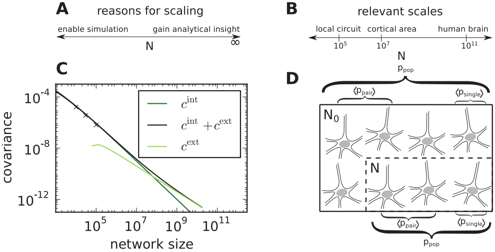


- Produces realistic spiking statistics in V1
- Functional connectivity compares to fMRI measurements

Schmidt et al. (2018) Multi-scale account of the network structure of macaque visual cortex. Brain Structure and Function 223(3):1409-1435
Schmidt et al. (2018) A multi-scale layer-resolved spiking network model of resting-state dynamics in macaque visual cortical areas. PLOS CB 14(10):e1006359

Importance of the correct network size

- Under which conditions can a small network represent a sub-sampled larger network?
- Analyzes scalability of binary and LIF neuron networks



- Mean activity can be preserved by adjusting the mean and variance of the input
- Temporal structure of pairwise averaged correlations depends on the effective connectivity and cannot always be preserved

van Albada et al. (2015) Scalability of Asynchronous Networks Is Limited by One-to-One Mapping between Effective Connectivity and Correlations. PLOS CB 11(9):e1004490

NEST = NEural Simulation Tool

- Focus on the dynamics, size and structure of neural systems rather than on the exact morphology of individual neurons
- Phenomenological synapse models (STDP, STP)
  + gap junctions, neuromodulation and structural plasticity
- Frameworks for rate models and binary neurons
- Support for neuroscience interfaces (MUSIC, libneurosim)

- Highly efficient C++ core with a Python frontend
- Hybrid parallelization (OpenMP+MPI)
- Same code from laptops to supercomputers

NEST design goals

- NEST development is always driven by scientific needs
- High accuracy and flexibility
  - Exact integration is used for suitable neuron models
  - Spike interaction in continuous time available for suitable neuron models
  - Extremely scalable: same code from laptop to supercomputers
- Constant quality assurance
  - Automated unit test suite included in NEST build
  - Continuous integration for all repository checkins
  - Peer review for all code contributions

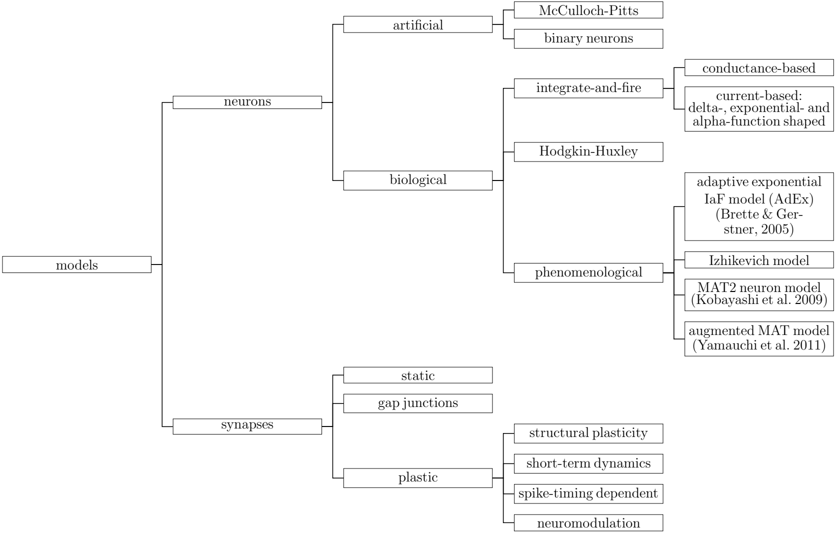
Main components of a NEST simulation

- **Nodes**
  - Neurons – Devices (– Sub-networks)
  - Have dynamic state variable(s) that changes over time ($V_{\mathrm{m}}(t)$)
  - Can be affected by events (spikes)

- **Events**
  - Pieces of information of a particular type (e.g., spike, voltage or current event)
  - Recording devices: `'spike_recorder'`,`'voltmeter'`,`'multimeter'`

- **Connections**
  - Communication channels for the exchange of events
  - Directed (from source node to target node)
  - Weighted (how strongly does an event influence the target node)
  - Delayed (length of transmission duration between source and target)
  - Connections are created using one global `Connect` function

Neuron and synapse models in NEST



Event-driven vs. time-driven simulation

	Event-driven	Time-driven
Pros	▪ more efficient for low input rates ▪ ‘correct’ solution for invertible neuron models	▪ more efficient for high input rates ▪ works for all neuron models ▪ scales well
Cons	▪ only works for neurons with invertible dynamics ▪ event queue does not scale well	▪ only ‘approximate’ solution even for analytically solvable models ▪ spikes can be missed due to discrete sampling of membrane potential

Event-driven vs. time-driven

NEST uses a hybrid approach to simulation

- Input events to neurons are frequent: time-driven algorithm
  - If the dynamics is nonlinear, we need a numerical method to solve it, e.g.:
    - Forward Euler: $y([i+1]h) = y(ih) + h \cdot \dot{y}(ih)$
    - Runge-Kutta (k-th order)
    - Runge-Kutte-Fehlberg with adaptive step size
    - ...

  $\rightarrow$ Use a pre-implemented solver, for example, from the GNU Scientific Library (GSL)

  - If the dynamics is linear (e.g. leaky integrate-and-fire), we can solve it exactly

Event-driven vs. time-driven

NEST uses a hybrid approach to simulation

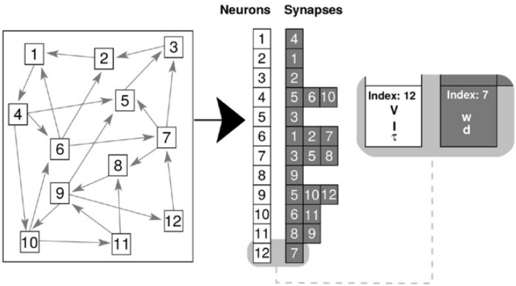
- Input events to neurons are frequent: time-driven algorithm
  - If the dynamics is nonlinear, we need a numerical method to solve it, e.g.:
    - Forward Euler: $y([i+1]h) = y(ih) + h \cdot \dot{y}(ih)$
    - Runge-Kutta (k-th order)
    - Runge-Kutte-Fehlberg with adaptive step size
    - ...

  $\rightarrow$ Use a pre-implemented solver, for example, from the GNU Scientific Library (GSL)

  - If the dynamics is linear (e.g. leaky integrate-and-fire), we can solve it exactly

- Events at synapses are rare: event driven component
  - Exception: gap junctions

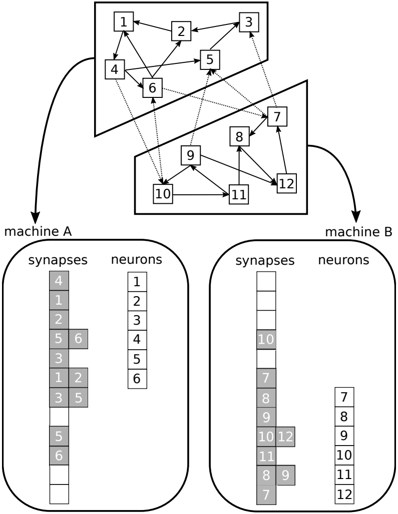
Representation of network structure: serial



- Each neuron and synapse maintains its own parameters
- Synapses save the index of the target neuron

Representation of network structure: distributed

- modulo operation distributes neurons
- one target list for every neuron on each machine
- synapse stored on machine that hosts the target neuron
- connections are established on each machine and the connectivity information subsequently propagated to other machines
  - → wiring is a parallelizable task



Creating custom models

- Discuss with developers via user mailing list
  - If your idea makes sense
  - If it has not yet been implemented
  - Start from most similar existing model
  - It may end up in a release!

- Extension modules (C++ knowledge required)
  - Loaded dynamically

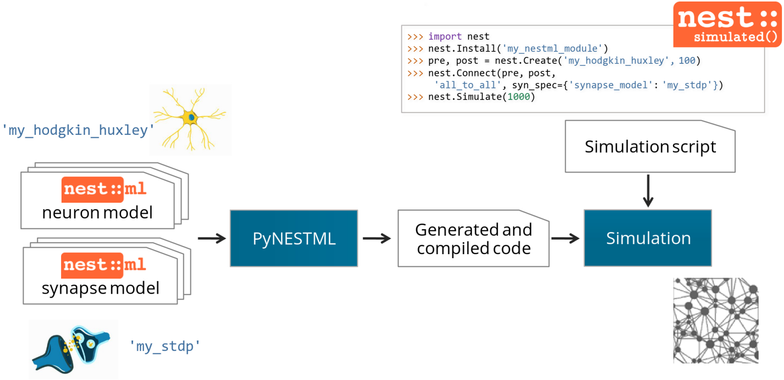
- Inside NEST (C++ knowledge required)
  - Re-compile and re-install after each change

Recommended:

- **NESTML (NEST Modeling Language)**

NESTML

NESTML is a domain-specific language for neuron and synapse models



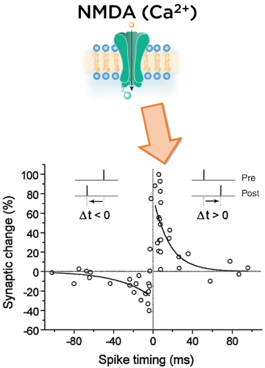
NESTML: Design principles

- Concise; low on boilerplate
- Speak in the vernacular of the neuroscientist (keywords such as `neuron`, `synapse`)
- Easy (dynamical) equation handling coupled with imperative-style programming (`if V_m >= threshold:  ...`)

NESTML comes with a code generation toolbox.

- Code generation (model definition but not instantiation)
- Automated ODE analysis and solver selection
- Flexible addition of targets using Jinja2 templates

Creating custom models with NESTML

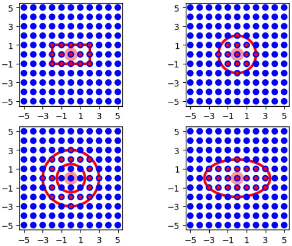


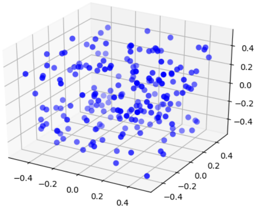

```
synapse stdp:
  state:
    w real = 1
    tr_post real = 0
    tr_pre real = 0

  equations:
    tr_pre' = -tr_pre / tau_tr
    tr_post' = -tr_post / tau_tr

  input:
    pre_spikes real <- spike
    post_spikes real <- spike

  onReceive(pre_spikes):
    w -= alpha * tr_post          # depress synapse
    tr_pre += 1                   # update presynaptic trace
    deliver_spike(w, delay)       # to postsynaptic partner

  onReceive(post_spikes):
    w += alpha * tr_pre           # potentiate synapse
    tr_post += 1                  # update postsynaptic trace

  parameters:
    delay ms = 1 ms               # dendritic delay
    tau_tr ms = 50 ms             # pre/post trace time const.
    alpha real = .02              # learning rate
```

Image credit: (c) S.B.C. Lehmann, FZ Jülich, (m) C. Linssen; Bi & Poo 2001

Topologically structured networks

- Functionality
  - Lay out elements on grids or at arbitrary points in space (2D or 3D)
  - Elements can be neurons or combinations of neurons and devices
  - Connect neurons in a position- and distance-dependent manner
  - Set periodic boundary conditions
  - Choose whether to allow self-connections (autapses) or multiple connections (multapses)
  - Distance-dependent or random weights and delays





Why should I use NEST?

1. NEST provides over 50 neuron models
2. NEST provides over 10 synapse models, including short-term plasticity (Tsodyks & Markram) and different variants of STDP
3. NEST provides many examples that help you getting started
4. NEST lets you inspect and modify the state of each neuron and each connection at any time during a simulation
5. NEST is fast and memory efficient; it makes best use of your multi-core computer and compute clusters
6. NEST has a large and experienced developer community
7. NEST was first released in 1994 under the name SYNOD and has been extended and improved ever since
8. NEST is open source software and is licensed under the GPL 2

Acknowledgments

This presentation is based on previous work by many people.

- Hannah Bos
- David Dahmen
- Moritz Deger
- Jochen Martin Eppler
- Espen Hagen
- Abigail Morrison
- Jannis Schuecker
- Johanna Senk
- Tom Tetzlaff
- Sacha van Albada
- Charl Linssen
- Anno Kurth

Getting help

Within Python:

```
nest.help('iaf_psc_exp')
nest.help('Connect')
```

Online documentation:

NEST: `https://nest-simulator.readthedocs.io/`
NESTML: `https://nestml.readthedocs.io/en/latest/`

Community:

- NEST and NESTML user mailing lists
- Bi-weekly open video conference
- `http://github.com/nest/nest-simulator/`
- `https://github.com/nest/nestml`
- Annual NEST Conference: a forum for users and developers

Please tell us about problems. We can only fix what we know of!

Now hands-on

1. **Check out the Github repo on EBRAINS**

2. **Get the exercises**
   - Go to `https://github.com/zbarni/nest_tutorial_eitn_24`
   - Download as zip (or clone)

3. **Enjoy the ride**
   - Open `0_hello_world.ipynb` for a first glance
   - Get started with `1_first_steps.ipynb`