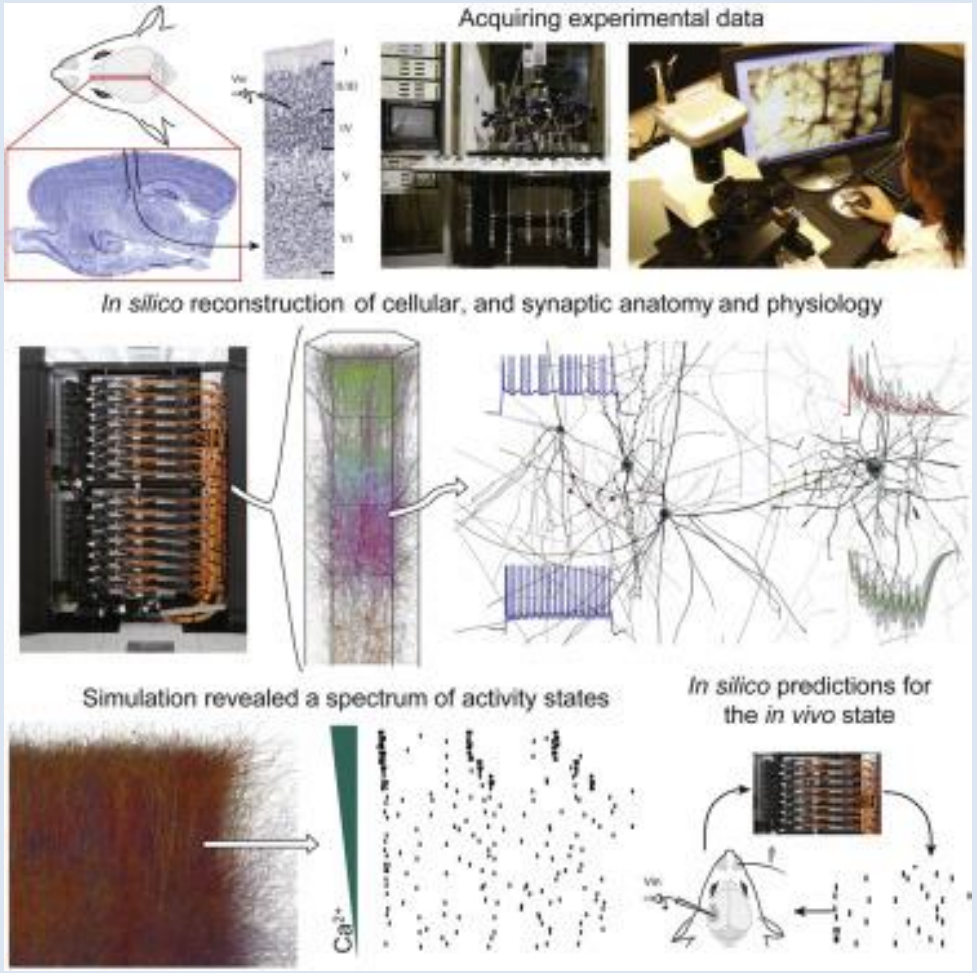
Alexander Peyser (FZJ, JSC) / Hans Ekkehard Plesser (NMBU)

TK1.2 NEUROSIMULATION (NEST)

The brain: irreducible complexity

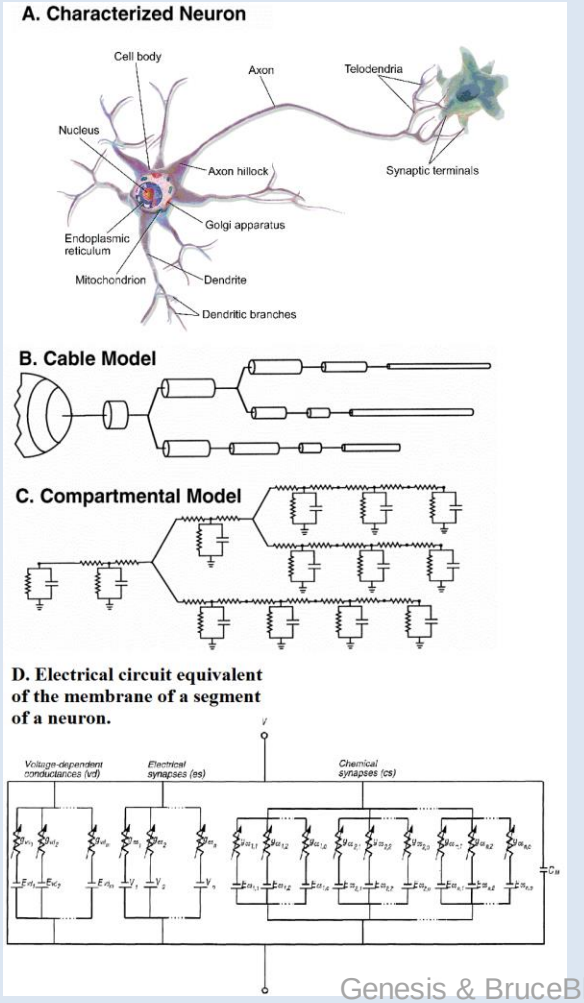
- The brain is multiscale
  - Molecules, cells, columns, regions, lobes, entire brains
  - Space: nanometer → meter
  - Time: millisecond → decades
- 10^{11} cells, each with
  - 10^4 incoming connections
  - a complex of dendrites
  - an integrating cell body
  - an axon which produces firing events at synapses
  - which can produce discrete events (spikes) needed by any downstream neuron, which may be anywhere in the brain
- Models constructed at different scales
  - STEPS: ionic diffusion
  - Arbor (NestMC): morphologically detailed neurons
  - NEST: cell & cell populations
  - The Virtual Brain: coupled brain regions



Markram et al (2015) (C) Elsevier doi:10.1016/j.cell.2015.09.029

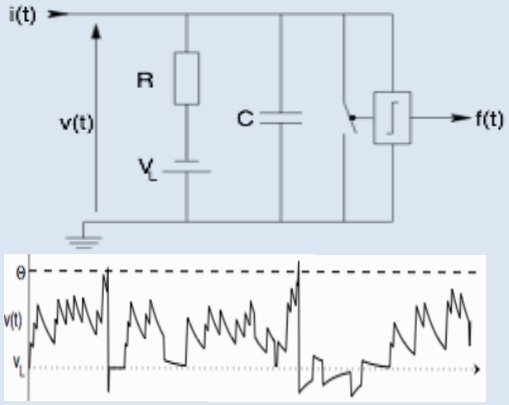
Simulations: reducing the complexity

Detailed Neuron



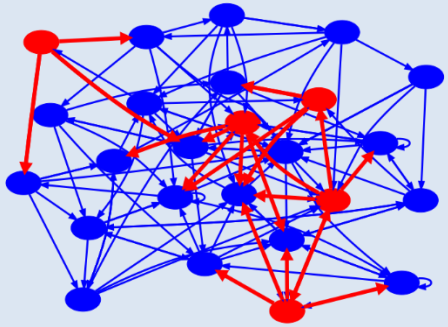
Genesis & BruceBlaus

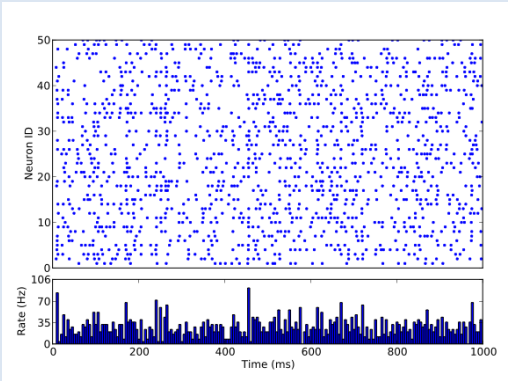
Point Neuron



$$C\dot{V} = -\frac{(V - V_L)}{R} + i(t)$$

Neural Network





with global spike exchange

Relevant simulation time scales

- Internal neuronal dynamics: 0.1–1 ms (time step)
  - Time for computation of kernels
- Propagation delays: 1–10 ms
  - Time for global exchange of spike events
- Network dynamics: 10–1000 ms (EEG signals)
  - Time of analysis and multiscale coupling
- Short-term plasticity: minutes
  - Simulation time for changing behavior
- Learning: minutes to days (... and counting ...)
  - Goal for simulation lengths

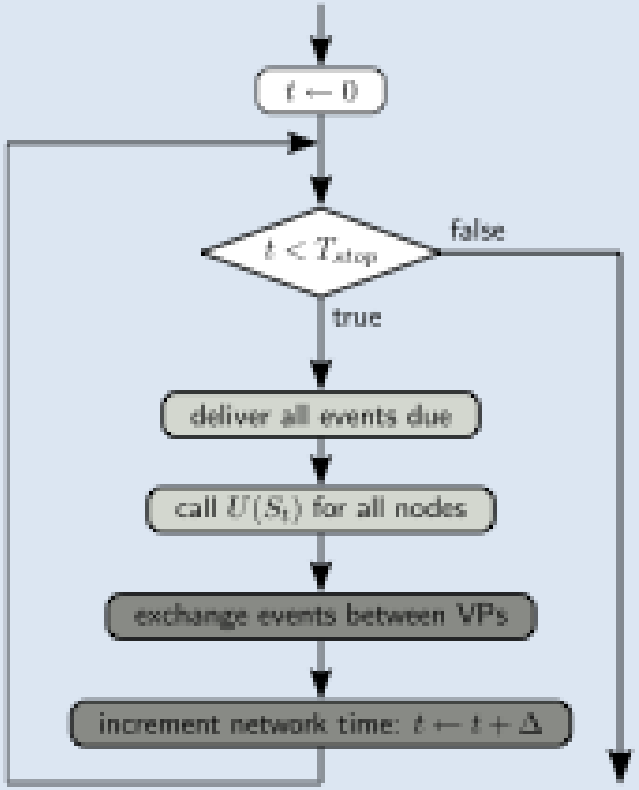
Typical simulation run-times today:
100–1000 times slower than simulated time

Computational demands for point neuron simulations

- Average neuron
  - 10^4 in/out connections
  - 10 spikes/s
- Total distributed over cluster (NEST)
  - 200 million cells @ 1k: 200 GB
  - 2 trillion edges @ 48 bytes: 96 TB
  - 2 billion spikes/s @ 8 bytes: 16 GB/s-simulated

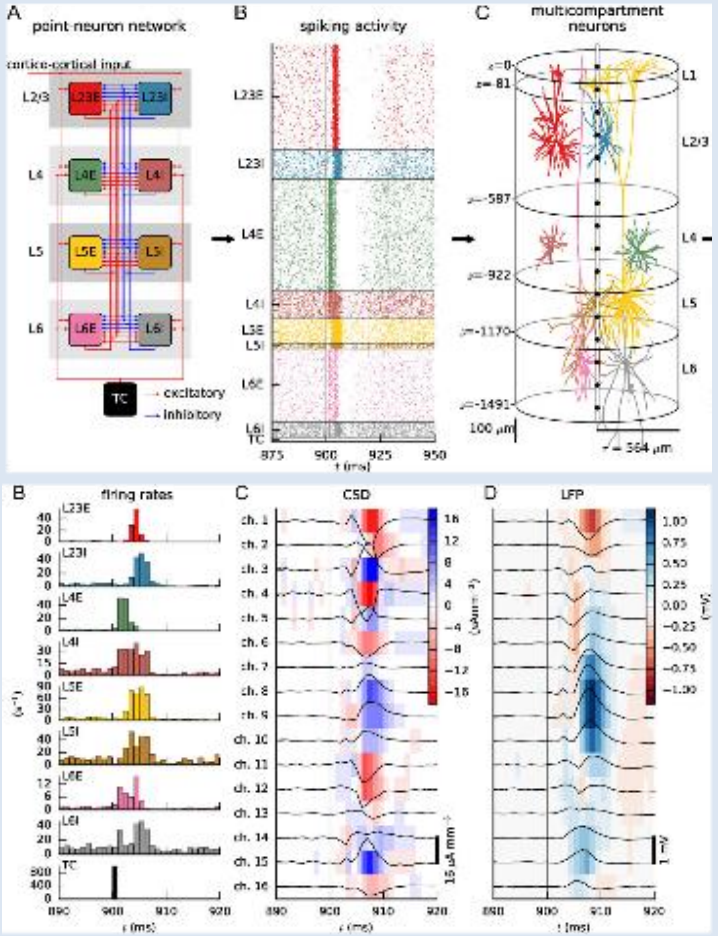
NEST simulation structure

- Network construction
  - Programmatic construction of connections and neurons: Python & C++
- Simulation
  - Runs of fixed time: C++
    - Set of time steps (~1 ms) computing kernels and state updates
    - Exchange of spikes (every ~10 ms)
    - Distribution of spikes
    - Communication with other modules/IO of spikes, voltages, currents
  - Python code before returning to next run



Multiscale and modular

- Simulate a large point neuron network using NEST:
  - random memory access
  - non-vectorized
- Simulate sub-network for electrical activity measurements using Arbor (NestMC)
  - computationally large models for dipoles
  - vectorized
- Analyze in-line for overall local field potentials and spiking activity (LFPy)



Hagen, Damen et al (in press)

Tools

- NEST [https://github.com/nest/nest-simulator]
  - C++03, MPI, OpenMP, Python >=3.4
  - Memory bandwidth limited
  - Communication bandwidth negligible, but need low latency for synchronization points
  - Fat node, traditional cores
- Arbor (NestMC) [https://github.com/eth-cscs/nestmc-proto]
  - C++11/14, MPI, TBB, Python >= 3.4
  - Vectorized for KNL, GPUs
    - could develop FPGA and other architecture back-ends
  - Communication bandwidth negligible
- Python LFPy computations
  - Python >=3.4, Numpy, Scipy, Matplotlib
  - Fast I/O

Changes needed

- Coupling of simulators with each other and analyzer
  - Current POC: NEST is the network, single neurons are simulated in NEURON post network activity, LFP done post single neuron calculations
- New vectorization backends for Arbor (NestMC)?
  - Current KNL, GPUs, FPGAs if resources
- Integration with interactive visualizer?

Issues for implementers

- Allocators
  - NEST makes a large number of malloc calls in the construction phase, which limits scaling with OpenMP
  - Solved with tcmalloc, jemalloc, tbb allocators
- NEST bottleneck:
  - main memory to all cores
  - MPI barriers
- Arbor (NestMC) bottleneck:
  - global memory to vectorization units

Some of the people



And many more…

Thank you

Questions?