The **Myriad** simulator: parallel computation for densely integrated models

Pedro Rittner, Andrew J. Davies, Thomas A. Cleland

1Dept. of Psychology, Cornell University, Ithaca, NY

### Why another simulator?

- Larger network models, especially many biophysical models, often have complex inter- and intra-neuron interactions with multiple non-linear, cyclic dependencies. Such densely integrated circuits are poorly optimized in general purpose simulators, devolving into solving multiple stiff equations linearly, step by step.
- Models often trade off performance for biophysical accuracy due to general purpose simulators’ implementations being difficult to parallelize, often requiring special coding to achieve limited multiprocessing capabilities.
- **Myriad** separates model design and code optimization into two discrete systems connected by powerful code generation middleware, enabling users to fully utilize multithreading capabilities on commodity hardware, GPUs, and clusters.

### Design overview of the **Myriad** simulator

- Idiomatic Python 3 with optional C code specified verbatim.
- High-level abstractions for neurons, 'sections', synapses, and network properties.
- Mechanisms and other elements (particles, ions, channels, pumps, etc.) are user-definable with object-based inheritance (e.g., channels inherit properties based on the ions that they flux).
- Simulations are represented as objects to facilitate iterative parameter searches and reproducibility of results.
- Inheritance functionality via Python’s native object system
  - **Automatic access** to properties of parent component
  - Functionality can be **extended & overridden** at will
- Mako templates for converting user-defined objects into C and Python-compatible structures for use in simulation.
- Removes all hierarchy from compartmental models, recognizing only two computational elements:
  - **Compartment** (isometric) with passive properties
  - **Mechanisms** that connect exactly two compartments in exactly one direction. (Connections to a common extracellular medium are a special case of this).
- AST-to-AST translation of Python code to C for compiling.
- Each compartment or mechanism is treated as a separate computational element, enabling parallel execution on separate CPU threads or GPU CUDA cores. Parameters are updated via shared memory access.
- Synchronization via barrier intrinsics at each timestep ensures safe access to shared data without races/blocking.
- Separate compilation phase enables compilers to aggressively inline mechanism and compartment functions.
- POSIX message queues used as signaling method for communicating data between the front-end and the simulation executable via Unix Domain Sockets.

### Granularity enables highly flexible model design

- 100 Hodgkin-Huxley neurons coupled with inhibitory synapses form an internunet network generate (IRG) oscillatory network.
- **User extension example:** AMPA Synapse with STDP

### Simulation examples

- **Granularity enables highly flexible model design**
  - Neuron receiving inputs from two presynaptic neurons: one normal synapse from a myelinated axon and one specialized synapse from an unmyelinated axon in which the user wishes to model the synaptic cleft explicitly as a separate compartment.
  - Synapse mechanisms can be modeled as aggregate transfer functions (upper), by separating out neurotransmitter release sequences from receptor binding and activation processes (lower) or by other user-defined schemes.

### Planned Extensions

- Docker support for facilitating deployments is planned on release.
- Implement simulation governor to run multiple instances in series or in parallel (e.g., on distributed-architecture GPU clusters), to support parameter exploration and algorithmic optimization.
- **Myriad** is an arbitrarily programmable GPU-enabled computational framework that is in principle as appropriate for (e.g.) 3-D spatial diffusion models as for neuronal modeling. Assess Myriad’s utility for these different applications, and their synthesis.
- Extend Myriad to a nonuniform memory access architecture to support multiple CUDA cards on a single high-speed bus (NVLink).

### References & Acknowledgments

- This research was supported by an equipment grant from the NVIDIA Corporation.