Poster presentation

A general biological simulator: the multiscale object oriented simulation environment, MOOSE

Subhasis Ray, Raamesh Deshpande, Niraj Dudani and Upinder S Bhalla*

Address: National Centre for Biological Sciences, Tata Institute of Fundamental Research, Bangalore, Karnataka, 560065, India

Email: Upinder S Bhalla* - bhalla@ncbs.res.in

* Corresponding author

Introduction

The MOOSE project is a next-generation simulator for large, detailed models that span multiple scales of computational biology. It does so using a modular, object oriented data representation combined with an event-driven messaging architecture. This framework is good both for intuitive representation of biological concepts, and for coupling disparate simulation engines. Thus MOOSE is not just a simulator, but a simulation environment. It reimplements, and is backward compatible with, the GENESIS [1] simulator capabilities, but is designed for multiprocessing, multiple scripting languages, interoperability, and scaling to very large simulations.

MOOSE core

Biological entities are implemented in MOOSE as objects whose fields include state variables and whose functions include the ability to calculate the time evolution of these variables. Objects communicate with each other through event-triggered messages that initiate function calls in other objects. Simulations are set up by creating objects (nodes) and messages between them (edges), using scripting languages and declarative data files such as GENESIS .p files.

MOOSE applications

MOOSE provides a repertoire of classes to create and simulate models in Neuroscience and Systems Biology. A key feature of MOOSE is its ability to encapsulate numerical engines, also known as solvers. These are typically highly optimized numerical programs designed for narrowly specialized domains in computational biology. For exam-

ple, the Smoldyn [2] simulator for 3-D single molecule Monte Carlo chemical simulations has been successfully integrated into MOOSE. Other current solvers include chemical kinetic solution engines using the GNU Scientific Library, and a Crank-Nicolson solver for compartmental models of neurons. Several other solvers are under development.

Implementation

MOOSE is written in C++ with extensive use of templates. It is being parallelized using MPI, but the architecture separates the communication layer so that developers of MOOSE classes just need to implement them serially. It inherits the legacy GENESIS parser, making it backward compatible with GENESIS. MOOSE uses SWIG to interface with the Python programming language. It is open source [3].

Results

MOOSE is in Beta 1.0.0. It currently supports chemical kinetic, single neuron, and network simulations including most of the core functionalities of GENESIS. Many GENESIS scripts run on MOOSE, including systems biology and kinetic simulations generated by "kinetikit" [4]. Kinetic simulations in MOOSE typically run 10–50 times faster than GENESIS. Compartmental models run at similar speeds as GENESIS. Network models currently run around twice as fast.

References
1. Wilson MA, Bhalla US, Bower JM: **GENESIS: A System for Simulating Neural Networks.** *Advances in Neural Information Processing Systems* 1989, **1:**485-492.
2. Andrews SS, Bray D: **Stochastic simulation of chemical reactions with spatial resolution and single molecule detail.** *Phys Biol* 2004, **1:**137-151.
3. [http://moose.ncbs.res.in].
4. Bhalla US: **Use of Kinetikit and GENESIS for modeling signaling pathways.** *Methods Enzymol* 2002, **345:**3-23.