Multi-compartmental and multi-scale modeling in MOOSE via Python

Subhasis Ray
National Centre for Biological Sciences
Tata Institute of Fundamental Research
Bangalore, INDIA

BrainScaleS CodeJam 5, Edinburgh 2012
Outline

- Walk through: simple compartmental model
- Multiscale modeling outline
- Architecture of MOOSE
- Extensions
- Preview of upcoming version
- Future directions
the Multiscale Object-Oriented Simulation Environment

- A general purpose simulator framework
- Draws from experience with GENESIS
What is MOOSE?

Depends on what you are and where you look.
A single passive compartment

import moose
soma = moose.Compartment('soma')
soma.Rm = 7.6e6
soma.Cm = 7e-9
soma.Em = -70e-3
soma.inject = 1e-6

- Getting help:
  moose.doc('Compartment')
  moose.doc('HHChannel.channel')
Recording data

```
vm_table = moose.Table('/vm')

vm_table.stepMode = 3

vm_table.connect('inputRequest', soma, 'Vm')
```
Scheduling

```python
moose.context.setClock(0, sim_dt)
moose.context.setClock(1, sim_dt)
moose.context.setClock(2, plot_dt)

moose.context.useClock(0, '/soma', 'init')
moose.context.useClock(1, '/##[TYPE=Compartment]', 'process')
moose.context.useClock(2, '/vm')
```
Running the simulation

moose.context.reset()

moose.context.step(50e-3)

vm_table.dumpFile('soma_vm.txt')
Adding another compartment
soma.Ra = 1e6
axon = moose.Compartment('axon')
...
soma.connect('raxial', axon, 'axial')
Inserting Hodgkin-Huxley-type ion channels
Setting up the channel

\[ G_k = G_{bar} \times m^3 \times h \]

na_chan = moose.HHChannel('/soma/Na')
na_chan.Gbar = 1e-9
na_chan.Xpower = 3
na_chan.Ypower = 1
na_chan.connect('channel', soma, 'channel')
Setting up the gates

\[ \frac{dm}{dt} = alpha \times (1 - m) - beta \times m \]

\[ alpha = \frac{A + B \times Vm}{C + \exp\left(\frac{D + Vm}{F}\right)} \]

Equation for \( beta \) has same form as \( alpha \)

\[ \text{na\_chan\_setupAlpha('xGate', alphA, alphB, alphC, alphD, alphF, betA, betB, betC, betD, betF, divs, vmin, vmax)} \]
Or simply ...

```python
from moose import neuroml
reader = neuroml.NeuroML()
reader.readNeuroMLFromFile('GranuleGenerated.net.xml')
cell = moose.Cell('/Gran_0')
```
Even simpler: use MOOSE GUI
Multiscale modeling

• Ticks with different time-steps run different components of a model at different rates.
Multiscale modeling

- Ticks with different time-steps run different components of a model at different rates.
Multiscale modeling

- Ticks with different time-steps run different components of a model at different rates.
Multiscale modeling

- Ticks with different time-steps run different components of a model at different rates.
A more realistic system

Chemical Compartment

NMDA Channel

Electrical Compartment (spine)

Ca Pool

Ca Channel
Multiscale modeling: networks

• You can connect neuronal compartments via synapse to create a network.
Multiscale modeling: networks

- You can connect neuronal compartments via synapse to create a network.
Multiscale modeling: networks

- You can connect neuronal compartments via synapse to create a network.
Multiscale modeling: networks

- You can connect neuronal compartments via synapse to create a network.
Multiscale modeling: networks

• You can connect neuronal compartments via synapse to create a network.
Multiscale modeling: Olfactory bulb

Aditya Gilra

Mitral Cells [Bhalla & Bower 1993]
10 gloms, 20 cells, 400 ORNs-->mit per mit

Glomerulus
Primary Dendrite
Soma
Lateral Dendrite

Granule Cells [Migliore & Shepherd 2008]
~8,500 aggregated cells
100 mitral-->granule per granule,
1000 granule--mitral per mitral

PG Cells [self],
~400 aggregated cells
500 ORNs-->PG per PG,
25 mitral-->PG per PG
100 PG--mitral per mitral
Multiscale modeling: Cortical column

- Detailed biophysical model of single thalamocortical column (based on Traub et al 2005) of rat barrel cortex:
  - 14 cell types
  - 50-120 compartments
  - 11 ion channel types
  - ~3500 cells
MOOSE Architecture

Figure: Upi Bhalla
MOOSE Architecture: scripting interface

The command line / scripting interface is where modeler interacts with MOOSE.
MOOSE Architecture: Shell

The user interface talks to **Shell** - the single point of access to all functionalities available to the user.

Derived from:
http://moose.sourceforge.net/images/stories/architecture_65.jpg
MOOSE Architecture

Shell creates, deletes and queries other MOOSE objects. 
It also executes some of the built-in functionalities reinit, start.

MOOSE Architecture: Messaging

Objects communicate state variables during simulation.

Destination fields give handle to functions for callback.

```python
na_channel.connect('channel', soma, 'channel')
```

![Diagram showing connections between `na_channel`, `Gk/Ek`, `Vm`, `handleVm()`, and `handleChannel()`]
MOOSE Architecture: scheduling

At start-up, the scheduling system is initialized

Derived from:
http://moose.sourceforge.net/images/stories/architecture_65.jpg
MOOSE Architecture: scheduling

Derived from:
http://moose.sourceforge.net/images/stories/architecture_65.jpg
MOOSE Architecture: scheduling

Derived from:
http://moose.sourceforge.net/images/stories/architecture_65.jpg
MOOSE Architecture: scheduling

Derived from:
http://moose.sourceforge.net/images/stories/architecture_65.jpg
MOOSE Architecture: solvers

Solvers can take over the calculation from simulation entities

Derived from:
http://moose.sourceforge.net/images/stories/architecture_65.jpg
Solvers: Avoid the penalty of object orientation
Solvers: Avoid the penalty of object orientation
Solvers: Avoid the penalty of object orientation

Compartment 1       Compartment 2       Compartment 3       Compartment 4

Solver

Tick0

Hsolve – neuronal (Niraj)
Ksolve – biochemical (Upi)
Gillespie – biochemical (Upi)
Extensions

• NeuroML and SBML support (9ml partially) (Aditya Gilra, Siji George).

• MUSIC API for runtime interaction with other simulators (Niraj Dudani, Johannes Hjorth)

• Smoldyn engine for reaction diffusion systems at cellular scale (Steve Andrews, Upi Bhalla).

• Markov models of ion channels (Vishaka Datta)

• Generally, MOOSE C++ API allows incorporation of other specialized simulation engines as solvers.
New MOOSE: sneak preview

- Complete rewrite applying lessons learned from previous releases.
- Facilities for dynamic class information.
- Focus on utilizing multi-core CPUs and multi-node clusters.
New MOOSE: sneak preview

- Complete switch to Python interface (kinetikit and GENESIS prototype files still supported).
- Computation cleanly separated from user interaction (runs in separate thread). No more blocking during long running simulation.
- Automatic distribution of load among different threads/nodes.
- Most of biophysics and biochemistry classes have been ported. Porting of Hsolve, NeuroML-reader, SBML-reader and the GUI are under development.
- Native support for saving data in HDF5 format.
New MOOSE: sneak preview

• Python interface rewritten using Python/C API
• Much slimmer and more efficient than SWIG-based interface.
• C++ programmer need not worry about Python – it dynamically queries the MOOSE core and generates the class hierarchy using metaprogramming.
Future developments

- Smoldyn support being ported.
- SigNeur – a specialized solver combining signaling models with neuronal models.
- Solvers to utilize GPU for computation in plan.
- Scope for many other solvers at/between various scales.

Join us at:

moose.sourceforge.net
moose.ncbs.res.in
Trying out new moose:

- To check out:
  
  ```bash
  svn checkout \ http://moose.svn.sourceforge.net/svnroot/moose/moose/branches/dh_branch/ \ new_moose
  ```

- To compile (needs `g++`, `make`, `python-dev`, `gsl-dev [optional HDF5-dev]`):
  ```bash
  cd new_moose
  make pymoose [optional USE_HDF5=1]
  ```

- To try squid axon demo (requires `numpy`, `PyQt4` and `matplotlib`):
  ```bash
  export PYTHONPATH={path to new_moose}/python:$PYTHONPATH
cd {path to new_moose}/Demos/squid
python squid_demo.py
  ```
Thank you!

• Upi, Niraj Dudani, Harsharani, Chaitanya, Aditya Gilra and many other contributors to the MOOSE project.

• NCBS/TIFR, DAE/SRC, INCF, SBCNY