morphforge
Biophysical simulation in Python

Mike Hull (s0897465@sms.ed.ac.uk)
University of Edinburgh
University of Bristol

FACETS Code Jam #4
Marseille 22-24 June 2010
Outline

Introduction
- My research
- Motivation
- What is morphforge?

Morphforge
- Morphologies
- Simulations
- Sweeps

From Here
- Wishlist
- Collaborations
About Me & My Work

- Collaboration with Alan Robert’s experimental lab
About Me & My Work

- Collaboration with Alan Robert’s experimental lab
- Locomotive networks in Xenopus *laevis* tadpoles
About Me & My Work

- Collaboration with Alan Robert’s experimental lab
- Locomotive networks in Xenopus *laevis* tadpoles
- Modelling:
  - Small Networks (2000 neurons)
About Me & My Work

- Collaboration with Alan Robert’s experimental lab
- Locomotive networks in *Xenopus laevis* tadpoles
- Modelling:
  - Small Networks (2000 neurons)
  - Hodgekin-Huxley type models of different neuron classes
About Me & My Work

- Collaboration with Alan Robert’s experimental lab
- Locomotive networks in Xenopus *laevis* tadpoles
- Modelling:
  - Small Networks (2000 neurons)
  - Hodgekin-Huxley type models of different neuron classes
  - Morphology of neurons important due to electrical coupling
Previous Workflow (MSc Project)

- Initial channel modelling
  Handwritten .hoc & .mod files (NEURON)

- Effects of changing parameters
  Cheetah generated .hoc & .mod files, scripts for building, re-import data as CSV for plotting

- Network modelling
  2 x YAML + XML files converted into another XML neuron containing network & output specification — cheetah generated .hoc & .mod files, Makefiles for building, re-import data from csv as for plotting, caching of results in a directory.

- Time for a rethink ?!!!
Previous Workflow (MSc Project)

- Effects of changing parameters
  Cheetah generated .hoc & .mod files, scripts for building, re-import data as CSV for plotting
Previous Workflow (MSc Project)

- Initial channel modelling
  - Handwritten .hoc & .mod files (NEURON)
- Effects of changing parameters
  - Cheetah generated .hoc & .mod files, scripts for building, re-import data as CSV for plotting
- Months passed
- Network modelling
  - 2 x YAML + XML files converted into another XML neuron containing network & output specification = cheetah generated .hoc & .mod files, Makefiles for building, re-import data from csv as for plotting, caching of results in a directory.
- Time for a rethink?!?
Previous Workflow (MSc Project)

- Network modelling
  2 × YAML + XML files converted into another XML neuron containing network & output specification → cheetah generated .hoc & .mod files, Makefiles for building, re-import data from csv as for plotting, caching of results in a directory.
Previous Workflow (MSc Project)

- Initial channel modelling
  - Handwritten .hoc & .mod files (NEURON)

- Effects of changing parameters
  - Cheetah generated .hoc & .mod files, scripts for building, re-import data as CSV for plotting

- Network modelling
  - 2 x YAML + XML files converted into another XML neuron containing network & output specification = cheetah generated .hoc & .mod files, Makefiles for building, re-import data from csv as for plotting, caching of results in a directory.

- Time for a rethink ?!?!?
Motivation

Morphology
- FACETS Code Jam '09 - Phillip Rautenberg
- Morphology Reconstructions in Bristol
- DIADEEM Project

Simulation
- Large Parameter Sweeps
What is morphforge?

Python Libraries for:

- Handling neural **morphologies**
What is morphforge?

Python Libraries for:

- Handling neural **morphologies**
- Defining & running **biophysical simulations**
What is morphforge?

Python Libraries for:

- Handling neural morphologies
- Defining & running biophysical simulations
- Analysing & storing simulation results
What is morphforge?

Python Libraries for:
- Handling neural **morphologies**
- Defining & running **biophysical simulations**
- Analysing & storing simulation **results**
- Simplifying **parameter sweeps**
Morphologies
Morphologies

> Represent morphologies as a **tree of cylinders**
> Cylinders can be assigned regions and/or id’s.
Import/Export & Visualisation

- Create morphologies in Python
- Load and save .swc files.
- Load MorphML files
Create morphologies in Python
Load and save `.swc` files.
Load MorphML files

Morphologies can be visualised using:
- 2D projections in `matplotlib`
- 3D visualisation in `MayaVi`
Analysis & Manipulation

- General purpose data structure
- Minimal classes + loose coupling $\implies$ Visitor pattern
- (Example of straightening tadpole)
Examples
Simulations
Simulation Overview

- Simulator Agnostic Description of:
  - Neurons
  - Active membrane channels (HH-style)
  - Passive membrane properties
  - Voltage & current clamps
  - Recording electrodes (voltages, currents & membrane properties)
Simulation Overview

- Simulator Agnostic Description of:
  - Neurons
  - Active membrane channels (HH-style)
  - Passive membrane properties
  - Voltage & current clamps
  - Recording electrodes (voltages, currents & membrane properties)

- Handles units
Simulation Overview

- Simulator Agnostic Description of:
  - Neurons
  - Active membrane channels (HH-style)
  - Passive membrane properties
  - Voltage & current clamps
  - Recording electrodes (voltages, currents & membrane properties)
- Handles units
- Simplification of plotting
Simulation Overview

- Simulator Agnostic Description of:
  - Neurons
  - Active membrane channels (HH-style)
  - Passive membrane properties
  - Voltage & current clamps
  - Recording electrodes (voltages, currents & membrane properties)
- Handles units
- Simplification of plotting
- Basic trace analysis
Simulation Overview

- Simulator Agnostic Description of:
  - Neurons
  - Active membrane channels (HH-style)
  - Passive membrane properties
  - Voltage & current clamps
  - Recording electrodes (voltages, currents & membrane properties)
- Handles units
- Simplification of plotting
- Basic trace analysis
- Caching simulation results
Neuron Specific

- Uses python-neuron interface
- Use existing .mod files directly
- Behind the scenes:
  - Generate .hoc and .mod files
  - Compiles .mod files
  - Registers .mod files into neuron-instance
Simulation Examples
Sweeps & Bundles
Bundles

A wrapper around Simulation objects, in order to:

- Attach pre/post-simulation **functors**
- Encapsulate **serialisation**
A 'Bundle' wraps a 'Simulation'

**Bundle**

```python
def load(...):
def save(...):
def addPreFunctor(...):
def addPostSimFunctor(...):
def execute(...):
```

**Simulation**

```python
def addNeuron(...):
def addCurrentClamp(...):
def addVoltageClamp(...):
def recordVoltage(...):
def recordCurrent(...):
def Simulate(...):
```

**Bundle::execute()**

- Execute Pre-Sim Actions
- Run Simulation::Simulate()
- Execute Post-Sim Actions
A simple mechanism for distributing simulations over XML-RPC has been written; comprising a **BundleServer** and **BundleClient**;

- the user needs to write a function returning a list of simulation-bundles to run, for the BundleServer.
A simple mechanism for distributing simulations over XML-RPC has been written; comprising a BundleServer and BundleClient; the user needs to write a function returning a list of simulation-bundles to run, for the BundleServer.

- for example, this could have a post-sim functor that analyses the output voltage traces and writes a row to a DB somewhere.
A simple mechanism for distributing simulations over XML-RPC has been written; comprising a **BundleServer** and **BundleClient**;

- the user needs to write a function returning a list of simulation-bundles to run, for the BundleServer.
  - for example, this could have a post-sim functor that analyses the output voltage traces and writes a row to a DB somewhere.
- The BundleServer is started on a single machine and acts as a daemon, keeping a track of which bundles have been handed out to which clients.
A simple mechanism for distributing simulations over XML-RPC has been written; comprising a **BundleServer** and **BundleClient**;

- the user needs to write a function returning a list of simulation-bundles to run, for the BundleServer.
  - for example, this could have a post-sim functor that analyses the output voltage traces and writes a row to a DB somewhere.
- The BundleServer is started on a single machine and acts as a daemon, keeping a track of which bundles have been handed out to which clients.
- The BundleClient can be started on many clients. Each client contacts the server, requests n Bundles, runs them, notifies the server about whether bundles ran successfully or not, then requests more bundles....
Parameter Sweeps Results Examples

- Modelling the effects of conductances on firing behaviour
- Na, Ca, Kf, Ks, Lk channels + injected current
- 110,000 simulations run in a night over 30 computers
Parameter Sweeps Results Examples

Kf Conductance Density Multiplier

Ks Conductance Density Multiplier

- Single Spiking Neurons
- Multiple Spiking Neurons
- Overly Excitable Neurons
- Non-Repolarising Neurons
- Ambiguous Neurons

Legend:
- Red: Single Spiking Neurons
- Blue: Multiple Spiking Neurons
- Green: Overly Excitable Neurons
- Purple: Non-Repolarising Neurons
- Orange: Ambiguous Neurons
From Here
WishList

High Priority
- Interface for neural connectivity
- Documentation & testing

Low Priority
- Summary pdfs/tex output
- Loading *ML formats
Collaboration

- Code is in a mercurial repository - email s0897465@sms.ed.ac.uk
- To be made public (advice on open licenses)
- Is this useful to other people?
- Integration with other open-source tools
- Keen to find collaboration.....
Acknowledgements

- Juan Reyero (Magnitude Units Package)
- Enthought (MayaVi Package)
- Everyone working on scientific python libraries!
- NEURON & Python Interface
- Organisers of Code Jams
- Supervisors: Alan Roberts & David Willshaw

Thankyou!