CARDIAC ELECTROPHYSIOLOGY WEB LAB

A technical tour

Jonathan Cooper
UCL Research Software Development Group

Previously: Computational Biology Group, CS Dept, Oxford
Different literature structures for $I_{Kr}$ (cardiac ion current)

Model A
$C_3 \leftrightarrow C_2 \leftrightarrow C_1 \rightarrow O$

Model B
$C_3 \leftrightarrow C_2 \leftrightarrow C_1 \rightarrow O$

Model C
$C_2 \rightarrow C_1 \rightarrow O$

Model D
$IC_1 \leftrightarrow I \leftrightarrow C_1 \rightarrow O$

Model E
$IC_2 \leftrightarrow IC_1 \leftrightarrow I \leftrightarrow C_2 \leftrightarrow C_1 \rightarrow O$

Model F
$IC_s \leftrightarrow I_s \leftrightarrow I_f \leftrightarrow IC_f \leftrightarrow IC_s \leftrightarrow C_f \rightarrow O_f \leftrightarrow O_s$

Model G
$(S_2 \leftrightarrow S_1 \leftrightarrow S_0)^4 \leftrightarrow C_2 \leftrightarrow C_1 \rightarrow O_1 \rightarrow O_2$

C = closed
O = open
I = inactivated
Different models, different predictions

(some of this variation is to be expected… but which model should the FDA use?)
A vision of the future…

• Knowledge about mechanisms is captured in quantitative models

• Best experiments to do are therefore the ones that best [select and] parameterise the model

• Provide these to experimentalists

• Automate model development

• Deploy in the Virtual Physiological Human!
Motivation

Current simulations

- Model repository
- Model
- Model
- Model
- Single hard-coded protocol

Simulator
What does the Web Lab enable?

https://chaste.cs.ox.ac.uk/WebLab
Key features summary

• Consistent application of a protocol to any model
  • Interface described at the level of biophysical concepts
    (ontology annotation)
  • Units conversions are all handled automatically

• Specify model inputs and outputs
  • Simulator works out which equations it needs for that simulation

• Replace components
  • For example encode your own stimulus protocol, or apply voltage clamps

• Includes all the post-processing and plotting instructions
  (array-based functional language)

• Able to do complex parameter sweeps, analysis, etc.
Demo

https://chaste.cs.ox.ac.uk/WebLab
### View of experiments run

<table>
<thead>
<tr>
<th>Model/Methodology</th>
<th>extracellular potassium variation</th>
<th>ICaL block</th>
<th>ICaT IV Curve</th>
<th>IK block</th>
<th>IKr IV Curve</th>
<th>IKr Block</th>
<th>INa block</th>
<th>INa IV Curve</th>
<th>RYR Block</th>
<th>S1-S2 block</th>
<th>S1-S2 block 0.5Hz pacing</th>
<th>S1-S2 block 1Hz pacing</th>
<th>S1-S2 block 2Hz pacing</th>
<th>S1-S2 block 3Hz pacing</th>
<th>S1-S2 block 4Hz pacing</th>
<th>Steady state block</th>
<th>Steady state block 0Hz pacing</th>
<th>Steady state block 1Hz pacing</th>
<th>Steady state block 2Hz pacing</th>
<th>Steady state block 3Hz pacing</th>
<th>Steady state block 4Hz pacing</th>
<th>Steady state block Restitution</th>
</tr>
</thead>
<tbody>
<tr>
<td>Aslanidi atrial model 2009</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Aslanidi Purkinje model 2009</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Beeler-Reuter 1977</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Benson 2008</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Bernus 2002</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Bondarenko 2004 Apical</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Decker 2009</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>DiFrancesco Noble 1985</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Earm Noble 1990</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Grandi 2010</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Li 2010</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Luo Rudy 1991</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Mahajan 2008</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Matsuoka 2003</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Noble 1998</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>O'Hara 2011</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Priebe 1998</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Shannon 2004</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>ten Tusscher 2006 Epi</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>tenTusscher 2004 Epi</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Winslow 99</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
</tbody>
</table>

**Key:**
- **not run**
- **queued**
- **running**
- **ran successfully**
- **partially ran**
- **failed to complete**
- **inappropriate**
Results of an experiment
Comparing experiments – drug block
Some of the technologies involved

- CellML & Combine Archive
- Python
  - Pyparsing, Amara, RDFLib
  - Numpy, pytables, numexpr
- Cython & CVODE
  - Original backend in C++
- Tomcat & JSP
- Celery & RabbitMQ
- MySQL
- Javascript & jQuery
  - rdfQuery
- Flot, Highcharts
Cython: ODE solves as fast as C

- Electrophysiology cell models are moderately complex ordinary differential equations
  - Right-hand side coded in Python => far too slow!
- “The Cython language is a superset of the Python language that additionally supports calling C functions and declaring C types on variables and class attributes. This allows the compiler to generate very efficient C code from Cython code.”
- CVODE is a best-of-breed adaptive ODE solver written in C
Wrapping a C library with Cython

.pxd file:

```python
from nvector/nvector_serial.h:

cdef N_Vector N_VMake_Serial(long int vec_length, realtype *v_data)

cdef struct _N_VectorContent_Serial:
    long int length
    realtype *data

ctypedef _N_VectorContent_Serial *N_VectorContent_Serial

from cvode/cvode.h:

int CV_ADAMS

cdef extern int (*CVRhsFn)(realtype t, N_Vector y, N_Vector ydot, void *user_data)

void *CVodeCreate(int lmm, int iter)

int CVode(void *cvode_mem, realtype tout, N_Vector yout, realtype *tret, int itask)
```
A Cython ODE model: .pxd file

cimport numpy as np

cdef class CvodeSolver:
    cdef void* cvode_mem  # CVODE solver 'object'
    cdef N_Vector _state  # The state vector of the model
    cdef public np.ndarray state  # Numpy view of the state
    cdef public object model  # The model being simulated

    cpdef Simulate(self, realtype endPoint)
A Cython ODE model: .pyx file

cimport numpy as np
import numpy as np
cimport fc.sundials.sundials as _lib

cdef extern from "Python.h":
    object PyBuffer_FromReadWriteMemory(void *ptr, Py_ssize_t size)

cdef object NumpyView(N_Vector v):
    """Create a Numpy array giving a view on the CVODE vector passed in."""
    cdef _lib.N_VectorContent_Serial v_content =
        <_lib.N_VectorContent_Serial>(v.content)
    ret = np.empty(v_content.length, dtype=np_dtype)
    ret.data = PyBuffer_FromReadWriteMemory(v_content.data, ret.nbytes)
    return ret

self._state = _lib.N_VMake_Serial(self._state_size,
    <realtype*>(<np.ndarray>self.state).data)
flag = _lib.CVodeInit(self.cvode_mem, _RhsWrapper, 0.0, self._state)
Numexpr: Post-proc faster than C++

“Numexpr is a fast numerical expression evaluator for NumPy. With it, expressions that operate on arrays (like "3*a+4*b") are accelerated and use less memory than doing the same calculation in Python.”

- No intermediates
- Good cache utilization
- Multi-threaded
- Can also use the Intel Vector Math Library

- So very quick at mapping calculations over one or more n-d arrays
## Timing results

<table>
<thead>
<tr>
<th>Test case</th>
<th>ICaL Protocol</th>
<th>S1S2 Protocol</th>
</tr>
</thead>
<tbody>
<tr>
<td>Original C++</td>
<td>197 (95)</td>
<td>201 (35)</td>
</tr>
<tr>
<td>Original Python</td>
<td>792</td>
<td>279</td>
</tr>
<tr>
<td>First Cython attempt</td>
<td>614 (583)</td>
<td>117 (54)</td>
</tr>
<tr>
<td>Optimised Cython</td>
<td>152 (125)</td>
<td>118 (27)</td>
</tr>
<tr>
<td>Final C++</td>
<td>266 (162)</td>
<td>204 (36)</td>
</tr>
</tbody>
</table>

- All times are in seconds
- Time just for simulation portion of protocol in ()
Web app: Tomcat

- Open source stack for Java-based web applications
  - Java Servlet, JavaServer Pages (JSP), etc.
- Would have been more logical to use a Python framework given the rest of the project, but this was what the intern that first developed the web interface knew!
  - So was able to get something working quickly
- Talks to:
  - MySQL database for metadata, user info, etc.
  - File system for model, protocol & result files
  - Celery via CGI
  - Javascript with AJAX + JSON
Task processing with Celery

• “Celery is an asynchronous task queue/job queue based on distributed message passing. It is focused on real-time operation, but supports scheduling as well.”

• Uses RabbitMQ broker (written in Erlang) for messaging, but tasks written in Python

• Aimed at handling large numbers of quick tasks; Web Lab uses it for distributing long-running experiments across workers
  • And extracting protocol interface info

• Nice extras, like live monitoring on the web with Flower
Celery usage in the Web Lab

- Messages should be small
  - Pass URLs for models & protocols; experiment task downloads & unpacks these on the worker
  - Also passed a callback URL for POSTing results files
    - Callbacks are auto-retried in case front-end is busy
- Our tasks are long
  - Workers don’t reserve extra tasks
  - Allow tasks to be revoked mid-run (by user action)
  - Track ‘pending’ and ‘running’ states
  - Return partial results if exceed time limit
- Optionally different workers for different users
  - At present needs manual setup
- Fairest scheduling for users still an open question
Celery code snippets

In `__init__.py`:

```python
def ScheduleExperiment(callbackUrl, signature, modelUrl, protoUrl, 
    user='', isAdmin=False):
    """Schedule a new experiment for execution."""
    from .tasks import CheckExperiment
    # Submit the job
    result = CheckExperiment.apply_async(
        (callbackUrl, signature, modelUrl, protoUrl),
        queue=GetQueue(user, isAdmin))
    # Tell web interface that the call was successful
    print signature, "succ", result.task_id
```

In `tasks.py`:

```python
app = celery.Celery('fcws.tasks')
app.config_from_object(celeryconfig)

@app.task(name="fcws.tasks.CheckExperiment")
def CheckExperiment(callbackUrl, signature, modelUrl, protocolUrl):
    ...
```
Visualization: Flot

- A pure Javascript plotting library integrated with jQuery
- Focus on simplicity & interactivity
  - But still many options!
- Chosen because a colleague had used it previously
- Data series passed as JS arrays
  - Parsed from CSV files created by experiment runs
- For many features you have to use plugins, or even add in yourself (copied from examples)
  - Graph legend with ability to turn traces on & off
  - Zoom & pan with ‘reset’ button
  - Hover over point for details tooltip
Visualization: Highcharts

- Commercial product but free for non-commercial use, and open source
- Wanted to find something that required less customisation
  - Has built-in hover, legend & zoom, for instance
- May be harder to customise if you want to though!
- API is similar to Flot, but various minor differences in naming & options structure
Data & Metadata in Javascript

- Currently serve CSV to the front-end
- Would like to move to HDF5, but no Javascript library?
- We use down-sampling for plots to speed up rendering

- Metadata encoded in RDF/XML within models
- Created a Javascript drag & drop model annotator
- Javascript RDF/XML support is patchy!
The future of Web Lab

With reusable virtual experiments

Model repository

Domain-specific ontology annotations

Experimental data

Model

Protocol

Protocol

Protocol repository

Library of protocol components

Simulator
Final thoughts

• It’s fun to have a complex project on which you can try out different technologies 😊

• Balance between choosing a ‘best’ solution and going with something you can get working in a reasonable time
  • Particularly for web apps, which framework is ‘best’ changes rapidly!

• Sometimes it’s best to throw away what you have and start afresh – learn from your prototype’s mistakes

• Comparing different implementations of numerical code is (very) hard
Acknowledgments

Gary Mirams, University of Nottingham

- Additional development work by:
  - Martin Scharm – 2020 intern – initial Web Lab
  - Aidan Daly – DPhil student & 2020 intern – fitting & electrochemistry
  - Erich Kerekes – summer student

- Ideas and inspiration:
  - Dagmar Waltemath
  - Jon Olav Vik
  - Steven Niederer
  - Alan Garny
  - David Gavaghan
  - Denis & Penny Noble