An HPC Implementation of the Finite Element Method

John Rugis
New Zealand eScience Infrastructure
Interdisciplinary research group:

David Yule  
Physiology

James Sneyd,  
Shawn Means, Di Zhu  
Mathematics

John Rugis  
Computer Science

Project funding:
The science goal:

Physically accurate modeling, simulation and visualisation of biological cell function.

*Scalable!*

- High Performance Computing
- *An Object Oriented Approach to Biological Cell Modeling*
Salivary glands
A Finite Element Method workflow overview

• Modeling equations *(How does the system work?)*
• Mesh generation *(What does the system look like?)*
• Discretisation *(How can this be implemented on a computer?)*
• Run simulation *(What does the system do?)*
• Evaluate results *(What exactly happened?)*
The Modeling Equations

Start with a conceptual model…
The Modeling Equations

**Calcium and IP$_3$ dynamics**

Partial differential equations model the cell calcium dynamics.

Reaction-Diffusion:

\[
\frac{\partial c}{\partial t} = D_c \nabla^2 c + (J_{\text{IPR}} + J_{\text{leak}})(c_e - c) - J_{\text{serca}} \\
\frac{\partial p}{\partial t} = D_p \nabla^2 p + V_{\text{PLC}}(x) - V_{\text{deg}} \left( \frac{c^2}{K_{3K}^2 + c^2} \right) p \\
\frac{\partial h}{\partial t} = \frac{h_\infty - h}{\tau}
\]

\[
J_{\text{serca}} = V_s \frac{c^2}{K_s^2 + c^2} \\
J_{\text{IPR}} = k_{\text{IPR}}(x)P_O \\
P_O = \phi_c \phi_p h \\
\phi_c = \frac{c^3}{K_a^3 + c^3} \\
\phi_p = \frac{p^4}{K_p^4 + p^4} \\
h_\infty = \frac{K_i^2}{K_i^2 + c^2} \\
c_e = (c_t - c)/\gamma
\]
Mesh Generation - Data Acquisition

Confocal microscopy

2D image stacks
32 slices in Z direction.

Cells & Lumen
red: Na-KATPase
green: Cl Channel

Real dimensions used:
70.7µm², 2.2µm spacing

TIFF files: 1024x1024
Cells & Lumen

The cells are grouped in tight clusters.

The lumen has a tree-like branching structure.
Mesh Generation - 3D Physical Model

Cells & Lumen

Each cell is held by a lumen “claw”.

The lumen has a central trunk.
Mesh Generation - for FEM

Cells

Solid volumetric meshing with tetrahedrons.

Tetrahedrons have vertices and edges.

*Element* = tetrahedron

*Node* = vertex
Discretisation

Facilitates (almost direct) translation to computer code.

- Build “mass” and “load” matrices.
- Construct (sparse) system matrix: ~ 400,000,000 elements!

*NOTE: Most of the simulation run-time is spent solving the sparse system matrix.*
Computer Code

Tool selection

Sparse matrix solver: PETSc
Portable, Extensible Toolkit for Scientific Computation
http://www.mcs.anl.gov/petsc

Vector and matrix library: Eigen
Eigen is a C++ template library for linear algebra
http://eigen.tuxfamily.org
Computer Code

Object oriented!

Language: C++

Once the behaviour of a cell “object” is defined, we can instantiate clusters of cells that interact with each other and their environment.
class cGeneric3dModel {
public:
  cGeneric3dModel(cCellMesh *mesh, cPetscSolver *solver);
  virtual ~cGeneric3dModel();
  void run();
  void save_results();

  MatrixXXC Amat, mass, u; // A, mass and solution matrices

private:
  void get_parameters();
  void init_u();
  MatrixXIC make_load(long i);
  ArrayRefMass make_ref_mass();
  Array1VC getbodyreactions(tCalcs c, tCalcs ip, tCalcs h, tCalcs ipr_f, tCalcs plc_f);
  tCalcs getboundaryflux(tCalcs c);
  void make_matrices();
  void load_node_data(std::string file_name, int dindex);
  void save_matrix(std::string file_name, MatrixXXC mat);

  cCellMesh *mesh;
  cPetscSolver *solver;
  tCalcs p[PCOUNT]; // the model parameters array
  long numt; // number of time steps
  Eigen::Array<tCalcs, Eigen::Dynamic, MODELNCOUNT> node_data;
  Eigen::Array<tCalcs, Eigen::Dynamic, MODELECOUNT> element_data;
};

Computer Code

Object definition
Computer Code

HPC – *How can we scale up?*

- One cell object per compute node
- Scale “out” (i.e. use more compute nodes)
  - MPI for cell interactions (light weight!)
- Scale “in” (i.e. use more cores per compute node)
  - Threads for utility functions
  - Accelerators for the main solver (i.e. GPU’s, Intel Phi)
Simulation

HPC - Auckland Pan Cluster

• Job submission scripting (shell scripting and python)
• Multiple parameter sweeps
• File and directory structures facilitate reproducibility!
Simulation

Job submission script
(extract)
Evaluate Results

Cell
Nodal view

*Precomputed values.*

Imprint of the lumenal “claw”.

distance to nearest lumen (um)

0.0  1.5  3.0  4.5  6.0
Evaluate Results

Cell
Nodal view

Static distributions.

IPR in red.

PLC in blue.
Calcium waves
Wave-fronts propagate from the apical end to the basal end the cell.
What’s next?

• Run simulations with coupled cells (i.e. gap junctions)
• Include fluid flow in lumen (computational fluid dynamics)
• Higher resolution digitisations
• Stay tuned, more to come…
<table>
<thead>
<tr>
<th>Time</th>
<th>Location</th>
<th>Title</th>
<th>Speakers</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tuesday 11:15</td>
<td>(Queenstown)</td>
<td>NeSI Update</td>
<td>Nick Jones</td>
</tr>
<tr>
<td>Tuesday 11:45</td>
<td>(Remarkables)</td>
<td>Growing NZ’s Researcher’s Computing Capability</td>
<td>Georgina Rae &amp; John Rugis</td>
</tr>
<tr>
<td>Tuesday 14:30</td>
<td>(Queenstown)</td>
<td>An HPC Implementation of the Finite Element Method</td>
<td>John Rugis</td>
</tr>
<tr>
<td>Tuesday 16:00</td>
<td>(Queenstown)</td>
<td>The NeSI National Platform Framework</td>
<td>Michael Uddstrom</td>
</tr>
<tr>
<td>Tuesday 16:30</td>
<td>(Queenstown)</td>
<td>Early Experiences with Cloud Bursting</td>
<td>Jordi Blasco</td>
</tr>
<tr>
<td>Tuesday 16:30</td>
<td>(Remarkables)</td>
<td>NeSI – NZGL Alliance</td>
<td>Dan Sun &amp; Nic Mair</td>
</tr>
<tr>
<td>Tuesday 17:00</td>
<td>(Queenstown)</td>
<td>Acceleration Made Easy</td>
<td>Wolfgang Hayek</td>
</tr>
</tbody>
</table>