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:

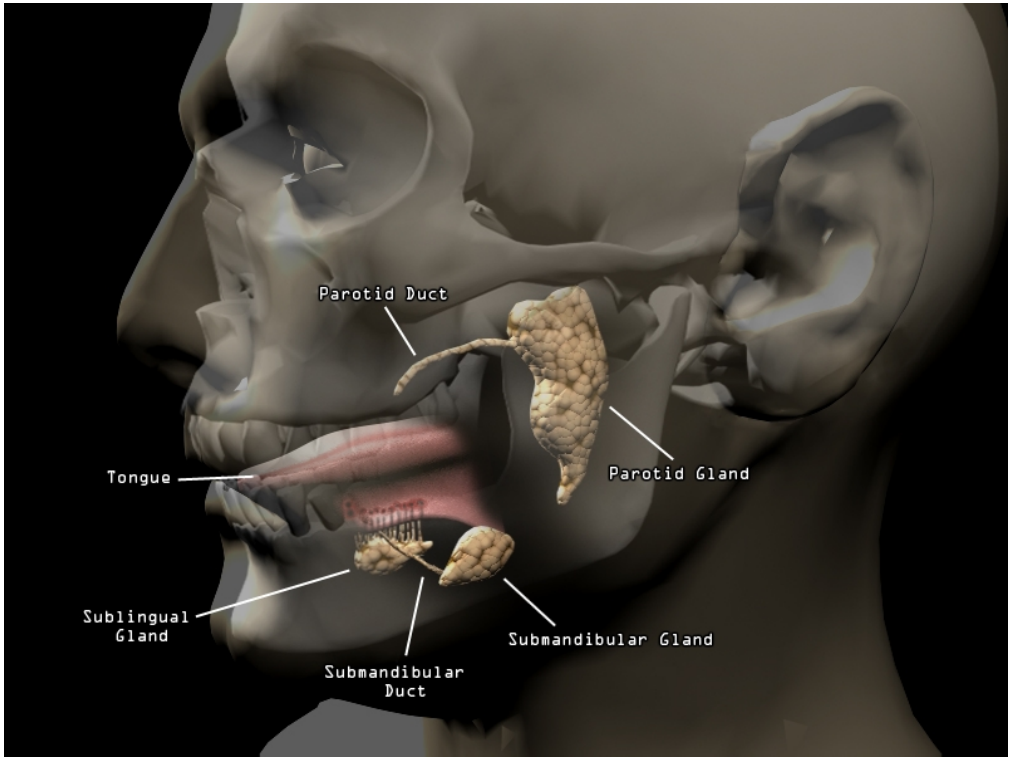
The science goal:

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

Scalable!

- High Performance Computing
- *An Object Oriented Approach to Biological Cell Modeling*

Salivary glands

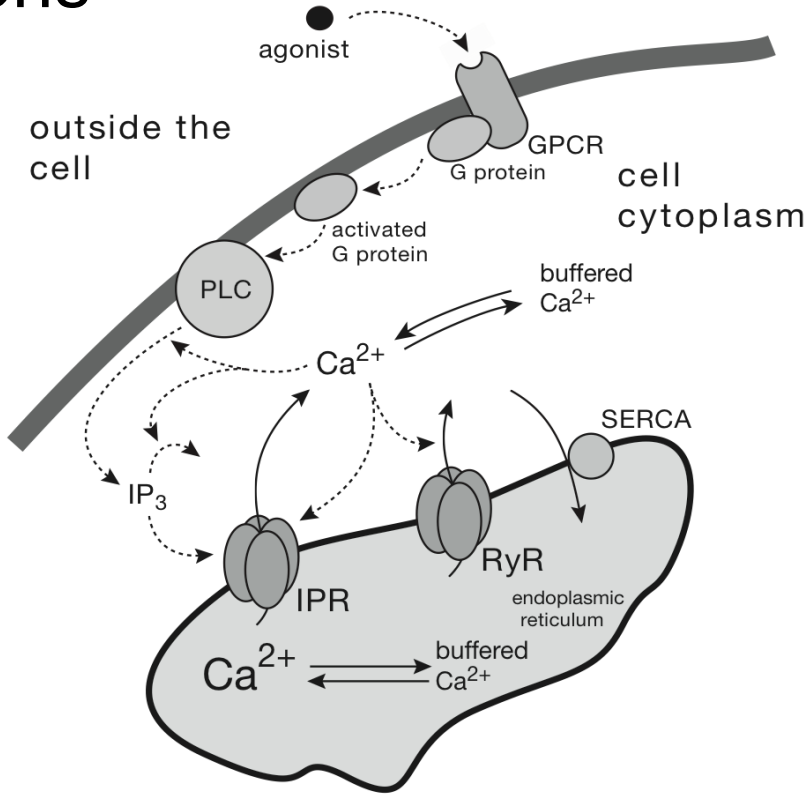


A Finite Element Method workflow overview

- 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:

$$\frac{\partial c}{\partial t} = D_{\mathrm{c}}\nabla^2 c + (J_{\mathrm{IPR}} + J_{\mathrm{leak}})(c_{\mathrm{e}} - c) - J_{\mathrm{serca}}$$

$$\frac{\partial p}{\partial t} = D_{\mathrm{p}}\nabla^2 p + V_{\mathrm{PLC}}(\vec{x}) - V_{\mathrm{deg}}\left(\frac{c^2}{K_{\mathrm{3K}}^2 + c^2}\right)p$$

$$\frac{\partial h}{\partial t} = \frac{h_\infty - h}{\tau}$$

$$J_{\mathrm{serca}} = V_{\mathrm{s}}\frac{c^2}{K_{\mathrm{s}}^2 + c^2}$$

$$J_{\mathrm{IPR}} = k_{\mathrm{IPR}}(\vec{x})P_O$$

$$P_O = \phi_c\phi_p h$$

$$\phi_c = \frac{c^3}{K_{\mathrm{a}}^3 + c^3}$$

$$\phi_p = \frac{p^4}{K_{\mathrm{p}}^4 + p^4}$$

$$h_\infty = \frac{K_{\mathrm{i}}^2}{K_{\mathrm{i}}^2 + c^2}$$

$$c_{\mathrm{e}} = (c_{\mathrm{t}} - c)/\gamma$$

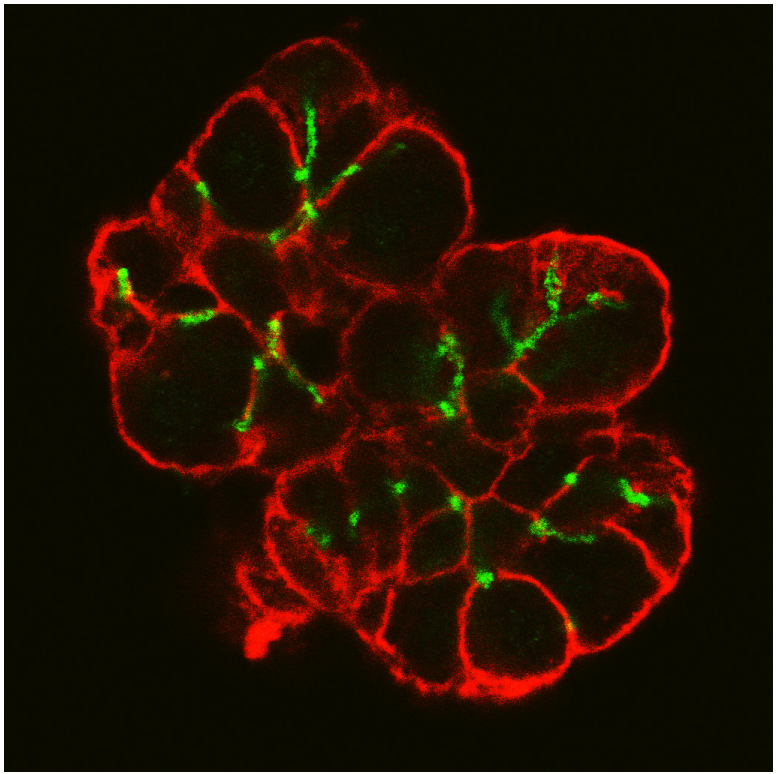
Mesh Generation - Data Acquisition
Confocal microscopy

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

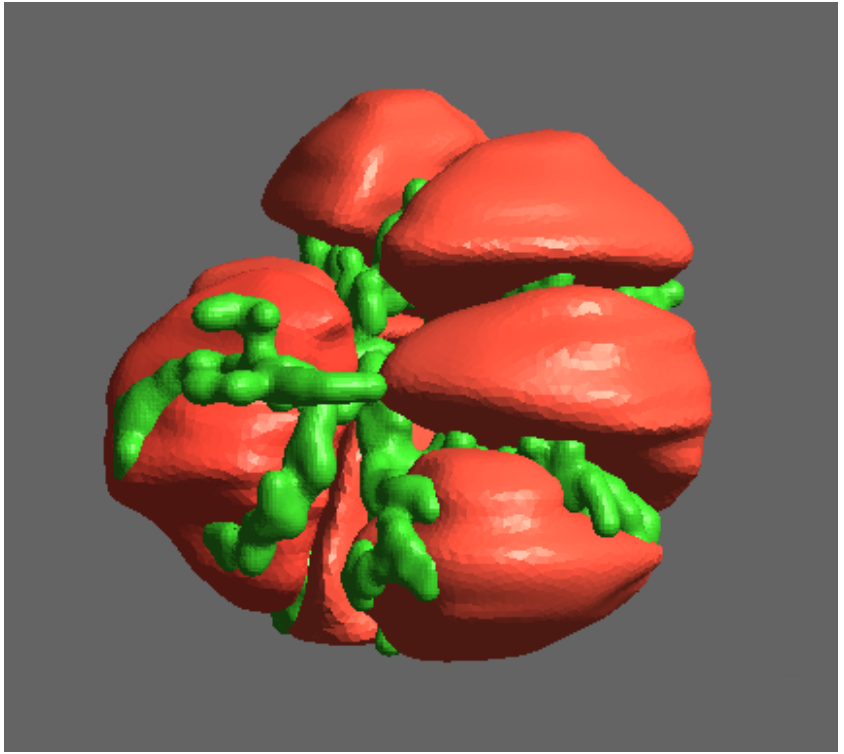


Mesh Generation - 3D Physical Model

Cells & Lumen

The cells are grouped in tight clusters.

The lumen has a tree-like branching structure.

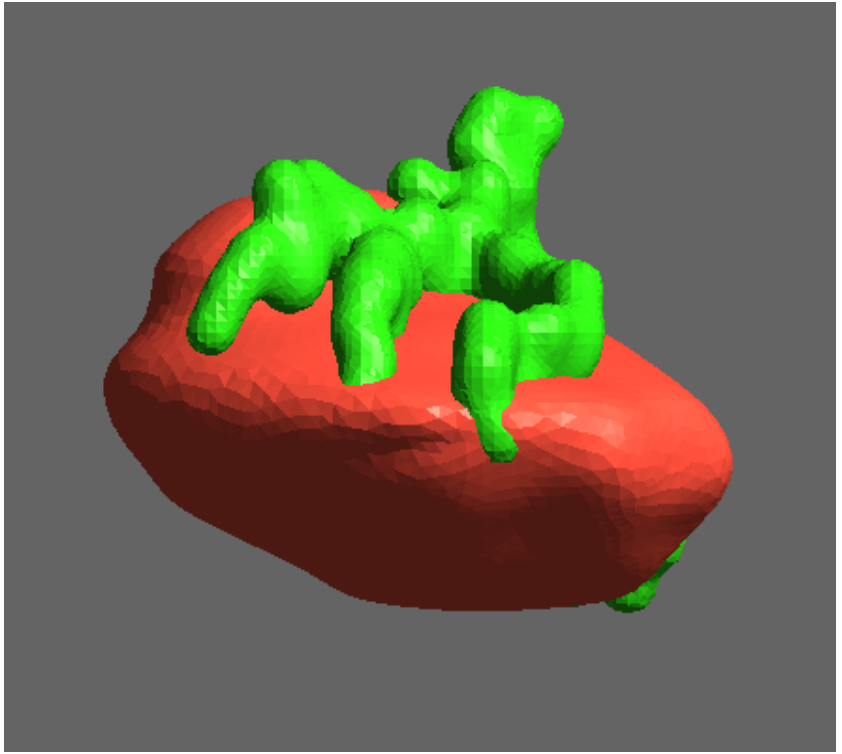


Mesh Generation - 3D Physical Model

Cells & Lumen

Each cell is held by a lumen “claw”.

The lumen has a central trunk.



Mesh Generation - for FEM

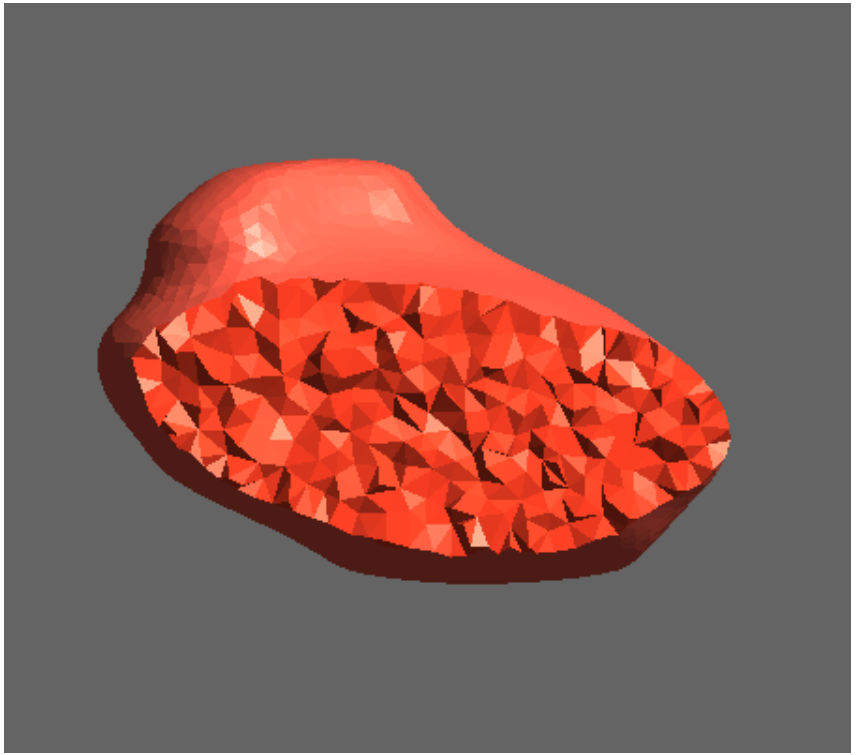
Cells

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.

Computer Code

Object definition

```
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();
    MatrixX1C 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

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)

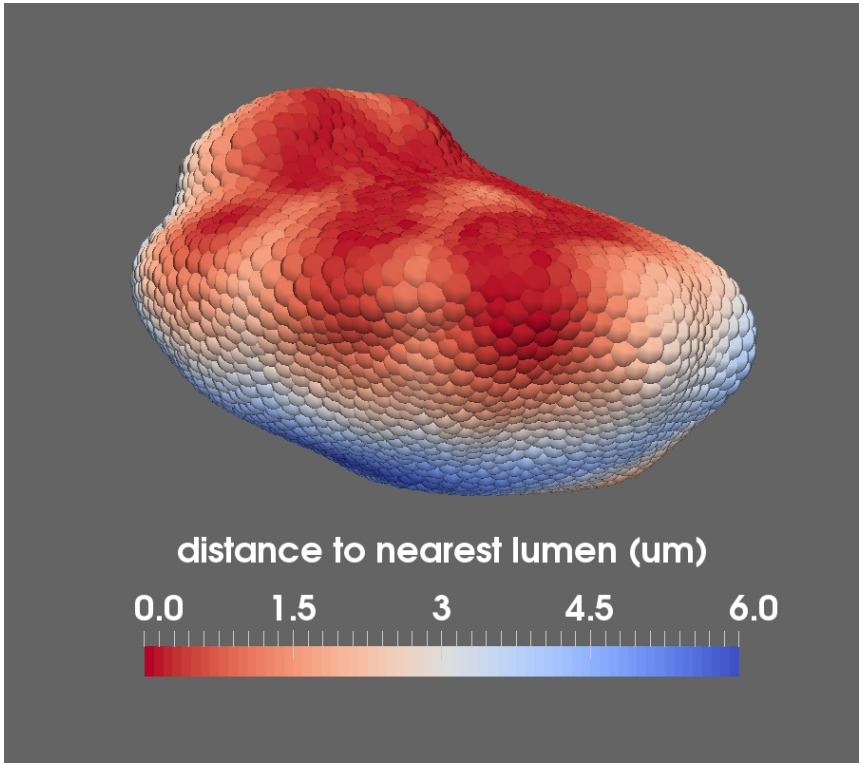
```
# create the top level results directory
csdir = os.getcwd()
path = "results/CS" + time.strftime("%y%m%d%H%M%S")
os.mkdir(path)
os.chdir(path)

# iterate through the parameters
for v1 in valsA:
  for v2 in valsB:
    pdir = ""
    for pv in parms:
      pdir += pv[0] + '-' + str(pv[1]) + '_'
    pdir += parmA + '-' + str(v1) + '_' + parmB + '-' + str(v2)
    pdir = pdir.replace('.', 'p')
    os.mkdir(pdir)
    os.chdir(pdir)

    # copy data files and execute the simulation
    os.system("cp " + csdir + "/parameters/" + model + ".dat cs.dat")
    if(os.path.isfile("ipr_REF.bin") and os.path.isfile("plc_REF.bin")):
      os.system("cp " + csdir + "/parameters/ipr_REF.bin .")
      os.system("cp " + csdir + "/parameters/plc_REF.bin .")

    for pv in parms: # set the fixed values
      replace_pvalue(pv[0], pv[1])
    replace_pvalue(parmA, v1) # set the swept values
    replace_pvalue(parmB, v2)
    os.system("cp " + csdir + "/meshes/" + mesh + ".msh cs.msh")
    os.system("sbatch " + csdir + "/run_sim.sl " + model + " " + mesh + " " + csdir)
    os.chdir("..")
```

Evaluate Results

Cell
Nodal view

Precomputed values.

Imprint of the lumenal “claw”.



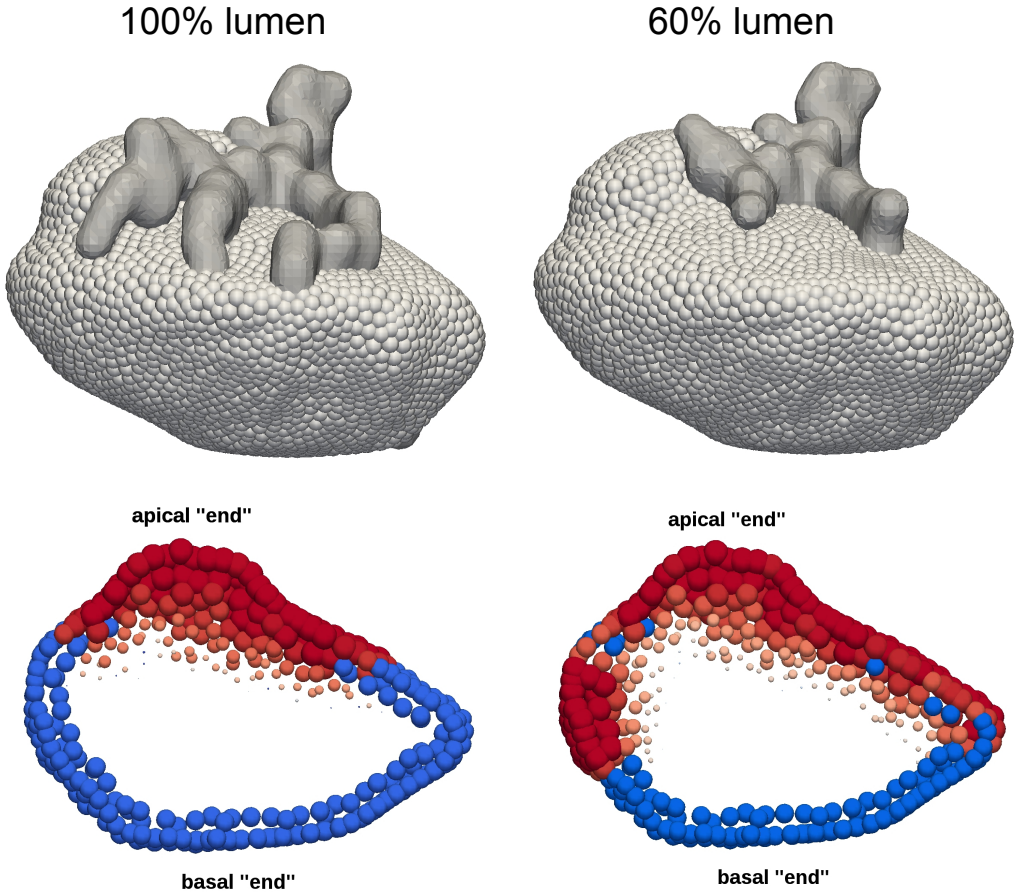
Evaluate Results

Cell
Nodal view

Static distributions.

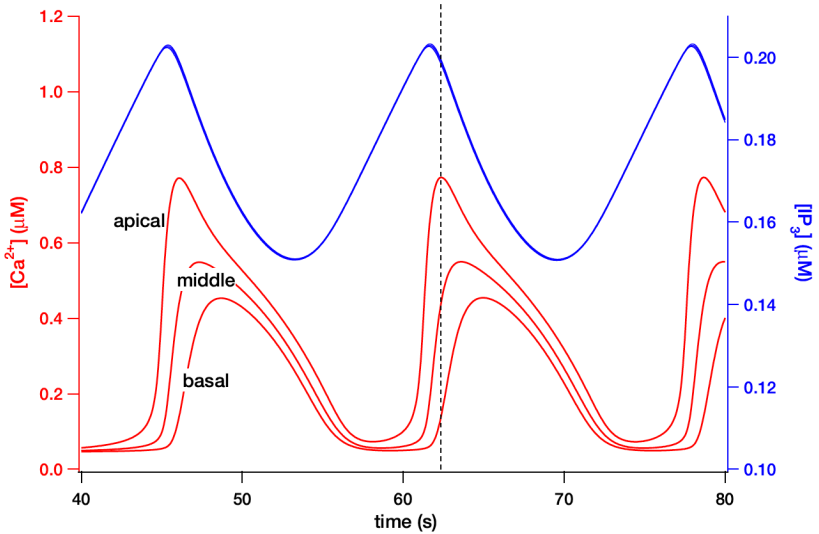
IPR in red.

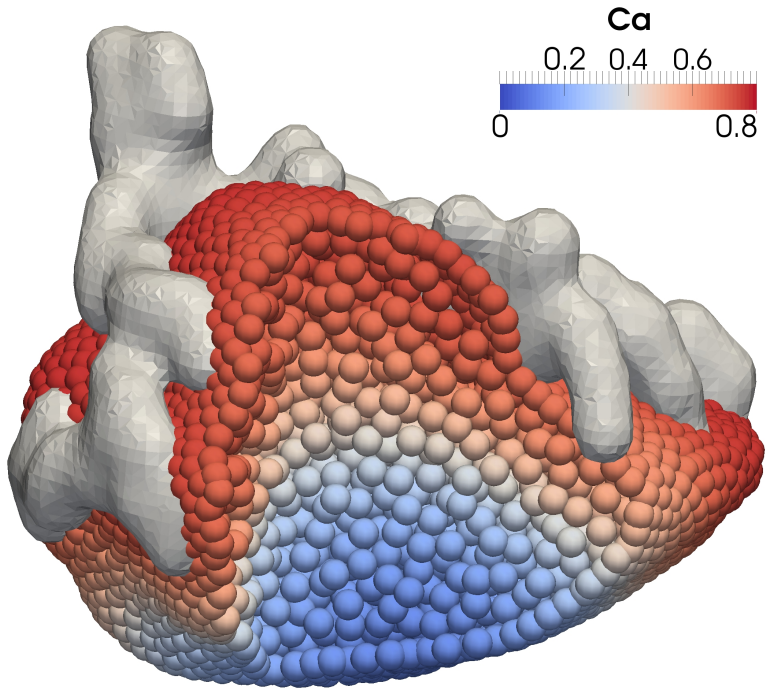
PLC in blue.



FEM Simulation Results

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…

Questions & Answers?

Tuesday 11:15 (Queenstown)	NeSI Update	Nick Jones
Tuesday 11:45 (Remarkables)	Growing NZ's Researcher's Computing Capability	Georgina Rae & John Rugis
Tuesday 14:30 (Queenstown)	An HPC Implementation of the Finite Element Method	John Rugis
Tuesday 16:00 (Queenstown)	The NeSI National Platform Framework	Michael Uddstrom
Tuesday 16:30 (Queenstown)	Early Experiences with Cloud Bursting	Jordi Blasco
Tuesday 16:30 (Remarkables)	NeSI – NZGL Alliance	Dan Sun & Nic Mair
Tuesday 17:00 (Queenstown)	Acceleration Made Easy	Wolfgang Hayek