# Use of CellML in Chaste

## *Chaste: a general purpose simulation package aimed at multi-scale, computationally demanding models*

Jonathan Cooper

Computing Laboratory, University of Oxford

# Outline

- What is Chaste?
- How does Chaste use CellML?
- Current status and next steps

# Chaste history

- Commenced as a 4-week taught course in Software Engineering in May 2005
- May 2005-September 2007 part-time activity, 1 day/week involving a group of around 6-10 PhD students and post-docs
- September 2007-date EPSRC-funding for two full time post-docs to join the team
- Focus remains primarily on cardiac electrophysiology, soft tissue modelling (including cardiac electro-mechanics) and cancer modelling
- Development methodology a key feature

# Current capability

- Cardiac
  - Monodomain and bidomain for a wide range of practical problems
  - Efficient parallel implementation
  - Open source (LGPL) release available
- Soft Tissue Mechanics
  - Non-linear (finite deformation) elasticity
  - Discrete cell-based models
- Cancer
  - Focus on colorectal cancer and tumour spheroids
  - Off-lattice cell-based simulations
  - Variety of cell models and field equations

# How does Chaste use CellML?

- Cardiac ionic cell models described in CellML
- Automatic conversion to C++ code using Chaste classes by PyCml
- Key concerns are correctness and efficiency of generated code
  - The preDiCT project aims to achieve faster than real time cardiac simulations using Chaste

# Computer Science perspective

- CellML looks like a (domain specific) programming language
- Interesting dynamic evaluation (semantics)
- Static (i.e. compile-time) checks
- Provably correct optimisations

# Achieving faster simulations

- Hand optimisation doesn’t scale
- Compiler optimisation is helpful
- But compilers only perform general optimisations
- There are also domain-specific optimisations
  - Staging work in an ODE solver
  - Lookup tables

# Staging by Partial Evaluation

- An ODE solver is essentially a loop over time
- Some computations are the same at every time-step and depend only on information available within the model
  - So perform them once only
- The context is too complicated for a compiler to do this for us so use a partial evaluator
- A partial evaluator is an automatic tool that pre-computes parts of a program known at compile time

# Lookup Tables

- Many expressions depend only on the transmembrane potential V
  e.g. $\beta_h = \frac{1}{e^{-(V+45)/10}+1}$
- Usually V takes values in the range [-100,60] mV
- We can thus:
  - Tabulate expression values prior to simulation
  - Use linear interpolation to look up a value for the expression given V
  - This is faster than computing an exponential

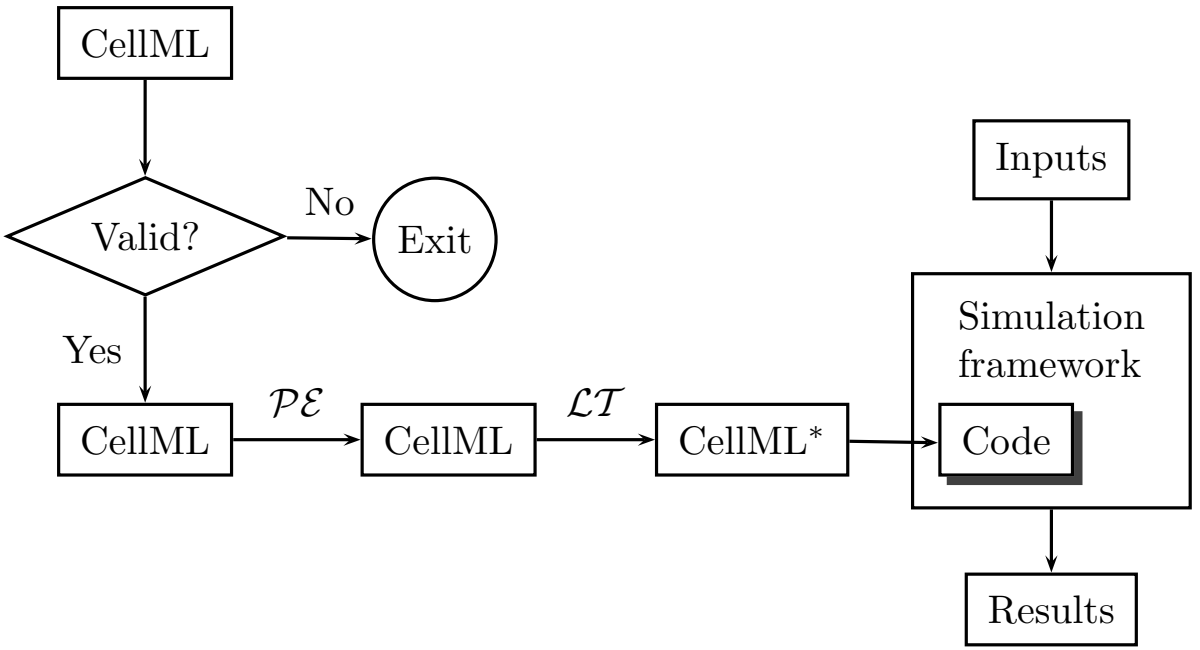
# Automatic Lookup Tables

- Analysis of when to use tables can be automated
  - Check variables used to compute the expression
  - Check for occurrence of expensive functions
- *A posteriori* error analysis to evaluate accuracy penalty
  - Can evaluate error in functionals of the solution

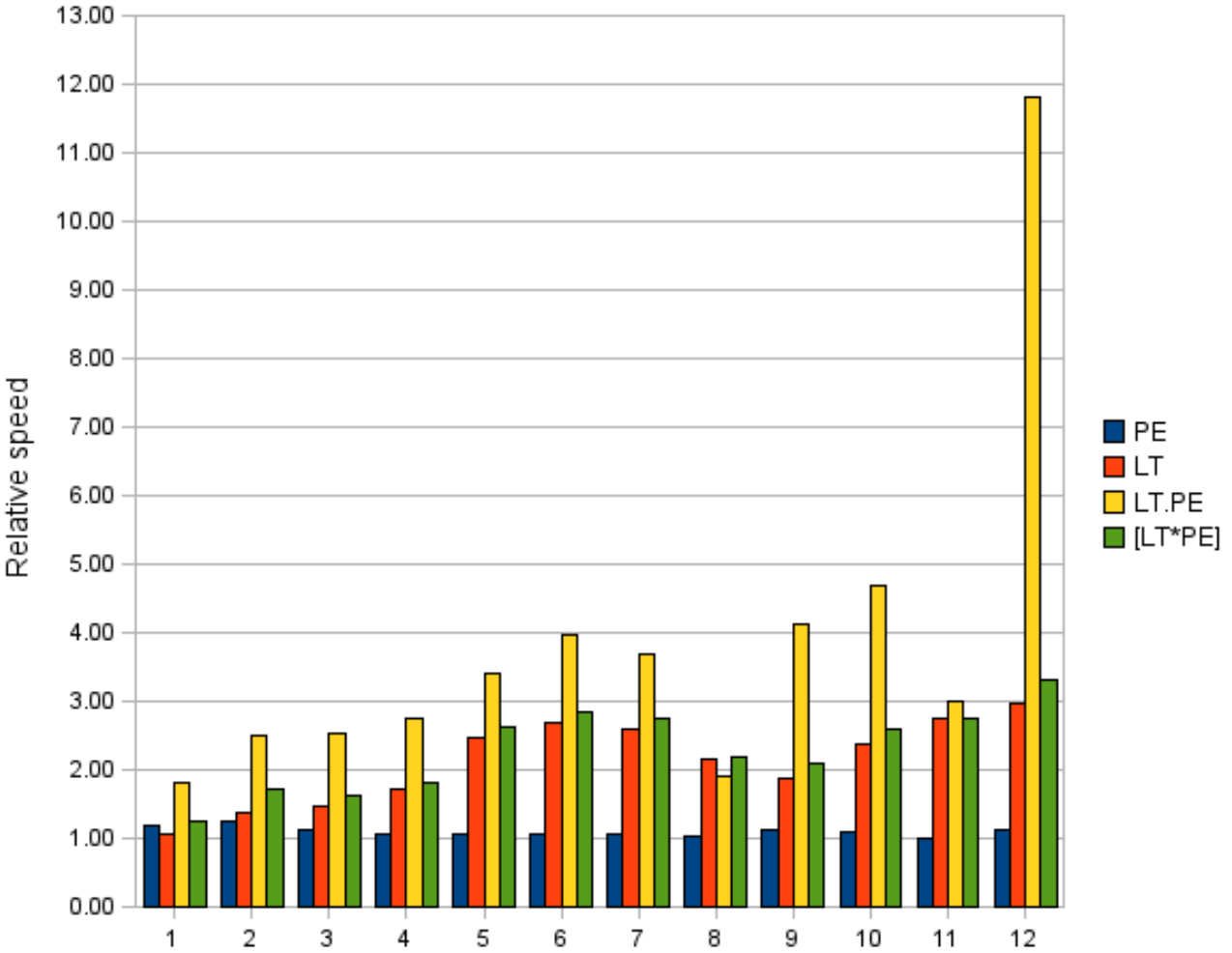
# Automatic Lookup Tables

- We can allow variables other than V to occur in the expression
  - Constants, and variables whose values depend only on constants, are OK
  - Key point: is the value known when the table is generated?
  - This kind of analysis is done by partial evaluation
  - So do partial evaluation then lookup tables

# Optimisation Framework




# Experimental Results — Single Cell




# So what can PyCml do now?

- Check models for units consistency, and apply automatic conversions
- Generate C++ code compatible with Chaste completely automatically
- Apply partial evaluation and lookup tables automatically (producing CellML or Chaste code)
- Generate a decoupled cell model which solves itself using backward Euler (for some models)
- Various options to fine-tune these processes
- Also generate Matlab code for *a posteriori* error analysis, etc.

# Next Steps

- Improvements to ODE solver performance in tissue simulations
  - e.g. changes to code structure for better cache utilisation
- Automate other optimising transformations
  - e.g. detect slow/fast currents
- Base work upon the CellML API
- Utilise (biological) metadata
- Use CellML and/or SBML in Cancer Chaste

# Acknowledgements


The Chaste team:

- Joe Pitt-Francis
- Miguel Bernabeau
- Pras Pathmanathan
- James Osborne
- Alex Fletcher
- *et al.*

Funding:







Key publications:


- Pitt-Francis *et al.*, Phil Trans Roy Soc A, 2008
- Pitt-Francis *et al.*, Computer Physics Communications (submitted)


`http://web.comlab.ox.ac.uk/chaste`

# Decoupled Cell Models

- In a tissue simulation, solve cell models using backward Euler
  - $V_m^n$ obtained from PDEs
- Update linear ODEs directly:

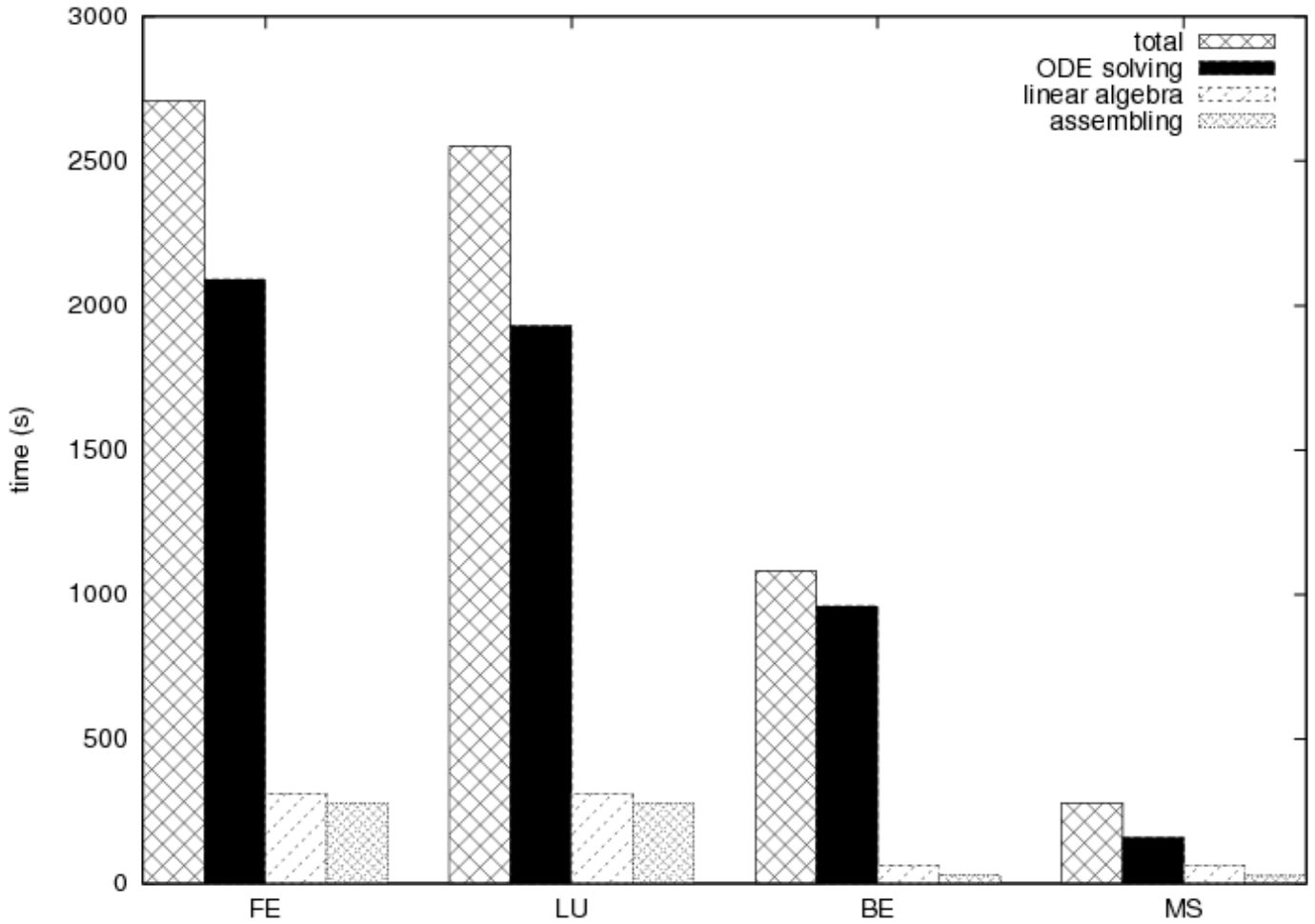
$$\frac{\mathrm{d}u_i}{\mathrm{d}t} = a_i(V_m) + b_i(V_m)u_i$$

$$\Rightarrow u_i^n = \frac{u_i^{n-1} + \Delta t_n a_i(V_m^n)}{1 - \Delta t_n b_i(V_m^n)}$$

- Update remaining ODEs using Newton's method:

$$\mathbf{g}(\mathbf{u}^n) := \mathbf{u}^n - \mathbf{u}^{n-1} - \Delta t_n f(\mathbf{u}^n, V_m) = 0$$

# Some Chaste Results




20 mm$^2$, monodomain, Noble'98