# A Multi-Scale Agent-Based Simulation Framework for Vascular Tissues using the Chaste Library

James A Grogan[1], Anthony J Connor[1,2], Philip K Maini[1], Helen M Byrne[1], and Joe Pitt-Francis[2]

[1]Wolfson Centre for Mathematical Biology, Mathematical Institute, University of Oxford, Oxford, OX2 6GG, UK.
[2]Department of Computer Science, University of Oxford, Oxford, OX1 3QD, UK.

**Abstract**

Multi-scale agent-based modelling is a popular technique in the study of vascular tissues in the areas of cancer, wound healing and development. Problems of interest in these areas involve the solution of blood flow, solute transport, cell cycling, vessel re-modelling and angiogenesis sub-problems. The development of software for solving such problems is a time-consuming process, due to the multiple physical processes and modelling techniques involved. There are a variety of agent-based methods used in the literature for representing both cells and vessels, including lattice-based and lattice-free approaches, and for solving transport problems involving discrete sinks and sources. Despite the recognized importance of model prediction cross-comparison, the expense of model implementation, variety of previosuly adopted modelling approaches and lack of 'free' open source code mean that little progress has been made in model benchmarking to date. Here, a new agent-based framework for simulating vascular tissue growth and treatment response is presented, based on the open source C++ library, Chaste (Cancer, Heart And Soft Tissue Environment). The framework is designed to address the conflicting requirements in vascular tissue modelling of computational performance, extensiblity and rapid model prototyping by using object oriented programming in C++, established linear algebra libraries (uBlas and PETSc), and Python bindings. In this study, the framework is overviewed, with example applications showing its use in model cross-comparison for well-known 'snail-trail' type angiogenesis problems using on- and off-lattice approaches and two- and three-dimensional representations. Problems of this type are of relevance in the areas of cancer, wound healing and development. It is envisaged that the framework will be useful in future vascular tissue model development and benchmarking, while features such as automated Python wrapping and dimensional analysis may be of general interest in the development of scientific software.

**Keywords:** Chaste - agent-based simulation - multi-scale model - vascular tumour growth - on-lattice model - off-lattice model

**AUTHOR SUMMARY:**

- New simulation framework for rapid vascular tissue agent based model development.
- Includes unit-tested code, detailed API documentation and tutorials.
- Predictions of several different modelling techniques compared for the well know 'snail-trail' angiogenesis problem.

# Contents

**1 1. Introduction** **3**

**2 2. Multi-Scale Agent-Based Vascular Tissue Models** **5**

**3 3. Code Design and Implementation** **6**

**4 4. 'Snail Trail' Example Problem** **8**

**5 5. Discussion and Future Work** **8**

**6 6. Conclusions** **8**

# 1. Introduction

Multi-scale agent-based modelling is a popular method for simulating tissue growth and treament response in the areas of cancer, wound healing and developmental biology. These approaches typically involve modelling cells as discrete entities, with indivudual cell-cycling and spatial positioning described using lattice-based [] or lattice free methods []. Cell interactions with each other can be through lattice-neighbour interactions [] or nearest-neighbour approaches based on touching spheres [], for example. Cells can interact with chemical fields by acting as discrete or distrbuted sink or source terms in partial differential equations arising from mass balances. There now exists a range of flexible, open-source frameworks for implementing the described multi-scale problems, including Chaste [], OpenCMISS [], CompuCell3D [] and BioFVM [] amongst others. In addition, a variety of bespoke, problem-specific, frameworks have been developed, often using MATLAB [][][].

Despite the prevalence of agent based software for *cells*, the aforementioned frameworks do not account for *blood vessels* in a detailed way (i.e. other than representing a vessel as a cell-like point in space). Blood vessels play important roles in nutrient and drug delivery in cancer, wound healing and development. In the former, the vessels themselves are often the target of a therapy [], and it is of interest to determine how changes in vascular structure affect tumour treatment response. Multi-scale agent based problems involving blood vessels typically involve the solution of network blood-flow problems, haematocrit, drug and other nutrient delivery by vessels, vessel structural adaptation in response to flow and other cues, sprouting angiogenesis and vessel regression sub-problems, in addition to the sub-problems used in cell-based simulations [][]. Further, cells and vessels can interact in terms of spatial occupancy, or vessels can themselves be comprised of multiple cell agents [].

The additional requirements for vascular tissue simulations and the wide variety in cancer, wound healing and development problems of interest mean that any framework needs to be able to account for multiple physical processes and different agent-based modelling strategies. There have been several studies into the requirements for software implementing models of this type []. In breif, it is important that the software framework is *flexible*, in that models can easily be composed of a collection of interchangeable sub-models and *extensible*, in that new models can be easily developed using common functionality from other models. Given that problems often involve multiple solutions of system of ordinary- and partial- differential equations (ODEs and PDEs) with many agents in the tissue volume, computational *efficiency* is also important. Due to current trends in scientific research, it is also desireable that the framework software and documentation are *available* under a suitable 'free' open-source license [], so that both source code and results can be inspected and replicated by the community [].

Existing open-source agent-based frameworks for cells do not contain sufficient functionality for many vascular tissue problems, while currently published frameworks for both cells *and* vessels, overviewed subse-

quently, do not yet cover all of the deriserable attributes of *flexibility*, *extensibility*, *efficiency* and *availablility*. A widely popular approach for multi-scale agent-based modelling of cells and vessels is the use of MATLAB, often with expensive parts of the simulation written in C or C++ and pre-compiled [][][]. This approach offers *efficiency* and *flexibility*, in that code has the potential to be reasonably performant and the MATLAB environment allows rapid model prototyping. Limitations are that proprietary software is used, limiting *availablility*, and MATLAB's matrix based language is needed, limiting opportunities for *extensibility* afforded by languages with greater focus (from both a language and developer perspective) on object oriented programming, like C++ and Python. Of the many models (and modeling frameworks) developed in such a manner, few have made their source public to the authors knowledge. Exceptions are [] and []. It is also common to write models and frameworks in C or C++ [][][], offering scope for efficiency and, in the latter, extensibility. Flexibility is compromised, given that software needs to be recompiled and linked for any changes, which can be time consuming in a relatively large code-base. Again, it is rare for source code to be made publically available, with the exception being several studies by Secomb and co-workers [][]. The open-source software of Secomb and co-workers is an in-valuable resource in this area, however it mostly does not follow object oriented design practices [] nor contain significant documentation. This limits oppurtunity for model extension.

As has been noted, it is rare for software associated with models of this type to be made publically available. This, in addition, to the significant time required in implementing such frameworks has limited the ability to cross-compare and reproduce models and model predictions. The important of, and challenges in, cross-comparison of such models has been discussed in the recent review of R et al []. In particular, it is noted that given a widespread difficulty in comparing models with experimental data for many vascular tissue problems of interest, it is important to verify that models developed to data are at least consistent with one another.

The aim of this study is the development of a 'free' open-source multi-scale agent-based simulation framework for vascular tissues. The framework is designed to balance *extensibility*, *flexibility* and *efficiency* as best possible, allowing future rapid-prototyping of new models and benchmarking of existing models. Given that problems of interest are a super-set of those already addressed by existing cell focused agent-based framerworks, it is sensible that the new framework builds on one of these. The new framework is based on the Chaste (Cancer, Heart And Soft Tissue Environment) open source C++ library [], which uses object-oriented programming including templating, implements a number of agent-based modelling techniques for cells [], interfaces with well established linear algebra libraries (uBlas [], PETSc []), ODE solvers (CVODE[]) and visualization software (VTK []) and has suitable unit-test, documentation and tutorial coverage []. A limitation of Chaste is that, due to being written in C++, it is difficult to quickly prototype models or over-ride existing methods without re-building the software. To address this limitation, the new framework uses automatically generated Python bindings to facilitate rapid model creation, allowing integrated model

development and post-processing with a rapidly growing collection of scientific tools for Python [].

The remainder of this study is now overviewed. Section 2 overviews common components of vascular tissue models, as described in the literature. Section 3 gives a high level over-view of the code design and implementation details. Due to the close linke with Chaste, the design of that software is also briefly summarized, however the reader should refer to [][] for further details. Section 4 demonstrates the use of the code, by comparing the predictions of different models of a well-known 'snail-trail' [][] angiogenesis problem in 2-D and 3-D and with on- and off-lattice representations of cells and vessels. The results in Section 4 may themselves be of scientific interest, given that it is the first detailed comparison, to the authors knowledge, of the predictions of several modelling paradigms. Section 5 overviews code availability, future directions and other potential applications.

## 2. Multi-Scale Agent-Based Vascular Tissue Models

This section gradually introduces the typical components of a multi-scale agent-based vascular tissue model and gives an indication of how they might be implemented in a simplified software framework. The approach follows that of Connor et al. [], although the actual implementations of the frameworks differ, reflecting implementation constraints.

A fundamental component in a vascular tissue model is the network of vessels, blood or lymphatic. A common method for representing vessels $\mathcal{V}$ is as a collection of straight-line segments $\mathcal{S} \in \mathcal{V}$ which have end points $\mathcal{N} \in \mathcal{S}$, as shown in Fig. 1. The end points, or vessel nodes, have spatial location $\mathbf{x}$ in $\mathbb{R}^n$ where $n$ is spatial dimension. The spatial location can be restricted to a regular lattice [][] or lattice-free descriptions can be adopted.

with further details given in Section 3. The introduced model is loosely based on that of Owen et al, for vascular tumour applications []. Many models in the literature can be identified as being composed of a sub-set of the components of this sample model [][][].

Model components are shown schematically in Fig. 1, while a simplified unified modeling language (UML) diagram for how it might be implemented in software is shown in Fig 2. Cells are represented as discrete 'agents'. Each cell can progress through a cell cycle and divide, become quisescent or apoptotic at suitable times. Progresssion through the cell cycle can range from a collection of simple, pre-imposed timing events [], or can be due to the solution of detailed sub-cellular reaction network problems, depending on nutrient and stimulus fields around the cells and intercellular protein concentrations []. A simplified UML diagram for a cell cycle is shown in Fig. 2 (a). Cells can have different user defined 'mutatation states', such as 'Wild Type', 'Cancer', 'Endothelial Tip' which determines how they might progress through the cell cycle or produce and consume stimulii. User defined 'proliferative types', such as 'Stem' and 'Differentiated' are also possible, affecting how a cell might produce progency. Cells occupy positions in two-dimensional (2D) or

three-dimensional (3D) space. Positions can be restricted to a regular lattice [], determined by the solution of a simple contact mechanics problem [] or use a range of other representations [][]. A design decision in Chaste means that cells and their locations are stored seperatly in a 'cell population', shown schematically and in UML form in Figs 1 c and 2c respectively.

A variety of representations are adopted for vessels [][], but a common approach is to define them as a collection of straight line segments. As per cells, segment end-points (denoted vessel nodes here) can be restricted to a regular lattice [] or have lattice-free descriptions [], again with the possiblity of mechanical interactions [] (not implemented in the present framework). A vessel network is a collection of vessels, shown schematically and in UML form in Figs 1 d and 2d respectively.

Given the described representations of cells and vessels, growth, flow, nutrient transport, angiogenesis and vascular remodelling problems can be solved. Given the size-scales involved in agent based modelling, blood flow problems are usually solved in micro-vessels. A typical approach is to represent vessels as a collection of cylindrical tubes and assume Pousille flow in each segment []. Flow in the entire vessel network can then be solved by assuming a simple mass balance where vessels meet and suitable inlet and outlet boundary conditions []. A sample flow solver is shown schematically in Fig. 3. A complication for micro-vessels is that the effictive viscosity of the blood flow depends on vessel diameter and the density of flowing red blood cells (haematocrit) [][]. Therefore, it is necessary to solve a haematocrit transport problem in tandem with the flow problem, as shown in Fig. 3 b. A further complication, is that vessel radius is known to change as a function of flow induced wall shear stress, hydrostatic pressure and a collection of flow-dependent metabolic cues [][]. These quantities therefore must also be calculated in tandem with the flow problem. A typical approach to solving this collection of coupled problems is to iteratively solve each problem in series, update solution variables and assess convergence to a tolerance. In terms of software development, a class can be used to contain and iterated through any required solvers for the coupled flow problem, as shown in Fig 3c.

## 3. Code Design and Implementation

Chaste is a C++ library with code that can be used to build a range of models in the area of computational biology. The code is over-viewed in detail in several publications [][][]. In brief, the core components are:

- global - basic mathematics, check-pointing and time stepping
- io - reading and writing of various file formats, including HDF5
- mesh - handling linear and quadratic meshes, including reading and writing
- linalg - linear algebra, including PETSc and uBLAS wrappers
- ode - ODE system solvers, including CVODE interface
- pde - PDE solvers using the finite element method
- continuum mechanics - non-linear elasticity problems

Beyond the core components further code is specialized for discrete cell problems and cardiac electrophysiology problems. The code for modelling discrete cell problems is further divided into []:

- cell - cell description, cell cycle models and mutation state definitions
- population - collections of cells described using cellular automaton, mesh based, node based, vertex based and cellular potts frameworks
- simulation - tissue growth using lattice and off lattice approaches
- writers - solution output to ascii and VTK files

Chaste can be extended by add-on user projects, which can share the same dependencies and build and test frameworks []. The vascular tumour growth functionality is implemented as such an add-on project, but with additional functionality implemented through Python bindings. As such the code is divided into a C++ framework, which follows a similar design to cell-based Chaste framework [] and a Python package, which encompasses most of the C++ functionality, but with easier integration with other solvers and libraries.

The primary C++ components are:

- vessel - vessel network reading, writing, analysis and management
- geometry - simple geometric descriptions and operations
- mesh - finite element meshing and regular grid tools
- pde - extensions to core PDE solvers with finite difference and semi-analytical methods
- simulation - solvers for flow, angiogenesis and structural adaptation. Modifiers for integration with cell based Chaste.

The code heavily uses Chaste core functionality, but is only loosely integrated with cell-based functionality through a 'simulation modifier' pattern. This pattern allows loose coupling during simulations between discrete representations of vessels and cells, through the solution of PDEs in which either agent can be a discrete sinks or sources or through simple models of mechanical contact. The new framework is developed with Python binding in mind. This leads to extensive use of shared pointers and factory constructor methods and reduction in the use of templating relative to the remainder of the Chaste code. Unrelated to Python binding, there is also limited parallelism and checkpointing functionality at present.

The Python framework is available as a standard Python package []. Due to fundamental differences between C++ and Python in regard to templating and memory management it does not encompass all of the functionality of the C++ framework. In particular all simulations assume a spatial dimension of 3 (i.e. 3D) and some extra wrappers are needed to generate meshes, cell populations and simulation instances. Extra functionality is implemented in Python, giving the following additional components:

- geometry - CAD operations and centreline extraction
- image - image processing and segmentation
- interface - interfaces with external finite element solvers
- plot - two dimensional and three dimensional post-processing and visualization

- gui - graphical user interface for model pre-processing steps

Both C++ and Python components are subject to unit-testing are documented and are accompanied with a collection of tutorials. Functionality with application in the solution of vascular tumour growth problems is demonstrated in the next section.

## 4. 'Snail Trail' Example Problem

## 5. Discussion and Future Work

## 6. Conclusions