A mathematical and computational framework for modelling epithelial cell morphodynamics

Fergus Cooper
Somerville College
University of Oxford

A thesis submitted for the degree of
Doctor of Philosophy
Trinity 2018

To dad.

Acknowledgements

Many thanks must go to my excellent supervisors Ruth Baker and Alexander Fletcher. How they had the patience to put up with me I may never know. Thanks, too, to my examiners Miguel Bernabeu and Peter Minary for their thorough examination of, and valuable feedback on, this thesis.

Many others have given me their time and expertise in ways directly beneficial to this work. Thanks, in particular, go to Jeremy Green, Jingjing Li, Jochen Kursawe, Martin Robinson, Gary Mirams, Joe Pitt-Francis, Jonathan Cooper and David Gavaghan.

During my time in Oxford I am fortunate to have been surrounded by so many excellent people, who have not only made my time here memorable but who kept me going when this thesis threatened to get the better of me. I must thank Joanna Raisbeck in particular, as well as Ben Lambert, Jean-Michel Johnston, Katharine Lauderdale, Hazel Tubman, Tobias Lutzi, Heidi Dritschel and Chris Lester. I have necessarily omitted many names, including from the Mathematical Institute, the Somerville College graduate community, and Oxford more widely, and I hope they will forgive a more general but nonetheless heartfelt thanks.

As well as those I have met in Oxford, I also wish to mention Jon Sherry, Tessa Simkins, Sam Ballance, Vicky Webb and Veronika Lipińska. Over the duration of my DPhil they have helped, in intangible ways, more than they know.

Finally, and most importantly, thanks to my parents, Alex and Jim Cooper. I could not have written this thesis without a lifetime of encouragement and support.

Abstract

Mathematical and computational modelling provides a useful framework within which to investigate the organisation of biological tissues. As experimental biologists generate increasingly detailed descriptions of cellular behaviour, models that consider cells as discrete entities have become a common tool to study how cell-level processes affect collective dynamics, form and function at the tissue level.

To date, however, models incorporating detailed biophysical descriptions of cell shape dynamics have gained little traction among the modelling community. Few model implementations are publicly available, and there often remains no comprehensive account of their method of solution, computational implementation, or analysis of parameter scaling, hindering our ability to utilise such models in practice. In addition, the quality of software underpinning such models (and academic research more widely) is coming under increasing scrutiny, with a growing recognition of the need for correct, reliable and sustainable research software tools.

This thesis aims to address these needs for one such cell-based modelling approach, the immersed boundary method. We develop and analyse the immersed boundary method and provide an efficient, open source implementation for simulating cell populations. The implementation is undertaken within Chaste, an open source C++ library that allows one to easily change constitutive assumptions, and is designed and built with software best practices in mind.

We explore such best practices and refine and add to the infrastructure and functionality of Chaste. We then carefully compare the immersed boundary method with the vertex model, a competing cell-based modelling approach for epithelial tissues, clearly elucidating relative strengths and weaknesses of the immersed boundary method. This furthers our understanding of the circumstances under which the immersed boundary method and similar cell-based approaches are an

appropriate computational tool. Finally, to demonstrate the efficacy of the immersed boundary method, we investigate the mechanics underpinning a novel form of epithelial bending, advancing our understanding of the formation of placodes, structures of epithelial thickening in the cranial region of developing embryos. Together, the contributions in this thesis advance the use of reproducible software development approaches in cell-based modelling.

Contents

List of Figures

List of abbreviations

Chaste	Cancer, Heart And Soft Tissue Environment
CPM	cellular Potts model
DFT	discrete Fourier transform
ESF	elongation shape factor
GDP	gross domestic product
GRF	Gaussian random field
IB	immersed boundary
IBM	immersed boundary method
JKR	Johnson–Kendall–Roberts
MVM	'many vertex' model
ODE	ordinary differential equation
PCD	polygon class distribution
PDE	partial differential equation
SEM	subcellular element model
SVG	scalable vector graphic
VM	vertex model

Chapter 1

Introduction

Embryonic epithelia, the sheets of tissue lining structures and cavities in the embryo, achieve complex morphogenetic movements including bending and folding through the coordinated action and rearrangement of individual cells. Recent technical advances in molecular and live-imaging studies of epithelial dynamics provide an excellent opportunity to better understand how cell-level processes facilitate these large-scale tissue rearrangements [90]. In combination with experimental approaches, computational modelling allows us to challenge and refine our current understanding of epithelial morphogenesis and to explore experimentally intractable questions. To this end a variety of 'cell-based modelling' approaches have been developed to describe how processes at the cell scale determine tissue-level behaviour. In many cases, however, the tools necessary to model observed biological processes are not yet developed, or are insufficiently refined to fully utilise the high-resolution information now available on the cellular scale. As such, the overarching aim of this thesis is to develop such mathematical and computational approaches, in order to improve our understanding of the role that individual cells play in facilitating tissue shape changes during morphogenesis, with particular emphasis on the development of placodes in developing cranial epithelia.

1.1 Background

Morphogenesis is frequently driven by the growth and deformation of epithelial tissues, which form polarized sheets of cells with distinct apical and basal surfaces and tight lateral attachments nearer their apical surface. The coordinated movement, shape change and intercalation of cells in an epithelial sheet facilitate complex morphogenetic processes, from tissue elongation through convergent extension [87], to

bending and invagination [98] and tube formation [92]. Well-studied examples of such processes include convergent extension in the mouse neural plate [171] and ventral furrow formation in *Drosophila* [128]. Importantly, understanding the functions of, and mechanisms giving rise to, these structures constitutes a significant step toward improving regenerative therapies and engineered tissues.

The mechanical forces driving epithelial morphogenesis affect individual cell morphologies, and are exerted on neighbouring cells and extracellular matrix through membrane-bound adhesion components, leading to tissue-level stresses and strains [10, 78, 81]. Cells can also be influenced by extrinsic forces arising, for example, from underlying tissues [20]. Until recently, such forces were not experimentally measurable, and thus the role of mechanics in morphogenetic processes not well characterized. This has changed, however, with recent advances in measurement techniques, in particular *in vivo* [158]. These advances, together with finer grained cell-scale imaging, improve the ability with which descriptions of mechanics can be probed and understood.

This interface between mechanics and predictive computational models is a driving motivation for this thesis. We begin by first reviewing the development of models that have sought to elucidate the mechanistic underpinnings of biological processes. This chapter draws on material from an invited review article published in *Philos. Trans. Royal Soc. B* [45], on which I am a co-author, and relevant sections are reproduced with permission.

1.2 Mathematical models of morphogenesis

Mechanistic mathematical modelling of morphogenesis dates back to the work of Turing, who first demonstrated symmetry-breaking pattern formation using reaction-diffusion models [166] (Figure 1.1a). Continuum models, based on Turing's and other mechanisms, have since been successful in advancing our understanding of the processes underlying developmental phenomena. The 'French flag' model, for example, links positional information and morphogen concentration with phenotype [173], while the 'clock and wavefront' model has provided a mathematical basis for observed specificity in somite number and size during somitogenesis [25]. More recent work has demonstrated that domain growth, of particular relevance to development, can have an impact on patterns and their robustness [29]. Such work demonstrates the enduring utility of partial differential equation (PDE) models in morphogenetic contexts, but these models typically remain phenomenological in

nature.

Increasingly, advances in experimental techniques allow us to probe developmental processes at a cellular and molecular level. As detailed cell-level information becomes available, and aided by advances in computer hardware, cell-based modelling approaches able to describe tissue-scale dynamics are becoming more prevalent. Such models have had success in explaining a broad range of biological processes. A brief overview of competing modelling approaches is given in Section 1.2.1; for more detailed reviews see, for example, [121, 163].

As the placode morphogenesis application described in detail in Chapter 5 highlights, we are able to capture cell shape dynamics with increasingly fine levels of detail, and so the nature of the biological questions being asked requires models capable of incorporating these more detailed descriptions of cell shape. When considering developmental processes characterised by movement and deformation on a cellular level, therefore, existing modelling frameworks summarised in Section 1.2.1 may not provide a detailed enough description of cell shape dynamics. Instead, we will require frameworks able to model cell shape on ever finer scales.

In a class of more detailed biophysical models of individual cell dynamics, cell shape is an emergent property. We refer to this class of models as being 'more geometrically detailed', as the description of cell shape is (theoretically) possible at arbitrary precision. Such models include the 'many vertex' model (MVM) [162], the subcellular element model (SEM) [113], and the immersed boundary method (IBM) [124]. These frameworks encode points representing spatial locations inside or on the cell membranes, and rules governing the motion of these points, and are summarised in Section 1.2.1.

1.2.1 Cell-based models

In this section, before describing these more geometrically detailed modelling frameworks, we first review a range of cell-based modelling approaches, focussing on off-lattice models.

Cellular Potts models. The cellular Potts model (CPM) is lattice-based, and seeks to incorporate cell shape by defining a cell as the union of contiguous lattice sites [55]. Each site carries a positive integer 'spin' representing the identity of the cell currently in occupation at that site. The basis of the model is a Hamiltonian energy function, H, which we seek to minimise by updating cell lattice site identities.

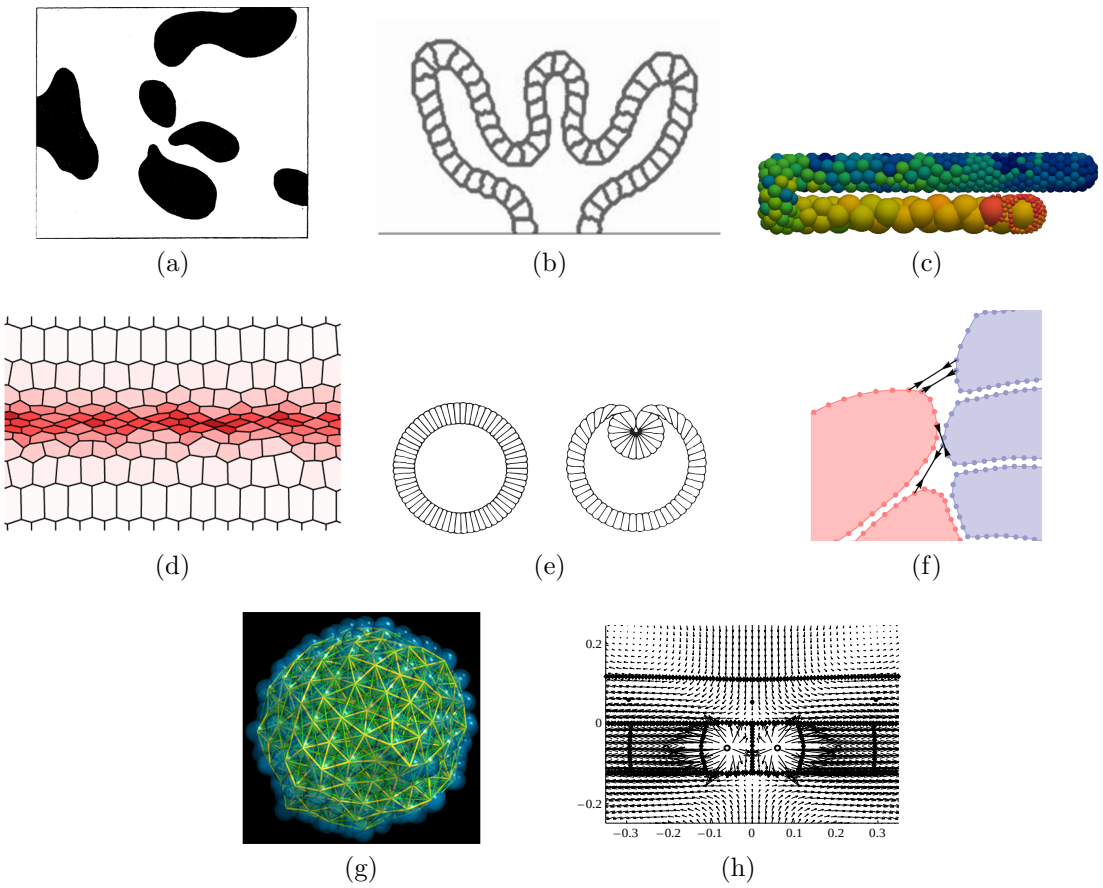


Figure 1.1: **Examples of mathematical modelling approaches for describing morphogenetic processes.** *(a) A continuous reaction-diffusion model of a morphogen system displaying a 'dappled' pattern, reproduced from [166] (Figure 2) with permission from the Royal Society. (b) A CPM of branching in the kidney, reproduced from [66] (Figure 4c) with permission from Elsevier. (c) A particle model of the C. elegans germ line, reproduced from [3] (Figure 8) with permission from The Company of Biologists. (d) A VM modelling the apical surface of cells during Drosophila ventral furrow formation, reproduced from [150] (Figure 2e) under the Creative Commons Attribution License. (e) A cross-sectional model of ventral furrow formation in Drosophila, reproduced from [116] (Figures 8a and 8h) with permission from Elsevier. (f) An MVM of Nematostella vectensis gastrulation, reproduced from [162] (Figure 3f) with permission from Elsevier. (g) A SEM, showing a tightly clustered ball of interacting cells, reproduced from [143] (Figure 4a) with permission from IOP Publishing. (h) An IBM model of the villous trophoblast bilayer, reproduced from [139] (Figure 5a) with permission from Springer Nature.*

A typical Hamiltonian takes the form

$$H = \sum_{<i,j>} J\left(\tau(\sigma(i)), \tau(\sigma(j))\right)\left(1 - \delta(\sigma(i), \sigma(j))\right) + \sum_{i} \lambda_v \left(V(\sigma(i)) - V_t(\sigma(i))\right)^2, \quad (1.1)$$

where the first sum is over neighbouring lattice sites, the second is over all lattice sites,

$\tau(\sigma)$ is the type of cell σ, J is a constant defining the surface energy between two cell types, δ is the Kronecker delta function, λ_v is a cell-volume penalization strength, V is the cell volume, and V_t is a cell-type-specific target volume. Note that, in two dimensions, these volume terms refer to surface area, but this description naturally generalises to higher dimensions.

Updating of the lattice sites is performed stochastically via a Metropolis algorithm, defining the probability p of a given lattice site changing spin as

$$p = \begin{cases} 1, & \text{if } \Delta H < 0, \\ e^{-k\Delta H}, & \text{otherwise,} \end{cases} \tag{1.2}$$

where ΔH is the change in Hamiltonian associated with the proposed spin change, and k is a constant analogous to temperature that governs the propensity with which energetically unfavourable spin changes are made.

One application of the CPM to developmental biology is a study of tube-branching in the kidney [66] (Figure 1.1b), where the authors used the CPM to investigate the impact of physical and chemical factors on branching morphology. While efficient CPM implementations exist that allow the simulation of many cells in two and three dimensions, it can be difficult to relate model parameters to real-world quantities, and there are risks of lattice anisotropies and cell fragmentation. It is worth noting that some such issues have been addressed: Mango and colleagues [93], for instance, include an explicit representation of cell cortical tension which, in their framework, improves cell cohesion and thereby mitigates problems of cell fragmentation.

Particle models. In these models, each cell is represented by a point in space and one or more parameters define the radius (or principal axes) of an ellipsoid centred at that point. Descriptions of cell–cell contact include the Hertz and Johnson–Kendall–Roberts (JKR) models [121]. In both cases, cell–cell interactions are modelled as soft-ellipsoid interactions with a certain potential function. The JKR model is a more advanced variant, which also considers adhesion [23].

Figure 1.1c shows an example of a particle model applied to the development of gonads in *C. elegans* [3], where the authors demonstrate that germ cell pressure may have a mechanical feedback on the cell cycle. Particle models are more physically motivated than lattice-based models and generalise naturally to three dimensions, however they lack any explicit description of cell shape.

Vertex models. In a vertex model (VM), cells are represented by polygons or polyhedra. Changes in cell shape over time are driven by the motion of points in two or three dimensional space, defined as the vertices of the polygons or polyhedra. Several geometric and mechanical assumptions are typically made in VMs, resulting in a coupled system of first-order ordinary differential equations (ODEs) for the evolution of the position of each vertex [47]. The VM is suitable for slow velocity, short-length-scale movements where inertial terms are negligible compared to viscous forces. These assumptions lead to overdamped motion governed by a system of equations of the form

$$\eta \frac{\mathrm{d}\mathbf{r}_i}{\mathrm{d}t} = \mathbf{F}_i = -\nabla_i E, \tag{1.3}$$

where η is a constant related to viscosity and defines the timescale over which motion occurs, $\mathbf{r}_i$ is the position of vertex i at time t, $\mathbf{F}_i$ is the sum of all forces acting on vertex i at time t, and E is the free energy, a phenomenological term we assume to be minimised by the motion of vertices (but which may be disrupted by certain other model choices, such as vertex rearrangements). Here, the index i runs over all vertices.

One widely adopted force law is that proposed by Farhadifar and colleagues [42], based on earlier work by Nagai and Honda [112]. We assume that the free energy of the epithelial tissue is comprised of three components: an area elasticity term, in which each element attempts to attain a target area; a perimeter contractility term, in which cells try to decrease their perimeter; and a line tension term associated with cell–cell adhesion. The free energy, E, is given by

$$E = \sum_{\alpha} \frac{K_\alpha}{2} \left(A_\alpha - A_\alpha^{(0)}\right)^2 + \sum_{\alpha} \frac{\Gamma_\alpha}{2} L_\alpha^2 + \sum_{<i,j>} \Lambda_{ij} l_{ij}, \tag{1.4}$$

where the first two sums are over all cells and the third is over all pairs of adjacent vertices; K_α, Γ_α and Λ_{ij} are, respectively, constants defining the area, perimeter and line tension penalties; A_α, L_α and l_{ij} are, respectively, the cell area, cell perimeter and junction length; and $A_\alpha^{(0)}$ is the cell target area. As vertices move to minimise the energy function (1.4), various vertex rearrangements occur, about which there is more detail in Section 4.1.1.

Figure 1.1d shows a VM of *Drosophila* ventral furrow formation [150], a widely studied model system for epithelial bending. The authors highlight that mechanical interactions can explain a number of important biological observations with the need for very little genetic regulation. While the VM allows straightforward

generation of experimentally testable summary statistics and explicitly incorporates a number of cell neighbour rearrangements, biophysical processes such as cell–matrix adhesion [168] and active cytoskeletal remodelling [131] are typically not considered. While there are numerous extensions to the VM, such as an attempt to incorporate curved cell–cell interfaces in order to improve the representation of cell shape [72], the VM fundamentally lacks the ability to faithfully represent complex cell shapes.

1.2.2 Incorporating mechanical complexity

Viscoelastic element model. While successful in recapitulating much of the gross behaviour of planar epithelial sheets, VMs lack resistance to shear deformation through active remodelling of cytoskeletal components. One approach to including cytoskeletal remodelling is to introduce viscoelastic elements representing the cell membrane and cytoplasm. This approach was first adopted by Odell and colleagues [115, 116], who modelled a cross section of an embryo as a ring of cells with interconnected vertices subject to a viscoelastic force (Figure 1.1e). The authors assumed that apical edges actively contract in response to stretch. With additional system-specific assumptions, this model recapitulated patterns of deformation as observed in, for example, sea urchin gastrulation and *Drosophila* ventral furrow formation.

Several more recent studies have focused on the different patterns of cell mechanical properties that can generate observed tissue deformations. For example, models of *Drosophila* ventral furrow formation have suggested a possible role for pushing by cells neighbouring the furrow, or buckling due to uniform tissue-wide changes in apical tension [130]. In contrast to the VM, these models include active cytoskeletal remodelling, however cells are still constrained to form confluent tissues.

1.2.3 Incorporating additional geometric complexity

'Many vertex' model. Another recent polygon-based model by Tamulonis and colleagues [162] represents each cell by a polygon in two dimensions comprising many vertices, allowing for much more complex cell shapes (Figure 1.1f), and the model incorporates the effect of filopodia, small projections from the cytoplasm containing actin filaments. Membrane elasticity is modelled by associating a linear spring with each cell edge, whose stiffness and equilibrium length varies according to whether the edge is apical, basal or lateral. The apical (and, in some simulations, basal) corners of neighbouring cells are also connected by very stiff springs, representing adherens

junctions. Apical constriction is implemented via an intracellular spring between each endodermal cell's apical corners. This model has been applied to study gastrulation of the starlet sea anemone *Nematostall vectensis*.

The authors were able to recapitulate key biological observations of bottle cell formation, invagination and zippering in *Nematostella vectensis* gastrulation, and were able to make a number of specific testable hypotheses based on their findings. The strength of such models is their explicit description of cell shape, but this comes at the cost of increased computational complexity, reducing the number of cells that can be simulated in practice.

Subcellular element model. The three previous models discussed share the common assumption that cell shape is well approximated by a polygon of specified degree. We next consider the (SEM), where discrete elements are used to represent both the cell membrane and cytoplasm. This model assumes that a cell's volume is divided into a number of subcellular elements. Each such element is represented by a point in space, which interacts with other such points via various potentials [144]. In addition to representing cells themselves, subcellular elements may be used to represent the extracellular matrix, the cell membrane, or individual organelles. The SEM was initially developed by Newman [113] as a flexible framework for simulating the detailed dynamics of emergent cell shape changes in response to mechanical stimuli.

In a similar way to the VM, each point moves as a result of an applied force, in an overdamped manner, and thus also obeys Equation (1.3). The simplest force law in the SEM is the gradient of potentials due to intra- and intercellular interactions, with additive Gaussian noise simulating the random motion of particles. This leads to an equation for the position $\mathbf{r}_{\alpha_i}$ of each subcellular element α_i of the form

$$\eta \frac{\mathrm{d}\mathbf{r}_{\alpha_i}}{\mathrm{d}t} = \xi_{\alpha_i} - \nabla_i \left(\{\text{intra}\} + \{\text{inter}\} \right), \tag{1.5}$$

where η is the viscous drag constant, ξ is a Gaussian-distributed random variable. Figure 1.1g shows an aggregation of cells modelled using the SEM.

The SEM allows detailed and emergent cell shapes in response to mechanical stimuli, but is computationally costly to simulate. This can be mitigated to a certain extent by careful parallel model implementation [64], but nonetheless remains costly. In addition, it may be difficult to relate the interaction function between different subcellular elements to specific cytoskeletal components or organelles.

Immersed boundary method. The immersed boundary method (IBM) is a numerical method to solve fluid–structure interaction problems and, like the SEM, allows for a richer variety of emergent and dynamic cell shapes than models such as the VM [27, 35, 124, 137, 138, 139, 164]. Figure 1.1h shows a bilayer of cells in the placenta, the villous trophoblast, modelled by the IBM, with arrows showing the underlying fluid flow. The authors used their IBM study to elucidate the roles of cell proliferation and cell fusion on the development of trophoblast tissue.

Originally developed to study the flow of blood around heart valves [123], the IBM considers the dynamics of one or more elastic membranes, which represent cell boundaries, immersed in a viscous incompressible fluid, which represents the cytoplasm, extracellular matrix and fluid [136]. The numerical scheme of the IBM allows the fluid–structure interaction to be decoupled from the discretisation of the immersed boundaries (IBs) in the sense that the fluid mesh does not need to conform to the geometry of the IBs. For this reason, after numerical discretisation, the fluid problem can be solved on a fixed grid that is independent of the geometry of the IBs, leading to efficient methods of solution. The fluid dynamics obey the Navier–Stokes equations with an imposed body force acting due to the elastic interactions of each cell. The precise functional form of this body force may be formulated rigorously as a strain relation [124], or else by decomposing it into intra- and intercellular interaction contributions [138]. Each IB moves due to the fluid flow without slipping. The numerical solution of this model involves discretising the fluid onto a regular square grid, while each IB is represented by a finite number of points along its length.

The first application to collective cell dynamics, by Rejniak and colleagues, focused on the growth of solid tumours under differing geometric configurations, initial conditions and progression models [137, 138]. Although not yet used extensively to model epithelial morphogenesis, the IBM has been applied in biology and elsewhere [35, 109, 139]. The flexibility of the IBM is exemplified by an application by Dillon and colleagues to vertebrate limb bud morphogenesis, where an IB now represents a tissue domain rather than a cell [34].

As with the SEM and MVM, the IBM incorporates an emergent cell shape from mechanical stimuli, however the fluid properties may be difficult to relate directly to biology. A much more thorough treatment of the strengths and weaknesses of the IBM can be found in Chapters 2 and 4.

Of the three more geometrically detailed frameworks described in this section, this thesis focusses on the IBM. While each has strengths and weaknesses, the MVM is not well characterised and has no reference software implementation. The

SEM, while well described, also has no reference software implementation and suffers from the increased computational overhead of representing the subcellular as well as the boundary elements, increasing the associated computational cost. The IBM is well characterised in a number of domains and has a mathematically rigorous underpinning. For this reason, we choose the IBM as the basis for much of the work in this thesis.

1.3 Software tools

The cell-based models outlined in Section 1.2 all require software implementations. No single framework has one canonical implementation, and no two implementations of the same underlying mathematical model are the same in every detail. In this section we introduce the ecosystem of simulation frameworks that are available, and highlight features and applications of several prominent implementations.

First, we must define 'available'. Computational results based on all modelling frameworks described in Section 1.2 are generated by running an executable program. That executable program results from code that has been written to perform some computational work, as specified by the mathematical rules of the framework being implemented. A software framework may be 'available' to the community if the underlying source code is published. This is 'open source' as anyone (in principle) can obtain the code and use it to reproduce the results presented by the authors. If the source code is not publicly available, it may be that the executable files necessary to reproduce the results are still published, and anyone (in principle) can still reproduce those results. We refer to this as 'closed source'. If neither the source code nor the executable programs are made available, there is no possibility of precisely reproducing published results as it is simply infeasible for every implementation choice to be detailed.

Not all software tools are available to the scientific community. The work by Tamulonis and colleagues on bottle cell formation [162], the work by Mao and colleagues on planar polarization [94] and differential proliferation rates [95], and the work by Sandersius and colleagues on modelling cell rheology [143], for instance, all use implementations of cell-based modelling frameworks that, at the time of writing, are not publicly available either as open or closed source software.

While this by no means invalidates the results presented by these and other authors whose computational implementations remain unpublished, strong arguments can be made for the software to be published alongside the results generated by it, and

for a more detail on this topic see, for instance [101, 172]. The first argument is reproducibility. Without the availability of software implementations, work can be very difficult for other researchers to reproduce, and this feeds the reproducibility crisis in academia. This leads into the second argument, which is wasted time (and by implication, money). There is a substantial overhead in time spent simply to recapitulate existing work when confirming, extending, or investigating further published results. While the process of implementing a modelling framework from scratch may well be beneficial for the implementers' understanding, published implementations greatly reduce the time to new results: adding new functionality to an existing implementation is clearly preferable over reinventing the wheel. This, though, leads to our final argument: quality. If an implementation is not open source, then the scientific community has no way of ascertaining its quality. By this, I mean the extent to which the software faithfully represents the underlying mathematical model, and whether bugs exist that might invalidate results generated by the software. Open source implementations can be scrutinised by the community, and poor quality code cannot easily be hidden. Indeed, entire journals such as the *Journal of Open Source Software*[1] and *SoftwareX*[2] now exist to publish open source software tools, allowing software implementations to be cited and peer-reviewed by the academic community.

These arguments highlight the need for high quality, open source implementations in cell-based modelling. As already highlighted, far from all computational modelling implementations have been made public. There are many, however, that have. While this section is not intended as an exhaustive list of software available for cell-based simulations, it is instructive to outline a number of popular frameworks together with some of their strengths and weaknesses. Table 1.1 shows a high level overview of several frameworks.

In my experience, general considerations for each of these pieces of software include the following. Who is the target user: experienced programmers, or scientists with little computational experience? What is the scale of the target biological problems: dozens of cells in high levels of detail, or billions of cells? Where will the code be run: on a laptop or a supercomputer? There are a large number of different cell-based computational frameworks, with each balancing these considerations to different degrees. As a result, each framework has unique characteristics that make it more or less suitable for addressing particular biological questions, and more or less

[1] `https://joss.theoj.org/`

[2] `https://www.journals.elsevier.com/softwarex`

Software package	Open source	GUI	On-lattice		Off-lattice					Refs
			CA	CP	PM	VT	VM	IBM	SEM	
Chaste	×		×	×	×	×	×	×[1]	×[2]	[107, 127]
CompuCell3D	×	×		×						[161]
Morpheus	×	×		×						[153]
EPISIM		×		×[3]	×					[159]
CellSys		×			×					[69]
PhysiCell	×				×					[51]
Biocellion					×					[75]
VirtualLeaf	×	×					×			[103]
LBIBCell	×							×		[164]
EmbryoMaker	×	×			×[4]				×[5]	[96]

Table 1.1: **Software tools for computational modelling.** *GUI: graphical user interface. CA: cellular automata. CP: cellular Potts. PM: particle model. VT: Voronoi tessellation. VM: vertex model. IBM: immersed boundary method. SEM: subcellular element model. Notes:* [1] *the IBM is implemented in a separate named branch of the Chaste software repository, the implementation of which is the subject of Chapter 2 ([27]).* [2] *a partial implementation of the SE exists in Chaste, but is not yet publicly accessible.* [3] *EPISIM interfaces with a variety of on- and off-lattice biomechanical models.* [4] *EmbryoMaker uses a mixture of spheres and cylinders, but most closely relates to particle models.* [5] *EmbryoMaker implements SEM-type dynamics for certain cell types, but this does not appear to incorporate subcellular resolution of cell shape.*

suitable for particular users.

Chaste. Chaste (Cancer, Heart And Soft Tissue Environment) is an open source simulation package for the numerical solution of mathematical models arising in physiology and biology. Chaste, and its extensions, are used and described in detail throughout this thesis. Its development has been driven primarily by applications to continuum modelling of cardiac electrophysiology ('cardiac Chaste'), to discrete cell-based modelling of soft tissues ('cell-based Chaste'), and to modelling of ventilation in lungs ('lung Chaste').

Cell-based Chaste focusses on efficient and validated implementations of a variety of cell-based modelling frameworks and, as such, it places an emphasis on reproducibility and employs extensive automated testing to ensure software quality and reliability. Chaste includes implementations of the cellular automaton model, the cellular Potts model, cell-centre models (including particle based and Voronoi tessellation), and the VM [118]. In addition, Chapter 2 (published; [27]) details an implementation of the IBM in Chaste, and this implementation is explored and used in this thesis. Chaste also has an implementation of the SEM [64], although this is not yet complete nor publicly available. The particle model is implemented in

parallel [64] and scales to $\sim 10^6$ cells on supercomputers.

While Chaste has focussed on a wide range of cell-based modelling frameworks, and has an emphasis on high-quality well-tested code, it is a complex C++ library with a steep learning curve for those without a software development background.

As of September 2018, 39 publications have resulted from cell-based Chaste. Selected applications have included the dynamics of intestinal homeostasis and carcinogenesis [2, 38], vascular tumour growth [60] and the dynamics of embryonic epithelia [84].

CompuCell3D. CompuCell3D is a software package that implements the CPM to model the morphology of cells, and is the successor to a previous software, the Tissue Simulation Toolkit [102]. Application development has focussed on multiscale modelling, with support for subcellular simulations using integrated ODE and PDE solvers. The software is open source, and available for download both as source code and in binary form for cross-platform use[3]. Via a graphical user interface, users can run simulations without significant programming experience.

Development of CompuCell3D is ongoing; it been under continuous development for at least the last six years. A diverse range of studies have been made possible by CompuCell3D, including understanding the roles of cell adhesion and proliferation in kidney disease [8], the CD8+ T-cell immune response [49], and somitogenesis [33].

Limitations of the software include the inability to simulate large cell populations (10^5 and above) [51], and only having the choice of the CPM for representing cellular biophysics.

Morpheus. In many ways, Morpheus is similar to CompuCell3D. It implements the CPM as the underlying biophysical model, and has built-in ODE and PDE solvers to allow the coupling of cell biophysics to subcellular dynamics. The software is open source, and available both as source code and as cross-platform binaries[4].

A great strength of Morpheus is that it is user friendly: an intuitive graphical user interface allows simulations to be performed without writing any code whatsoever. Development of the software is still active. An example study using the CPM biophysical model in Morpheus looked at the role of vascular endothelial growth factor in vascular network patterning [80], while a recent study used the ODE solution capabilities of Morpheus to understand the fluid dynamics of bile in the liver [104].

[3] `http://www.compucell3d.org/`

[4] `https://imc.zih.tu-dresden.de/wiki/morpheus/doku.php`

As with CompuCell3D, limitations of Morpheus include the size of cell populations that can be simulated, and the lack of alternative biophysical models.

EPISIM. EPISIM uses 'process diagrams' inside a graphical interface to allow the easy composition of the problem description. This is achieved by use of the Systems Biology Markup Language (SBML)[5], which allows models to be imported and exported from the software in a portable format that could, in theory, be loaded directly into other software compatible with the SBML. The model description is then translated automatically into executable code to allow two- and three-dimensional simulations through a closed source simulation framework. The underlying biomechanical models implemented in EPISIM include an ellipsoid-based particle model and a lattice-based model.

EPISIM has been used to model wound healing [141] and barrier formation in the epidermis [160].

While the intuitive model construction is designed to allow those without a computational background to be able to conduct simulations, this, coupled with the closed source implementation, significantly limits both extensibility and the extent to which precise details of the model solution can be determined.

CellSys. CellSys implements an off-lattice particle model, with built-in support for cell cycle, signalling and migration. CellSys employs a graphical interface for constructing models, and has a specific focus on performance, making use of OpenMP[6] for shared memory parallelism, allowing simulations in excess of 10^5 cells.

To date, applications have focussed on the liver, including studies on liver tissue regeneration [70, 68, 37] and disease [146]. Similar to EPISIM, CellSys is closed source, but free for academic use and, as such, suffers from the same problems of extensibility and detailed understanding of the underlying algorithms. The future direction of CellSys appears to be a new package, TiSim, which will be an open source successor, although few details are available at the time of writing.

PhysiCell. PhysiCell is another mechanistic particle model implementation with facilities to simulate the cell cycle, cell death, volume changes, motility and adhesion. The software is open source and cross platform, and simulations can be run on both desktops and supercomputers.

[5] `http://sbml.org/Main_Page`
[6] `https://www.openmp.org/`

PhysiCell brings high-performance off-lattice simulations of cell dynamics to an open source project, and aims to be an independent codebase able to cross-validate predictions made by other frameworks including Chaste, CellSys/TiSim and Biocellion.

The target application area for PhysiCell is cancer biology and, while the software is only recently released at the time of writing, one study describes the use of PhysiCell in cancer hypothesis testing [119].

Biocellion. Biocellion is a closed source commercial software solution for the simulation of extremely large cell populations, using a particle model framework. Biocellion is capable of simulating $\sim 10^6$ cells on a desktop computer, scaling up to $\sim 10^9$ cells on a supercomputer, several orders of magnitude more cells than can be simulated by any competing simulation framework.

While the software is free for academic use, it is closed source. This prevents detailed knowledge of the implementation details and extending of functionality, and it is impossible to assess to what extent the functionality is rigorously tested. The software has not yet seen wide adoption despite being the only solution for simulating billions of cells, but one simulation study demonstrates the applicability of such capabilities to the patterning of yeast colonies [76].

VirtualLeaf. VirtualLeaf is a simulation framework for modelling plant tissue growth and development. The software is open source, and consists of an implementation of the VM, but one in which relative motion between cells is restricted; this being appropriate for the study of plant tissues. VirtualLeaf incorporates ODE-based transport models, as well as PDE reaction–diffusion models coupled to the underlying mechanical model. A main difference compared to the VM described in Section 1.2, in addition to the restricted relative cell motion, is that the energy function is minimised by use of a Monte Carlo algorithm; vertices are moved in a random direction with movements rejected if the energy function is not reduced.

Due to the specific requirement of plant tissue modelling, VirtualLeaf is necessarily not widely applicable beyond its main application area. Studies to date include root vascular patterning [100], growth and patterning during vascular tissue formation [15], auxin transport [36] and emergence of tissue polarization [170].

LBIBCell. LBIBCell is, other than Chaste, the only software framework to implement any of the more geometrically detailed models described in Section 1.2.3.

LBIBCell implements the IBM using a Lattice–Boltzmann-based fluid solver, and has a focus on coupling cell signalling with the biophysical model of cell geometry.

LBIBCell is open source[7], but at the time of writing no development of the code is evident since early 2015, and no simulation studies that make use of it have been published. A major drawback to LBIBCell is the low number of cells that can be feasibly simulated on a desktop computer, but this is a consequence of the increased geometric and biophysical realism of the model itself, and not a criticism of the software implementation.

EmbryoMaker. EmbryoMaker implements a model that is capable of explicitly representing epithelia, extracellular matrix and mesenchyme as having different mechanical properties. This has been identified as important is a range of different biological scenarios [11], and so the EmbryoMaker framework has been built to treat the three differently: epithelial cells as cylinders, and extracellular matrix and mesenchymal cells as spheres, with differing properties. While drawing on the SEM model, there is no indication that EmbryoMaker seeks to model complex cell shapes by using numerous subcellular elements.

An application of EmbryoMaker investigates the role of tissue growth and cell adhesion in early tooth morphogenesis [11], and a recent review showcases several aspects of the software [147].

Summary. The software frameworks highlighted here show a wide range of tools available for cell-based simulation, and each is best suited to a specific problem domain. I believe it is worth highlighting several of those domains. Biocellion leads the way in high-performance simulation of particle models, being the only simulation framework having demonstrated the simulation of billions of cells. PhysiCell occupies a similar application space, although does so in the open source domain, which is a major drawback to using Biocellion. Morpheus, while capable of simulating far fewer cells, leads the way in terms of ease of use, with an intuitive GUI allowing non-expert users to conduct simulation studies. Chaste occupies yet another niche: focussing on a wider variety of frameworks, and on high-quality code, being the only framework with a rigorous testing infrastructure.

Choosing the right tool for the job is important, and having a range of high-quality open source simulations tools available to the academic community is therefore an important step towards understanding the processes of developmental biology.

[7] `https://bitbucket.org/tanakas/lbibcell`

1.4 Thesis structure

In this chapter we have reviewed the mathematical and computational tools that have been developed to explore cell-scale problems in developmental biology. In the process, we have identified the need for more geometrically detailed cell-based models to tackle a class of problems where cell shapes are complex and emergent.

It is worth noting that work in this thesis includes an implementation of the IBM for cell-based simulation. Work on this implementation began in 2014, contemporaneous with that of Tanaka's LBIBCell implementation which was published in 2015. While it may have been possible to extend Tanaka's implementation had it been available before work on my Chaste implementation began, LBIBCell focusses on coupled morphogenetic problems, while my Chaste implementation focusses on the ability to modulate mechanical properties while interfacing with the rich utility of the underlying Chaste framework. While there is, inevitably, some overlap between my work and that of Tanaka, the work was independent and was tailored to address different biological questions.

There are, nonetheless, few computational implementations of such geometrically detailed models, and those that do exist are typically not open source or developed with software engineering best practices in mind. Perhaps as a result of this lack of availability, there are very few examples of such models being used to understand the dynamics of cell populations and, in addition, there are no comprehensive account of the relative strengths and weaknesses of such models, hindering our ability to effectively utilise them. This thesis addresses these key themes, and is structured as follows.

Chapter 2 addresses the need for high-quality implementations of more geometrically detailed cell-based models by implementing one such model, the IBM, within Chaste. This implementation is accompanied by a comprehensive account of the model, method of solution, computational implementation and analysis of parameter scaling, giving the firm grounding necessary to utilise it and to accurately compare it to other models.

Chapter 3 explores in greater detail the software engineering practices that underpins high-quality academic software, and describes work to develop and refine those practices within Chaste.

Chapter 4 addresses the need for careful comparisons between different cell-based modelling frameworks. This chapter compares the IBM with the VM on a benchmark problem of cell sorting and advances both the fundamental implementation details necessary for such a comparison, and our understanding of the IBM. No such comparison has been undertaken previously, and this work substantially increases our understanding of the scenarios under which the IBM is an appropriate tool for computational modelling.

Chapter 5 then demonstrates the efficacy of the IBM within developmental biology by applying it to understand the mechanics underpinning a novel form of epithelial bending. This work, in direct collaboration with experimental biologists, addresses a real and relevant question in developmental biology that hinges on out-of-equilibrium cell morphodynamics.

Main contributions of this thesis. Utilising an implementation of a more geometrically detailed cell-based model, my work advances the understanding of mechanics involved in a novel form of epithelial bending in a hitherto poorly understood biological system. This work, in direct collaboration with experimental biologists, used a novel computation model to advance our understanding of early-stage placode development and, further, demonstrated for the first time that such cell-based models can be effectively utilised to study problems involving out-of-equilibrium cell shapes. This not only proved the utility in a specific domain area, but importantly demonstrates the potential wider efficacy of such models in general.

While the headline contribution of this work is the demonstration that such models can be used to understand complex biology, other advancements to the state of the art are also presented. Work in this thesis for the first time contextualises the IBM next to other modelling frameworks, improving our ability to appropriately select computational models. Furthermore, the availability of an open-source implementation of an IBM for cell-based modelling, along with careful demonstration of convergence, provides the first such detailed account in the literature, serving as a useful benchmark for future researchers and library implementers. Finally, numerous technical advancements underpin work in this thesis, each of which is described in detail in the forthcoming chapters.

Chapter 2

Numerical analysis of the IBM for cell-based simulation

Having discussed the state of the art in cell-based modelling approaches and their software implementations, we now focus on one particular approach, the IBM, and undertake the first systematic numerical analysis of this model applied to cell populations. The contents of this chapter were first published in the *SIAM J. Sci. Comput* [27], published by the Society for Industrial and Applied Mathematics (SIAM).

2.1 Introduction

Chapter 1 reviews a range of cell-based models and highlights a subset that are more geometrically detailed and capable of explicitly representing complex cell shapes. However, a firm mathematical and numerical foundation for the analysis of such models, which is required for confidence in the conclusions drawn from them, remains lacking. Surprisingly few numerical analyses of such models have been undertaken; see [83] for a rare exception. To help address this gap we present a detailed examination of the IBM, one such geometrically detailed model, and a computational implementation thereof, designed to study interacting populations of eukaryotic cells.

The IBM is a numerical framework for simulating the dynamics of one or more elastic membranes immersed in a viscous Newtonian fluid. It was first developed by Peskin to investigate flow patterns around heart valves [123]. The model is formed from two coupled components: elastic boundaries representing, for instance, heart valves or cell membranes, and a fluid extending over the entire spatial domain. The

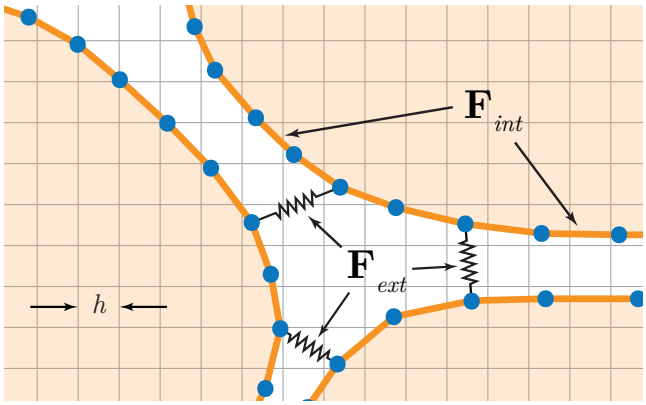


Figure 2.1: **Schematic of IBs.** *Circular nodes represent an off-lattice discretization of the IB contours. The regular grid behind the boundaries represents points on which a viscous Newtonian fluid, ubiquitous across the domain, is discretised. Adhesion links, specified as explicit force terms, exist between nodes within each IB as well as between neighbouring boundaries. The terms h, $\mathbf{F}_{int}$ and $\mathbf{F}_{ext}$ are defined in the main text.*

elastic boundaries exert a force on the fluid, which induces a flow that, in turn, causes the boundaries to move. In the context of interacting cell populations, each IB may be thought of as representing the membrane of an individual cell or, more generally, structures on smaller or larger scales such as intracellular detail [35] or multicellular regions of tissue [34]. Intra- and intercellular interaction terms, which represent phenomena such as cortical tension in the cell membrane and the action of adhesive transmembrane proteins, are specified as explicit forces acting between discrete locations on each IB. A schematic of parts of three neighbouring IBs is shown in Figure 2.1. The set of such interactions defines, at any given time, a resultant force acting on the membranes. This resultant force is applied to the fluid, which induces a flow. This flow carries the membranes along with it, thereby updating the positions of the boundaries. Thus, the role of the fluid is to provide a mechanism by which the boundary locations are updated; a more detailed discussion of this mechanism is presented in Section 2.2.

The IBM has several features that make it well suited to modelling the collective dynamics of cell populations. First, and most importantly, the shape of cell boundaries can be represented with theoretically arbitrary precision. This enables investigation of processes at a subcellular scale while allowing cell shapes to be an emergent property rather than a constraint of the model, in contrast to other approaches such as VMs [128, 150] and particle-based models [69]. Second, volume is preserved within any given closed contour of the fluid unless specifically altered by fluid sources or sinks. Thus, the IBM allows for the study of regulated processes

that affect cell size, such as cell growth, shrinkage, division and death. Third, implementations of the IBM typically have a small number of parameters. As shown in Section 2.3, the fluid dynamics depend only on the Reynolds number while cell mechanical interactions are usually modelled by means of simple forces such as linear springs. This opens the possibility of calibration against biological data; Rejniak, for instance, has demonstrated this by successfully parametrising an IBM implementation with numerical values estimated from various experimental studies [138]. Finally, unlike numerical schemes that employ structured or unstructured grids conforming to the immersed body, in the IBM the fluid is discretised using a regular Cartesian grid that may be generated with ease. This allows a relatively simple numerical scheme, discussed in Section 2.4.8, which has a fairly straightforward and efficient computational implementation and enables the use of a fast and direct spectral method for computing the fluid flow.

Several previous studies have detailed aspects of the IBM, including a thorough treatment of the underlying mathematics in a general three-dimensional setting by Peskin [124]. Biological applications include those by Rejniak and colleagues, who use an IBM implementation to investigate the growth of solid tumours under differing geometric configurations, initial conditions and tumour progression models [137, 138]. The same authors have investigated the mechanics of the bilayer of trophoblasts in the developing placenta [139]. Rejniak and Dillon employ a similar framework to explain the variety of different architectural forms in intraductal tumours [135]. Dillon and Othmer use an IBM to model spatial patterning of the vertebrate limb bud [34], and an IBM framework for tackling general morphogenetic problems is presented by Tanaka and colleagues [164]. Cell deformation is investigated by several authors; by Bottino in the context of passive actin cytoskeletal networks [14], by Jadhav and colleagues with a three-dimensional implementation in the context of cell rolling [73], and by Li and colleagues [86], Bagchi [6], Fedosov and colleagues [43] and Krüger and colleagues [82] who use the IBM to study deformation and flow of red blood cells. Also in three dimensions is a comprehensive model of the cochlea by Givelberg and Bunn [53], in a very different Reynolds number regime, demonstrating the versatility and range of IBMs. More recently, IBMs in three dimensions have been applied to ever more complex geometries, and studies include an investigation of the hydrodynamics of diatom chains [114], a model of actively swimming jellyfish [71], and a study of aortic heart valve dynamics [57]. These studies utilise the IBAMR implementation[1]. Finally, a review by Mittal and Iaccarino gives excellent background on the method

[1] `https://github.com/IBAMR`

and cites many other examples of its use across various, including non-biological, application areas [109].

While, collectively, these papers provide an excellent overview of the IBM and several implementations thereof, there remains no comprehensive account of the model, method of solution, computational implementation or analysis of parameter scaling. The aim of this work is therefore to provide comprehensive details of an IBM implementation aimed specifically at describing the collective dynamics of multicellular tissues. We provide a free, open source implementation of the IBM complete with example simulations: we build on the established Chaste library [107, 127] to ensure that the code is robust and well-tested; we present the code necessary to reproduce all figures in this chapter; and we conduct a thorough numerical analysis detailing how parameters scale with respect to each other in order to build a recipe allowing consistent parametrisation of models. It is worth noting that, while there are a number of IBM implementations in three dimensions, there are several reasons for which we choose to focus on a two-dimensional implementation in the present work. The computational demands of a fully three-dimensional implementation are high, and important advances continue to be made: the use of distributed-memory parallel processing to compute larger three-dimensional problems, and improvements to the IBM framework itself such as adaptive mesh refinement, have been instrumental in undertaking several of the three-dimensional studies mentioned above [59]. While this work opens up many avenues for future study, we target our framework at applications in multicellular tissues where we wish to simulate many hundreds of cells, and for which two-dimensional simulation is able to effectively capture important mechanisms. Thus, our focus is on the integration of biological processes into a two-dimensional IBM framework, in order to address specific questions in the field of epithelial morphogenesis.

The remainder of this chapter is structured as follows. Sections 2.2 to 2.4 give details of the IBM, its discretization and a numerical solution using a fast-Fourier transform algorithm. Section 2.5 outlines the C++ implementation in Chaste. Section 2.6 details a numerical analysis demonstrating that the computational implementation converges, and elaborating on how parameters scale relative to each other. Section 2.7 provides a prototype simulation study, and details specific biological questions that the IBM is well suited to explore. Section 2.8 concludes with a discussion of the choices made in our implementation, and future work in this area.

2.2 IBM formalism

Consider a viscous Newtonian fluid, with velocity $\mathbf{u} = \mathbf{u}(\mathbf{x}) = \mathbf{u}(x, y)$, flowing in a two-dimensional, doubly periodic domain $\Omega = [0, L] \times [0, L]$. The fluid motion obeys the Navier–Stokes equations

$$\rho \frac{\partial \mathbf{u}}{\partial t} + \rho \left(\mathbf{u} \cdot \nabla \right) \mathbf{u} + \nabla p - \mu \left(\nabla^2 \mathbf{u} + \frac{1}{3} \nabla s \right) - \rho \mathbf{f} = 0, \tag{2.1a}$$

$$\nabla \cdot \mathbf{u} = s, \tag{2.1b}$$

where ρ and μ are the fluid density and viscosity, respectively, and are both assumed constant; p is the pressure field; $\mathbf{f}$ is the force per unit area acting on the fluid; and s is the fluid source field, representing the proportional volume change per unit time. The periodic boundary conditions enforce $\mathbf{u}(x, 0) = \mathbf{u}(x, L)$ and $\mathbf{u}(0, y) = \mathbf{u}(L, y)$, for $0 \leq x, y \leq L$.

We next consider a set of N non-overlapping closed curves in the fluid, which we will refer to as IBs, and which we think of as representing cell membranes. We associate upper-case Roman indices (e.g. $\mathbf{F}$ and $\mathbf{X}$) with the IBs, and lower-case Roman indices (e.g. $\mathbf{f}$ and $\mathbf{x}$) with the fluid domain Ω. Let $\Gamma_1, \ldots, \Gamma_N$ denote the collection of IBs, and let each IB Γ_k be parametrised by γ_k. Further, let

$$\Gamma = \bigcup_{k=1}^{N} \Gamma_k \tag{2.2}$$

denote the union of these IBs, parametrised by γ, which is composed of $\gamma_1, \ldots, \gamma_N$ in the natural way. Let $\mathbf{X} = \mathbf{X}(\gamma_k, t)$ denote the coordinates of the k^{th} IB and let $\mathbf{X} = \mathbf{X}(\gamma, t)$ be the combined coordinates of all IBs.

We denote the resultant force acting on the IBs by $\mathbf{F} = \mathbf{F}(\gamma, t)$. The precise functional form of the resultant force $\mathbf{F}$ varies with application, and is formulated in Section 2.4. We relate the resultant force on the IBs to the body force acting on the fluid through the relation

$$\mathbf{f}(\mathbf{x}, t) = \int_{\Gamma} \mathbf{F}(\gamma, t)\, \delta(\mathbf{x} - \mathbf{X}(\gamma, t))\, \mathrm{d}\gamma = \sum_{k=1}^{N} \int_{\Gamma_k} \mathbf{F}(\gamma_k, t)\, \delta(\mathbf{x} - \mathbf{X}(\gamma_k, t))\, \mathrm{d}\gamma_k, \tag{2.3}$$

where $\delta(\cdot)$ denotes the Dirac delta function. The force on the fluid at location $\mathbf{x}$ thus vanishes away from the IBs, and equals the resultant force $\mathbf{F}$ at location $\mathbf{X}$ on an IB precisely at $\mathbf{x} = \mathbf{X}$.

The IBs are assumed to move due to the fluid flow without slipping, so that a point along Γ moves at precisely the local fluid velocity:

$$\frac{\partial \mathbf{X}(\gamma, t)}{\partial t} = \mathbf{u}\left(\mathbf{X}(\gamma, t)\right) = \int_{\Omega} \mathbf{u}(\mathbf{x}, t)\, \delta(\mathbf{x} - \mathbf{X}(\gamma, t))\, \mathrm{d}\mathbf{x}. \tag{2.4}$$

Thus, the velocity of an arbitrary IB point $\mathbf{X}(\gamma)$ is equal to the velocity of the fluid at $\mathbf{x} = \mathbf{X}$.

The source field, s, is considered to be a finite linear combination of individual point sources. The number, location and strength of each source is formulated in Section 2.4, but for now we consider s as an arbitrary (but known) scalar field.

2.3 Non-dimensionalization

We non-dimensionalize the model to reduce the number of parameters and allow us to estimate the relative importance of each term. For the Navier–Stokes equations, we introduce the standard choices for viscous dynamics: a length scale, L; velocity scale, U; time scale, L/U; pressure scale, $U\mu/L$; source scale, U/L; and force scale, U^2/L. Substituting the rescaled variables and operator

$$\mathbf{x} = L\overset{*}{\mathbf{x}},\quad \mathbf{u} = U\overset{*}{\mathbf{u}},\quad t = \frac{L}{U}\overset{*}{t},\quad p = \frac{U\mu}{L}\overset{*}{p},\quad s = \frac{U}{L}\overset{*}{s},\quad \mathbf{f} = \frac{U^2}{L}\overset{*}{\mathbf{f}},\quad \nabla = \frac{1}{L}\overset{*}{\nabla}, \tag{2.5}$$

into Equations (2.1a) and (2.1b) and dropping the stars yields

$$\frac{\partial \mathbf{u}}{\partial t} + (\mathbf{u}\cdot\nabla)\,\mathbf{u} + \frac{1}{Re}\left(\nabla p - \nabla^2\mathbf{u} - \frac{1}{3}\nabla s\right) - \mathbf{f} = 0, \tag{2.6a}$$

$$\nabla\cdot\mathbf{u} = s, \tag{2.6b}$$

where $Re = \rho LU/\mu$, the Reynolds number, represents the ratio of inertial to viscous forces. At very low Reynolds number it is appropriate to take the limit $Re \to 0$ in Equation (2.6a) and, assuming the body force, $\mathbf{f}$, to be of order $1/Re$, obtain the Stokes equations

$$\nabla^2\mathbf{u} - \nabla p + \frac{1}{3}\nabla s + \mathbf{f} = 0, \tag{2.7a}$$

$$\boldsymbol{\nabla}\cdot\mathbf{u} = s. \tag{2.7b}$$

Note that we assume $\mathbf{f} \sim \mathcal{O}(1/Re)$ as otherwise no flow would be induced by the force on the IBs.

Small scale systems typically exhibit low velocities, and thus Reynolds numbers for biological regimes can be very small. Small swimming organisms, for instance, may experience Reynolds numbers as low as $Re \approx 10^{-4}$ [129]. Tanaka and colleagues [163] estimate Reynolds numbers for the fluid-like properties of embryonic tissues as $Re \approx 10^{-13}$ using assumptions of $L = 10^{-3}[m]$, $U = 10^{-8}[ms^{-1}]$ and $\mu/\rho = 10^{2}[m^2 s^{-1}]$. Rejniak and colleagues [139] arrive at $Re \approx 10^{-9}$ by considering the length scale to be the size of a large cytotrophoblastic cell (20μm) and a characteristic velocity of $30\mu m$ in 24 hours.

The Stokes equations (2.7a) and (2.7b) are computationally less expensive than the full Navier–Stokes equations (2.6a) and (2.6b) to solve; for example, their linearity permits the use of efficient Green's function methods [24]. This raises the question of the circumstances under which it is appropriate to assume Stokes flow for the IB fluid component, as described in [17, 88]. Here, we choose to solve the full Navier–Stokes equations, the reasons for which are discussed in Section 2.8, while keeping in mind that there are particular simulations for which the reduced problem may be suitable and computationally less expensive to solve.

Having chosen to solve the non-dimensional Navier–Stokes equations (2.6a) and (2.6b), we non-dimensionalize Equations (2.3) and (2.4) using the rescaled parameters

$$\mathbf{X} = L \overset{*}{\mathbf{X}}, \quad \mathbf{F} = \frac{U^2}{L} \overset{*}{\mathbf{F}}, \quad \gamma = L \overset{*}{\gamma}, \tag{2.8}$$

dropping the stars, as before.

2.4 Discretization

We solve the coupled problem, consisting of Equations (2.3), (2.4), (2.6a) and (2.6b), numerically, as follows. The IBs are discretised into a finite union of points (small circles in Figure 2.1) that we call *nodes*. The fluid domain Ω is discretised onto a regular Cartesian grid (square lattice in Figure 2.1) that we refer to as the *mesh*. In our non-dimensional coordinates, $\Omega = [0, 1] \times [0, 1]$ is discretised with $N \times N$ grid points with mesh spacing h. We must also discretize Equation (2.3) relating the force $\mathbf{F}$ on the IBs with the body force $\mathbf{f}$ acting on the fluid, and Equation (2.4) relating the fluid and node velocities.

In the following, time is discretised in steps of Δt, and we refer to an arbitrary function $\mathbf{\Phi}(\cdot, t)$ at the n^{th} time step by $\mathbf{\Phi}(\cdot, n\Delta t) = \mathbf{\Phi}^n(\cdot)$.

2.4.1 Discrete Dirac delta function

In the discretised system, the fluid and IBs interact only via a discrete version of the Dirac delta function. To approximate the Dirac delta function on the discrete mesh, we require a function with finite support for which, when interpolating between the IB and fluid domains, the contributions at each fluid mesh point in the support sum to unity. Various such functions have been proposed, of which four examples from different IBM implementations are detailed by Mittal and Iaccarino [109].

Here, we make the common choice of a trigonometric function, used in several other IBM implementations [34, 138, 139], given by

$$\delta_h(\mathbf{d}) = \frac{1}{h^2}\phi\left(\frac{\mathbf{d}_x}{h}\right)\phi\left(\frac{\mathbf{d}_y}{h}\right), \tag{2.9}$$

where h is the mesh spacing, $\mathbf{d} = (\mathbf{d}_x, \mathbf{d}_y)$ is an arbitrary distance from the origin, and the function ϕ is given by

$$\phi(r) = \begin{cases} \frac{1}{4}\left(1 + \cos\left(\frac{\pi r}{2}\right)\right), & |r| \leq 2, \\ 0, & \text{otherwise.} \end{cases} \tag{2.10}$$

This choice of ϕ differs from, but takes extremely similar numerical values to, that derived and used by Peskin [124]. Given the numerical similarity, the choice of function is unlikely to make much practical difference, and we have found the version presented here to be less computationally expensive to compute (see Section 2.8).

We also note that, due to the bounded support of both functions, $\delta_h(\mathbf{d})$ will only ever be non-zero at the 4×4 mesh points closest to any given node. The choice of support size is discussed by Peskin [124], and is made purely on computational grounds: one could choose a delta function approximation with wider support, but each node on an IB would then interact with many more mesh points, slowing down the computation.

2.4.2 Discretization of IBs

We discretize each IB Γ_k by a set of N_k nodes whose positions are given by $\mathbf{\Gamma}_k^1, \ldots, \mathbf{\Gamma}_k^{N_k}$. We suppose that these nodes are initially spaced equally along the original parameter range $\gamma_k = (0, \gamma_k^{max})$, so that the length element $\Delta\gamma_k$ associated with the k^{th} IB is equal to the initial node spacing, γ_k^{max}/N_k. Because we impose the condition that each IB forms a closed contour we have $\mathbf{\Gamma}_k^{N_k+1} = \mathbf{\Gamma}_k^1$.

2.4.3 Discrete force relations

We are now in a position to define the resultant force $\mathbf{F}$ acting on the IBs. The discretization treats $\mathbf{F}$ as the union of a finite set of point forces given by the resultant force on each node in each IB.

We will consider the resultant force on each node as the combination of two types of force: 'internal' forces, that depend on the positions of other nodes in the same IB; and 'external' forces, that depend on the positions of nodes in different IBs. Here, we introduce specific choices for the force terms to represent both the mechanical properties of the actomyosin cortex of a cell and the protein–protein interactions between neighbouring cells. We represent both these mechanical interactions by linear springs, following previous IBM implementations [34, 137, 138, 139, 164], although different functional forms could, in principle, be chosen.

Internal forces represent the contractile properties of a eukaryotic cell's actomyosin cortex, which we describe by connecting each node to its neighbours by a linear spring of stiffness κ_{int} and rest length l_{int}. The internal force acting on node $\mathbf{\Gamma}_k^p$ is thus given by

$$\mathbf{F}_{int}\left(\mathbf{\Gamma}_k^p, t\right) = \kappa_{int} \sum_{j=p\pm1 \bmod N_k} \frac{\mathbf{\Gamma}_k^j - \mathbf{\Gamma}_k^p}{||\mathbf{\Gamma}_k^j - \mathbf{\Gamma}_k^p||} \left(||\mathbf{\Gamma}_k^j - \mathbf{\Gamma}_k^p|| - l_{int}\right). \tag{2.11}$$

External forces represent the adhesive properties of transmembrane proteins, such as integrins and cadherins, linking neighbouring cells. We assume that any node in an IB is connected to all nodes in different IBs that are situated within a distance d_{ext} by a linear spring with stiffness κ_{ext} and rest length l_{ext}. The external force acting on the node $\mathbf{\Gamma}_k^p$ is given by

$$\mathbf{F}_{ext}\left(\mathbf{\Gamma}_k^p, t\right) = \kappa_{ext} \sum_{\substack{q=1 \\ q\neq k}}^{N} \sum_{j=1}^{N_q} H\left(d_{ext} - ||\mathbf{\Gamma}_q^j - \mathbf{\Gamma}_k^p||\right) \frac{\mathbf{\Gamma}_q^j - \mathbf{\Gamma}_k^p}{||\mathbf{\Gamma}_q^j - \mathbf{\Gamma}_k^p||} \left(||\mathbf{\Gamma}_q^j - \mathbf{\Gamma}_k^p|| - l_{ext}\right), \tag{2.12}$$

where the outer sum runs over all other IBs, the inner sum runs over the N_q nodes in boundary q, and $H(\cdot)$ is the Heaviside step function. Our choice of linear spring interactions is motivated primarily by their ease of implementation and low computational overhead (see Section 2.8), although in our software implementation the user is free to define their own functional forms.

The total force $\mathbf{F}$ on a node is given by the sum of the internal and external forces,

$$\mathbf{F}\left(\mathbf{\Gamma}_k^p, t\right) = \mathbf{F}_{int}\left(\mathbf{\Gamma}_k^p, t\right) + \mathbf{F}_{ext}\left(\mathbf{\Gamma}_k^p, t\right). \tag{2.13}$$

2.4.4 Discretization of the Navier–Stokes equations

Due to the periodicity of the spatial domain, we employ a fast-Fourier transform algorithm to solve Equations (2.6a) and (2.6b) numerically. We use the following numerical scheme, used by Peskin and McQueen [125] and later, with the addition of fluid sources, by Dillon and Othmer [34]. The numerical scheme is due to Chorin [21, 22] who demonstrated applicability of the following temporal discretisation, where the sums are taken over the two dimensions, $d \in \{1, 2\}$:

$$\frac{\mathbf{u}^{n+1} - \mathbf{u}^n}{\Delta t} + \sum_d u_d^n D_d^{\pm} \mathbf{u}^n + \frac{1}{Re}\left(\mathbf{D}^0 p^{n+1} - \sum_d D_d^+ D_d^- \mathbf{u}^{n+1} - \frac{1}{3}\mathbf{D}^0 s^n\right) - \mathbf{f}^n = 0, \tag{2.14}$$

$$\mathbf{D}^0 \cdot \mathbf{u}^{n+1} = s^n, \tag{2.15}$$

where the forward and backward divided difference operators, D_d^+ and D_d^-, the vector of central divided difference operators, $\mathbf{D}^0$, and the upwind divided difference operator, $D_{dd}^{\pm}$, are defined by

$$\left(D_d^+ \phi\right)(\mathbf{x}) = \frac{\phi(\mathbf{x} + h\mathbf{e}_d) - \phi(\mathbf{x})}{h}, \tag{2.16}$$

$$\left(D_d^- \phi\right)(\mathbf{x}) = \frac{\phi(\mathbf{x}) - \phi(\mathbf{x} - h\mathbf{e}_d)}{h}, \tag{2.17}$$

$$\left(D_d^0 \phi\right)(\mathbf{x}) = \frac{\phi(\mathbf{x} + h\mathbf{e}_d) - \phi(\mathbf{x} - h\mathbf{e}_d)}{2h}, \tag{2.18}$$

$$\mathbf{D}^0 = \left(D_x^0, D_y^0\right), \tag{2.19}$$

$$u_d^n D_{dd}^{\pm} = \begin{cases} u_d^n D_d^-, & \text{if} \quad u_d^n > 0, \\ u_d^n D_d^+, & \text{if} \quad u_d^n < 0, \end{cases} \tag{2.20}$$

respectively. Here, $\mathbf{e}_d$ denotes the unit vector in the d^{th} dimension.

2.4.5 Discretization of force relation

We discretize Equation (2.3), relating the force on the fluid to the force on the IBs, as follows. For each point $\mathbf{x}$ in the fluid mesh, we sum the force contributions from every IB node using the discrete delta function to assign the appropriate weight,

$$\mathbf{f}^n(\mathbf{x}) = \sum_{k=1}^{N} \left(\sum_{j=1}^{N_k} \mathbf{F}^n(\mathbf{\Gamma}_k^j)\ \delta_h(\mathbf{x} - \mathbf{\Gamma}_k^j)\ \Delta\gamma_k \right), \tag{2.21}$$

where the outer sum runs over the N IBs, the inner sum runs over the N_k nodes in the k^{th} IB, and $\Delta\gamma_k$ is the length element associated with the k^{th} IB.

2.4.6 Discretization of position-updating relation

For simplicity, we discretize Equation (2.4) using a forward Euler scheme. At the n^{th} time step, a given IB node $\mathbf{\Gamma}_k^j$ is displaced by $\Delta t\ \mathbf{u}^n(\mathbf{\Gamma}_k^j)$. Because, in general, $\mathbf{\Gamma}_k^j$ will not coincide with a fluid mesh point, the value $\mathbf{u}^n(\mathbf{\Gamma}_k^j)$ is an interpolation of the 4×4 fluid mesh points closest to $\mathbf{\Gamma}_k^j$. The discretised relation for updating node locations is therefore given by

$$\left(\mathbf{\Gamma}_k^j\right)^{n+1} = \left(\mathbf{\Gamma}_k^j\right)^n + \Delta t \sum_{\mathbf{x}\in\mathcal{N}\left(\mathbf{\Gamma}_k^j\right)} \mathbf{u}^{n+1}(\mathbf{x})\ \delta_h(\mathbf{x} - \mathbf{\Gamma}_k^j)\ h^2, \tag{2.22}$$

where $\mathcal{N}\left(\mathbf{\Gamma}_k^j\right)$ represents the 4×4 fluid mesh points nearest $\mathbf{\Gamma}_k^j$ (the only points with non-zero contributions, due to the implementation of δ_h).

2.4.7 Discretization of fluid sources

The source term, s, allows individual regions enclosed by contours in the fluid domain to increase or decrease in volume. In the absence of s, due to the volume conservation property of the IBM, the quantity of fluid within any given closed contour remains fixed. In the context of simulating multicellular biological systems, the source term, s, allows the modulation of cell size.

To allow the fluid volume within each IB to be modulated, we decompose s into a finite number of point sources and initially put a single source at the centroid of each IB. To ensure a constant total volume of fluid in the domain Ω, we additionally include a number of sinks (sources with a negative strength) located outside all IBs which balance the magnitude of the N sources associated with the boundaries.

Suppose there are M combined sources and sinks, $s_1, \ldots, s_M$, with $M > N$, located at the positions $\mathbf{s}_1, \ldots, \mathbf{s}_M$. Each source s_k has specified strength T_k, where $\sum_{k=1}^{M} T_k = 0$, and the source field $s(\mathbf{x})$ at an arbitrary fluid mesh point $\mathbf{x}$ then satisfies

$$s(\mathbf{x}) = \sum_{k=1}^{M} T_k \delta_h(\mathbf{x} - \mathbf{s}_k). \tag{2.23}$$

A convenient method to ensure that fluid sources always remain inside (or outside) IBs entails updating their locations in the same way as for the IB nodes,

$$(\mathbf{s}_k)^{n+1} = (\mathbf{s}_k)^n + \Delta t \sum_{\mathbf{x}\in\mathcal{N}(\mathbf{s}_k)} \mathbf{u}^{n+1}(\mathbf{x})\ \delta_h(\mathbf{x}-\mathbf{s}_k)\ h^2, \tag{2.24}$$

where $\mathcal{N}(\mathbf{s}_k)$ represents the 4×4 fluid mesh points nearest $\mathbf{s}_k$.

The regulation of source strengths depends on the application and on the biological process underlying the cell size change, and may, for example, be linked to a description of cell cycle progression and growth. Some examples of biological processes and their feedback on source strengths are discussed in Section 2.5.2. Note also that the number of extra 'balancing sources' is not fixed, and this choice is discussed in Section 2.8.

2.4.8 Numerical solution

We are now in a position to solve Equations (2.6a) and (2.6b) numerically. We reiterate that this numerical scheme is due to Peskin and McQueen [125] with updates by Dillon and Othmer [34]. Equation (2.21) allows the direct computation of $\mathbf{f}^n$, but Equation (2.22) requires $\mathbf{u}^{n+1}$, which we must compute, given $\mathbf{f}^n$, from Equations (2.14) and (2.15).

Rearranging Equation (2.14) to separate the terms evaluated at different time steps yields

$$\left(I - \frac{\Delta t}{Re}\sum_d D_d^+ D_d^-\right)\mathbf{u}^{n+1} + \frac{\Delta t}{Re}\,\mathbf{D}^0 p^{n+1} = \mathbf{R}^n, \tag{2.25}$$

where

$$\mathbf{R}^n = \left(I - \Delta t \sum_d u_d^n D_{dd}^{\pm}\right)\mathbf{u}^n + \frac{\Delta t}{3Re}\mathbf{D}^0 s^n + \Delta t \mathbf{F}^n, \tag{2.26}$$

and I is the 2×2 identity matrix.

We solve Equations (2.15) and (2.25) directly for $\mathbf{u}^{n+1}$ by applying a discrete Fourier transform (DFT) to eliminate p^{n+1}. For our domain $\Omega = [0,1] \times [0,1]$ discretised using an $N \times N$ square mesh of spacing h, we define the DFT from the spatial coordinates $(\cdot)_{x,y}$ to the spectral coordinates $\hat{(\cdot)}_{k_1,k_2}$ by

$$\hat{(\cdot)}_{k_1,k_2} = \sum_{y=0}^{N-1}\sum_{x=0}^{N-1}(\cdot)_{x,y}\ \exp\left(-\frac{2\pi i}{N}\left(xk_1 + yk_2\right)\right). \tag{2.27}$$

Under this transformation, Equations (2.15) and (2.25) become

$$\left(1+\frac{4\Delta t}{h^2 Re}\sum_d \sin^2\left(\frac{\pi k_d}{N}\right)\right)(\hat{u}_d)^{n+1}_{k_1,k_2}+\frac{i\Delta t}{hRe}\sin\left(\frac{2\pi k_d}{N}\right)\hat{p}^{n+1}_{k_1,k_2}=\left(\hat{R}_d\right)^n_{k_1,k_2}, \quad (2.28)$$

$$\frac{i}{h}\sum_d \sin\left(\frac{2\pi k_d}{N}\right)(\hat{u}_d)^{n+1}_{k_1,k_2}=(\hat{s})^n_{k_1,k_2}, \quad (2.29)$$

where i is the imaginary unit, sums are taken over dimension, $d \in \{1,2\}$, and each equation is now of a single variable so holds for $d = 1, 2$.

We substitute Equation (2.29) into Equation (2.28) to solve directly for p: multiplying Equation (2.28) by $(i/h)\sin(2\pi k_d/N)$, summing it over the two dimensions, and rearranging for $\hat{p}$ gives

$$\hat{p}^{n+1}_{k_1,k_2}=\frac{\left(1+\frac{4\Delta t}{h^2Re}\sum_d \sin^2\left(\frac{\pi k_d}{N}\right)\right)(\hat{s})^n_{k_1,k_2}-\frac{i}{h}\sum_d \sin\left(\frac{2\pi k_d}{N}\right)\left(\hat{R}_d\right)^n_{k_1,k_2}}{\frac{\Delta t}{h^2Re}\sum_d \sin^2\left(\frac{2\pi k_d}{N}\right)}, \quad (2.30)$$

where every term on the right-hand side depends only on information available at the current time step. We can therefore substitute Equation (2.30) back into Equation (2.28) to solve for $\hat{\mathbf{u}}^{n+1}_{k_1,k_2}$, obtaining

$$(\hat{u}_d)^{n+1}_{k_1,k_2}=\frac{\left(\hat{R}_d\right)^n_{k_1,k_2}-\frac{i\Delta t}{hRe}\sin\left(\frac{2\pi k_d}{N}\right)\hat{p}^{n+1}_{k_1,k_2}}{1+\frac{4\Delta t}{h^2Re}\sum_d \sin^2\left(\frac{\pi k_d}{N}\right)}. \quad (2.31)$$

Care must be taken at the mesh points $(k_1,k_2) = (0,0),(0,N/2),(N/2,0)$ and $(N/2,N/2)$, where the denominator of the right-hand side of Equation (2.30) vanishes. At these points, however, the sine term multiplying $\hat{p}^{n+1}_{k_1,k_2}$ in Equation (2.28) also vanishes, and we may thus solve directly for $(\hat{u}_d)^{n+1}_{k_1,k_2}$. We therefore avoid this problem by setting $\hat{p}^{n+1}_{k_1,k_2}=0$ in Equation (2.31). Finally, having computed $(\hat{u}_d)^{n+1}_{k_1,k_2}$, we apply the inverse DFT to obtain $\mathbf{u}^{n+1}$,

$$(u_d)^{n+1}_{x,y}=\frac{1}{N^2}\sum_{k_2=0}^{N-1}\sum_{k_1=0}^{N-1}(\hat{u}_d)^{n+1}_{k_1,k_2}\ \exp\left(\frac{2\pi i}{N}\left(xk_1+yk_2\right)\right). \quad (2.32)$$

2.5 Computational implementation

In this section, we describe the time-stepping algorithm for solving the IBM and how it fits into the computational modelling framework Chaste. We go on to highlight some of the computational challenges addressed during the implementation of this

method, and we present some benchmarking and profiling results. Finally, we detail how rule-based processes such as cell division, needed for simulating populations of cells, are handled within this IBM implementation.

2.5.1 Chaste

We have implemented our IBM framework as part of the Chaste C++ library [107, 127]. The IBM code is released as a feature branch of the latest development version of Chaste[2], which is open source and available under the 3-clause BSD licence. Chaste is developed with a test-driven approach using the unit testing framework CxxTest[3]. Using this framework, unit tests verify the correctness of every individual method within the implementation, and simulations are themselves written as test suites. Details of how to obtain the IBM implementation, as well as code to recreate all simulations from this chapter, can be found in Appendix A.1.

As it is written in C++, Chaste is fast and able to utilise object orientation and class inheritance, enabling modularity and easy extensibility of the code base. This structure enables the IBM to integrate with Chaste as a new example of the pre-existing class of 'off-lattice simulations', within which much of the core functionality such as simulation setup, time stepping and cell cycle models are already implemented and thoroughly tested. In addition, new specialised functionality is built upon existing abstract classes, meaning a consistent and familiar interface is presented to existing code users.

Using the numerical method described in Section 2.4, we solve the IBM by iterating through the following fixed time-stepping algorithm:

1. update the cell population to take account of cellular processes including cell death, division, growth, shrinkage and procession through the cell cycle, discussed shortly;

2. calculate the internal and external forces acting on each node, using Equation (2.13);

3. loop over each IB node and propagate the associated force to the fluid mesh domain, as described by Equation (2.21);

4. loop over each fluid source and propagate the associated strength to the fluid mesh domain, as described by Equation (2.23);

[2] `http://www.cs.ox.ac.uk/chaste/download.html`
[3] `http://cxxtest.com/`

5. solve the Navier–Stokes equations, Equations (2.6a) and (2.6b), using the fast-Fourier transform algorithm detailed in Section 2.4.8 to generate new fluid velocities;

6. use the new fluid velocities to update IB node and fluid source locations as described by Equations (2.22) and (2.24).

The computational complexity of this timestepping algorithm is described in Section 2.5.3. An example of a simple simulation performed using this implementation within Chaste is visualised in Figure 2.2, where an elliptical IB relaxes over time towards a circular shape. The fluid flow is shown as a vector field of arrows.

2.5.2 Implementation of cellular processes

Sections 2.2 to 2.4 detail our IBM and a numerical solution thereof, and together these constitute a method of solving fluid–structure interactions. In addition to this, we need the facility to include various cellular processes that occur when modelling a multicellular tissue. Such processes include regulated cell growth, division and death, and can be thought of as a collection of rules by which the properties of the IBs or fluid sources are altered, but which do not directly alter the underlying fluid problem.

An example of rule-based modification of IBs is cell division. Within Chaste, we make use of existing functionality for encoding cell cycle progression. In this framework, a cell may at some time step be deemed 'ready to divide', at which time the following scheme is employed to replace the single IB (representing the cell about to divide) with two IBs (representing the daughter cells). First, a division axis through the centroid of the IB is selected, by means of some rule chosen by the user. This rule may, for instance, select the shortest axis of the IB, or a random axis, depending on the biological assumptions particular to the scenario being modelled. Second, with the division axis fixed, the boundary is divided in two: nodes on each side of the axis define the shape of each daughter cell, with the daughter cells separated by a pre-determined fixed distance. This fixed distance is selected to mimic the average distance between cells in the simulation, however it must be large compared to the fluid mesh spacing h to avoid the problem of 'locking' where IBs cannot move relative to one another due to insufficient precision in the fluid discretisation; this problem, as well as guidance on a suitable distance, is discussed at length in Section 4.3.7. We make the choice that each daughter cell is represented by the same number of nodes as the parent cell, and a re-meshing process instantaneously spaces these nodes evenly around the outline of each daughter cell.

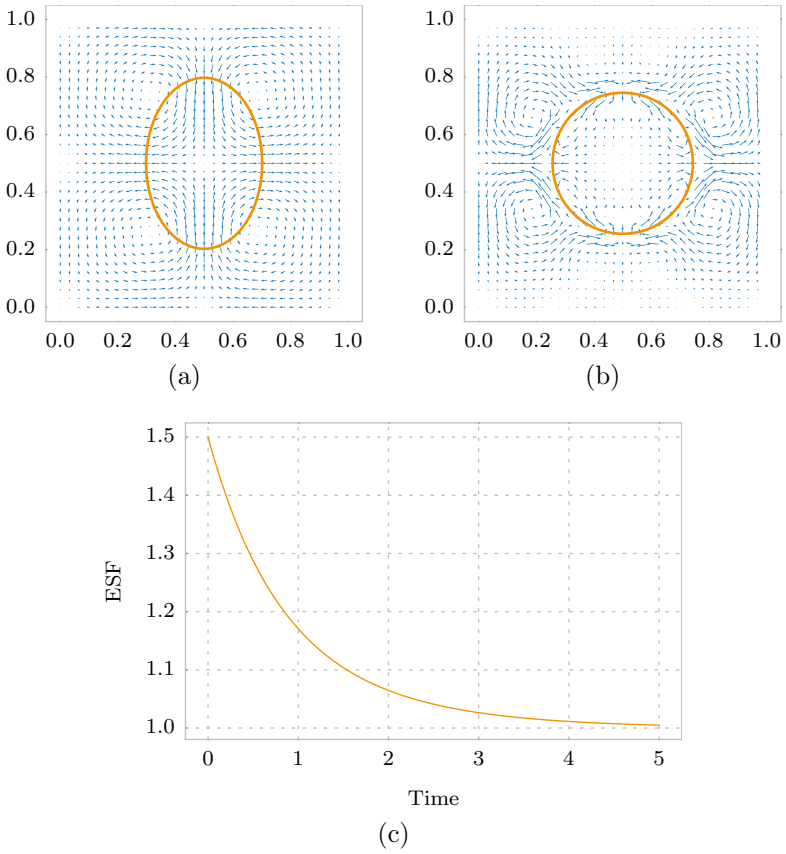


Figure 2.2: **An example IBM simulation.** *An elliptical IB relaxing over time under the action of internal forces (Section 2.4.3), and no fluid sources. In this simulation,* $h = 1/32$, $\Delta t = 0.05$ *and* $N = 128$ *nodes. Full details of all parameters can be found in the simulation code, available as part of the test suite 'TestNumericsPaperSimulations' (see Appendix A.1). (a) State of the simulation after one time step, where flow is acting to reduce the elliptical IB in height and expand it in width. (b) State of the simulation after* 100 *time steps, where flow vanishes at the boundary. (c) Dynamics of the aspect ratio of the ellipse, quantified by its elongation shape factor (ESF; see Section 2.6 for details), over time.*

We remark that this scheme defines a rule-based implementation of cell division as a process occurring during a single time step. Depending on the time scale over which the tissue is modelled, one may wish to explicitly represent pinching during cytokinesis, as implemented by Rejniak and colleagues [137, 139]. This can be achieved within Chaste, using existing functionality that allows feedback between the cell cycle and arbitrary cell properties such as a 'target' surface area that cells seek to attain. In this manner, when a cell is selected to divide, processes such as an increase in size followed by the formation of a contractile furrow could be specified (for instance, via a feedback with fluid source strengths); however, we stress that our

implementation is left flexible and extendible. The modular and hierarchical nature of Chaste allows the user to easily specify appropriate cell cycles, division rules and cell property modifiers for a given biological scenario.

Modelling such biological processes does pose certain problems which must be carefully accounted for. First, as a cell grows by means of an active fluid source, the length of the associated IB increases. As a result, the boundary nodes become spaced further apart, increasing the energy stored in the membrane and resulting in a cell cortex that is more resistant to deformation. A method to mitigate these issues requires adaptive insertion and re-meshing of the nodes along the boundary so as to keep the elastic properties constant. Currently within Chaste we have elected not to re-mesh, but this necessitates careful choice of h with respect to the largest node spacings that might occur, so as to ensure volume conservation. This, thus, necessitates the selection of a sufficiently refined node spacing, and this is described in more detail in Section 2.6.1. Second, as boundaries move around and change size, changes in connectivity are necessitated between neighbouring cells. Within Chaste, all connections between neighbouring cells are recalculated every time step based on a list of all pairwise nodes within the threshold distance d_{ext}. To prevent this recalculation being prohibitively costly, an efficient spatial decomposition algorithm is employed, as described in Section 2.5.3. Finally, with the active motion of, and interaction between, IBs, the node spacings within a single boundary inevitably fluctuate. To cope with this, Chaste implements a static re-meshing algorithm to redistribute the existing nodes around a given boundary in a volume-preserving manner, when required.

2.5.3 Computational efficiency and profiling

The two most computationally demanding components per timestep in our IBM implementation are solving the Navier–Stokes equations and calculating the forces acting on the IB nodes. The former is demanding due to the calculations necessary in the finite difference scheme, the five two-dimensional DFTs per time step, and the term-by-term calculation of the pressure field, of which the DFTs dominate and is thus $\mathcal{O}(N^2 \log(N))$ for a fluid grid with $N \times N$ points. The latter is costly due to the potentially large number of pairwise interactions between nodes on neighbouring IBs that must be kept track of, which is $\mathcal{O}(k^2)$, where k is bounded above by the total number of IB nodes, as discussed shortly. The complexity of the timestepping algorithm depends, therefore, on the balance between the fineness of the fluid-mesh and that of the IB discretisation.

	$h = 1/512$	$h = 1/1024$	$h = 1/2048$
Approximate memory footprint (MiB)	39	102	355
Time to advance 2000 time steps (s)	73.9	211	817
Time solving the fluid problem (%)	40.7	62.7	77.3

Table 2.1: **Code profiling.** *The memory footprint, time to complete 2000 time steps, and the proportion of time spent solving the Navier–Stokes equations is presented for each of three increasingly fine simulation representations. Each simulation comprises a regular hexagonal lattice of 20 IBs, allowed to relax for the fixed number of time steps. Each boundary has 300, 600 and 1200 nodes in separate simulations with 512, 1024 and 2048 fluid mesh points, respectively. Profiling was performed on a desktop machine with an Intel Xeon E5-1650 v3 CPU and 16GiB RAM, using the GNU gprof profiler. For details of how to obtain the code for these profiling simulations, see Appendix A.1.*

To reduce the time spent dynamically allocating memory used in solution of the Navier–Stokes equations, we ensure that all arrays storing values needed during the computation are created during simulation set up, and remain in place in memory throughout the simulation. For $N \times N$ fluid mesh points, this means permanently storing $12N^2$ double-precision numbers. The result of this is a drastic speed-up compared to dynamically allocating memory, with the drawback of a large memory footprint. In practical terms, this scheme puts an upper bound of $N \approx 4096$ when running a single simulation on a desktop computer, which is not prohibitive.

To optimise the second problem of efficiently calculating pairwise interactions between nearby IB nodes, we employ a spatial decomposition algorithm [64]. The domain is broken into squares each with side length equal to the interaction distance, d_{ext}, and at each time step the nodes are placed into their corresponding square. For a given node, the only possible set of interactions are then between nodes in the same or neighbouring squares. Thus, we dramatically reduce the computation necessary when $d_{ext} \ll 1$.

Table 2.1 shows various profiling statistics for a prototype simulation of 20 cells initially arranged in an hexagonal packing. The columns of Table 2.1 each represent a successive doubling of the resolution of both the fluid mesh and the number of IB nodes. As can be seen, solution of the fluid problem scales less well than calculation of the forces; however, neither component individually dominates the simulation runtime.

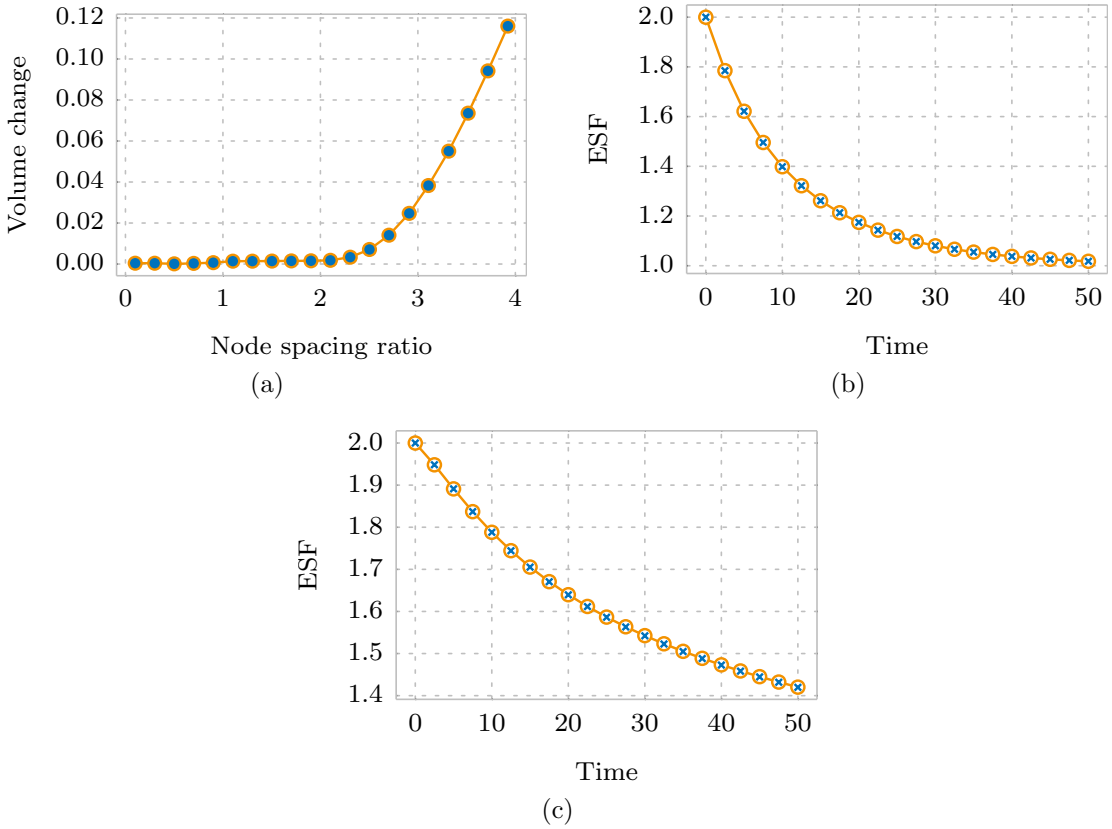


Figure 2.3: **Scaling properties in the IBM framework. (a) Node spacing ratio and volume change.** *A set of simulations of a single circular IB, each run for the same fixed simulation time. Across the set of simulations the node spacing ratio, $\Delta\gamma_k/h$, is varied and the proportional volume change of the IB is recorded. As the node spacing ratio increases beyond 2.0 there is a sharp increase in the proportional volume change, as a result of fluid escaping between the distantly spaced nodes.* **(b) Scaling intracellular spring properties with node spacing.** *Two simulations, each of an ellipse relaxing towards a circle, are run with the ESF sampled at 21 time points. Circles represent a simulation in which the IB is represented by $N = 256$ nodes, with intracellular spring constant $\kappa_{int} = \bar{\kappa}$. Crosses, coinciding with the circles, represent a simulation with a modified representation of $N = 512$ nodes and intracellular spring constant $\kappa_{int} = 4\bar{\kappa}$.* **(c) Scaling intercellular spring properties with node spacing.** *Two simulations, each of two neighbouring ellipses relaxing, are run in which the ESF of one ellipse sampled at 21 time points. Circles represent a simulation in which the IB is represented by $N = 256$ nodes, with intercellular spring constant $\kappa_{ext} = \bar{\kappa}$. Crosses, coinciding with the circles, represent a simulation with a modified representation of $N = 512$ nodes and intercellular spring constant $\kappa_{ext} = 0.5\bar{\kappa}$, and with the intracellular spring properties scaled as in (b). For details on how to obtain the code for these simulations, which contains full details of all parameter values used, see Appendix A.1.*

2.6 Numerical results

In this section, we run a number of simulations to demonstrate various properties of our IBM implementation. We first highlight an important relationship between the

IB node spacing, $\Delta\gamma_k$, and the fluid mesh spacing, h. We go on to explore how certain parameters in the IBM scale with each other, and use this to work towards a recipe by which a model of a particular biological process may be simulated. Finally, we demonstrate that the implementation converges in time step, in fluid mesh spacing, and in IB node spacing. We employ a summary statistic for an individual cell in a simulation, referred to as the elongation shape factor (ESF). For a polygon this is a dimensionless positive real number that defines a measure similar to aspect ratio. Formally, it is defined as $\sqrt{i_2/i_1}$, where $i_1 < i_2$ are the eigenvalues of the matrix of second moments of area of the polygon around its principal axes [41]. The ESF for a circle is 1, and for an ellipse it is the ratio of major to minor axis length.

2.6.1 Node spacing ratio and volume change

In the continuous IBM, IBs are carried at precisely the local fluid velocity (Equation (2.4)) because they are impermeable to fluid, in the sense that any given fluid particle will remain either inside or outside a particular IB for all time. In the discretised IBM, however, there is a gap of average length $\Delta\gamma_k$ between any two adjacent nodes in boundary k. If this gap is much larger than the fluid mesh spacing h, fluid flow between the nodes will have no impact on the propagation of node locations, and this discretisation error invalidates the volume conservation property of the IBM. In general, lack of volume conservation is a known problem with IBM numerical methods themselves [124]. While various improvements to the numerical method can be made [58, 126], these are often technically complex and bring with them a computational cost, and it is worth noting that these errors go to zero with the mesh spacing h and $\Delta\gamma_k$ in any case [124].

Therefore, to ensure conservation of fluid volume within each IB, $\Delta\gamma_k$ must be small enough in relation to h, and h must also be small; the trade-off of making these parameters overly small is simply computational expense.

To determine how small is small enough, Figure 2.3a shows the results of a set of simulations relating the change in volume of a circular IB to the node spacing ratio, $\Delta\gamma_k/h$. In each simulation, a circular cell is simulated for a fixed number of time steps. The intracellular spring properties are set with $l_{int} < \Delta\gamma_k$ to ensure the linear springs are under tension and will, in the absence of the volume conservation property of the IBM, contract to reduce the perimeter of the IB. For each simulation we measure the proportional area change of the cell (the absolute change in area of the polygon divided by the original area), for a particular initial value of the node spacing ratio. From Figure 2.3a, we see that a node spacing ratio much above 2.0

results in poor volume conservation. A node spacing ratio below 1.0, though, seems to ensure that the numerical scheme matches the continuum limit well, from which we also conclude that h is sufficiently small to mitigate the issues brought about by the choice of numerical method.

2.6.2 Scaling of individual cell properties

A single cell represented by an IB that is displaced out of equilibrium by, say, stretching, will relax back to a circle. If we were to run an identical simulation with half the time step, we would expect the dynamics to remain unchanged (up to numerical imprecision introduced as a result of the numerical scheme). Likewise, halving the fluid mesh spacing h would, provided we obey the criteria of Section 2.6.1, leave the simulation output unchanged. Changing the IB representation, however, by altering the number of nodes per boundary, N_k, requires a scaling of various parameters if we wish to recapitulate the same simulation.

To investigate this interplay, we consider the case where the node spacing in a single IB is decreased by a factor of α, starting from a reference value. Our goal is to derive the scaling required to ensure that the fluid flow, which determines the dynamics, remains unchanged. Two effects come in to play. First, the node spacing $\Delta\gamma_k$, which appears explicitly in the discretised force relation Equation (2.21), is reduced by a factor α, and therefore $\mathbf{F}$ must be increased by this factor in order to compensate. Second, because the boundary is represented by linear springs, we are now considering a system with α times the number of springs, each with length reduced by a factor α. Assuming the rest length, l_{int}, scales proportionally with the length of the connection, the average energy of a spring in the reference configuration is given by

$$E_{ref} = \frac{1}{2}\kappa_{int}^{ref}\left(\Delta\gamma_k - l_{int}\right)^2, \tag{2.33}$$

whereas the average energy of a spring in the new configuration is given by

$$E_{new} = \frac{1}{2}\kappa_{int}^{new}\left(\frac{\Delta\gamma_k}{\alpha} - \frac{l_{int}}{\alpha}\right)^2. \tag{2.34}$$

To ensure the potential in the IB is identical in both the reference and the new configurations, we must equate E_{ref} with αE_{new}, giving $\kappa_{int}^{new} = \alpha\kappa_{int}^{ref}$. Combining the scaling by α from both considerations, we thus find that to increase the number of nodes in an IB by a factor α we require an α^2 increase in κ_{int}. Figure 2.3b verifies this scaling.

We now consider the case of two interacting cells with identical mechanical properties. If we alter the resolution of nodes around each IB, how must we change the cell–cell interaction force parameters k_{ext} and l_{ext} to recapitulate the same dynamics in a given simulation? Increasing the number of nodes by a factor α in each IB relative to a reference scenario will also increase the number of connections, determined via Equation (2.12), by a factor α. As the IBs are unchanged in size, l_{ext} should remain the same, and thus the potential contained within the boundary interactions will have increased in proportion to the number of connections. Thus, $\kappa_{ext}^{new} = \kappa_{ext}^{ref}/\alpha$ is the necessary scaling to ensure the simulation dynamics remain unchanged. Figure 2.3c shows summary statistics from a simulation verifying this scaling.

Putting these two results together, when increasing the density of nodes in a simulation by a factor α we must scale κ_{int} by α^2 and κ_{ext} by $1/\alpha$. To encapsulate this within our computational framework, we introduce an 'intrinsic length' relative to which the scaling described here is applied. Due to this, the required scaling is not manually applied by the user; the simulation dynamics remain unchanged when the user alters the node spacing.

2.6.3 Convergence analysis

Here, we demonstrate how the numerical implementation converges with time step, fluid mesh spacing, and IB node spacing. We conduct this convergence analysis using a simple prototype simulation of an elliptical IB undergoing relaxation for a fixed simulation time. For each of the three parameters of interest, Δt, h, and $\Delta\gamma_k$, we perform a series of simulations where only the parameter of interest is varied, and collect a single summary statistic, the ESF, from which we can verify convergence.

To analyse convergence with time step, we run the relaxation simulation nineteen times, starting with $\Delta t = 0.5$ and each time reducing Δt by a factor of $\sqrt{2}$. Figure 2.4a demonstrates convergence of the ESF with time step. We assume the ESF associated with the finest time step to be the best approximation to the continuum limit, and define the error in ESF for each simulation to be the absolute difference between the ESF and this best value. Omitting the penultimate value, the gradient of a log–log plot of this error against time step is 1.11, demonstrating the order of convergence is approximately linear.

Similarly, to demonstrate convergence with fluid mesh spacing we run fifteen relaxation simulations starting with $h = 1/32$ and each time reducing h by a factor of $\sqrt{2}$. We need to pick a fixed large number of IB nodes to eliminate the node spacing ratio issue discussed in Section 2.6.2, and so as not to vary $\Delta\gamma_k$. Figure 2.4c shows

convergence of the ESF with h. Defining the error in a similar manner to above, we find the log–log gradient to be 1.37, demonstrating the order of convergence to be subquadratic. Finally, to demonstrate convergence with IB node spacing, we run sixteen relaxation simulations, starting with $\Delta\gamma_k \approx 0.014$ and each time reducing $\Delta\gamma_k$ by a factor of $\sqrt[3]{2}$. Figure 2.4 shows the ESF converging. The log–log gradient is 1.49, demonstrating the order of convergence to be subquadratic.

In addition to convergence of the numerical implementation, we also require our implementation of cell division to converge with IB node spacing: for a given cell division, the shape of the resulting daughter cells should be independent of the choice of boundary parametrisation. We verify this convergence by performing cell division operations on a number of elliptical IBs, each represented by a different number of nodes, and using the ESF as a summary statistic of daughter cell shape. Figure 2.5 shows results with a log–log gradient of 1.96, demonstrating the order of convergence to be quadratic.

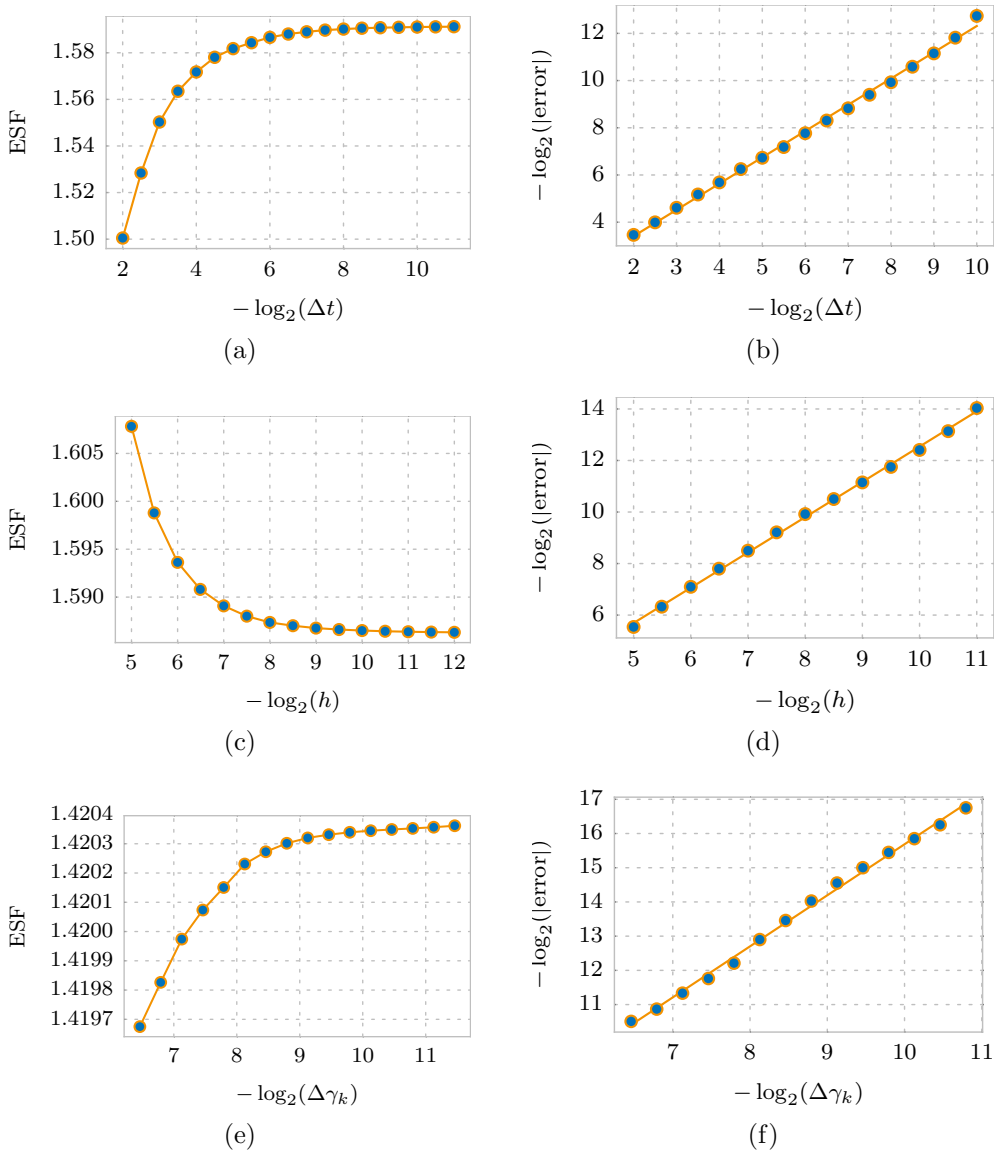


Figure 2.4: **Convergence of computational implementation.** *(a) Convergence with time step. Nineteen simulations with different values of Δt were run for a fixed duration of 10 time units, with the following fixed parameters: initial $ESF = 2.0$, $N = 128$ nodes, $l_{int} = 50\%$ of node spacing, $\kappa_{int} = 10^7$, $Re = 10^{-4}$, with 128×128 fluid mesh points, relative to an intrinsic spacing of 0.01. (b) Linear fit between error and time step, with a gradient of 1.11. (c) Convergence with fluid mesh spacing. Fifteen simulations with different fluid mesh spacings, h, were run, for a fixed duration of 10 time units, with the following fixed parameters: initial $ESF = 2.0$, $N = 8192$ nodes, $l_{int} = 50\%$ of initial node spacing, $\kappa_{int} = 10^7$, $Re = 10^{-4}$, and $\Delta t = 0.01$, relative to an intrinsic spacing of 0.01. (d) Linear fit between error and fluid mesh spacing, with a gradient of 1.37. (e) Convergence with IB node spacing. Sixteen simulations with different numbers of IB nodes, therefore modulating $\Delta\gamma_k$, were run for a fixed duration of 10 time units, with the following fixed parameters: initial $ESF = 2.0$, $l_{int} = 50\%$ of initial node spacing, $\kappa_{int} = 10^7$, $Re = 10^{-4}$, $\Delta t = 0.01$, and 64×64 fluid mesh points, relative to an intrinsic spacing of 0.01. (f) Linear fit between error and node spacing, with a gradient of 1.49. For details on how to obtain the code for these simulations, see Appendix A.1.*

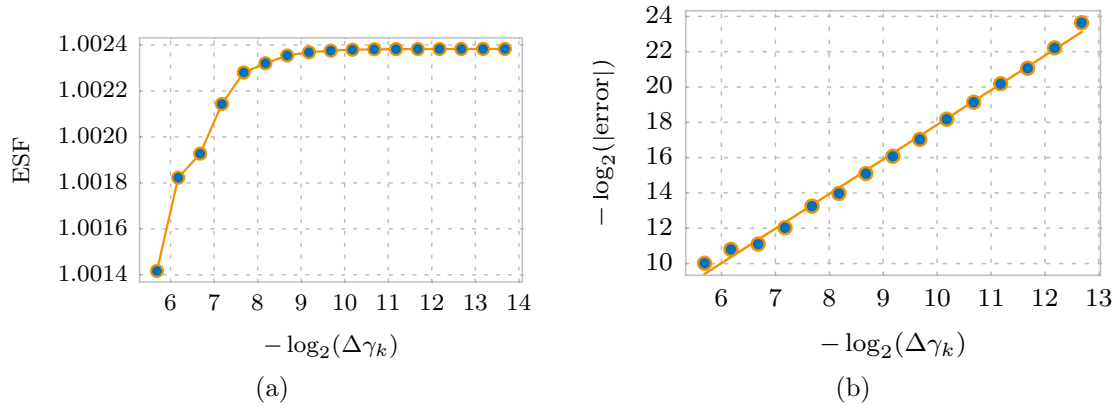


Figure 2.5: **Convergence of cell division implementation.** *(a) Seventeen simulation results showing the ESF of an IB resulting from the application of the cell division algorithm (Section 2.5.2), from elliptical IBs with varying values of $\Delta\gamma_k$. (b) Linear fit between error and $\Delta\gamma_k$, with a gradient of 1.96. For details on how to obtain the code for these simulations, see Appendix A.1.*

Guideance on choice of parameter values. With this convergence analysis completed, we can give a sensible rule of thumb for selecting appropriate values of Δt, h and $\Delta\gamma_k$. As we can see from Figures 2.4 and 2.5, the choice of $\Delta\gamma_k$ is by far the least important in the sense that the error over sensible choices of $\Delta\gamma_k$ is much smaller than for Δt and h. We, therefore, suggest first choosing an appropriate value for h based on the modelling constraints. Next, selecting an appropriate $\Delta\gamma_k$ to ensure the node spacing ratio remains less than 1 (see Section 2.6.1) will certainly not increase the magnitude of any error. Finally, with reference to Figure 2.4 selecting $\Delta t \approx h$ will keep the error due to timestepping roughly equivalent to the error associated with the fluid mesh spacing.

2.7 Potential applications to epithelial morphogenesis

Simple epithelia are cell monolayers that cover many surfaces in complex organisms, and are important systems to study due to the effects of their complex cellular rearrangements on embryonic development. We design the following prototype simulation study to draw comparisons with, and propose extensions to, studies of epithelial packing using VMs, such as those of the *Drosophila* wing imaginal disc presented by Farhadifar and colleagues [42]. The aims of this section are threefold. First, we demonstrate that our implementation is capable of simulating proliferation and the growth of entire tissues, comprising hundreds and potentially thousands of cells. Second, we elaborate on details of how our computational implementation

can be used to explore problems in developmental biology, where cell geometries are impacted by biological processes such as progression through the cell cycle. Finally, we highlight specific questions relating to epithelial morphogenesis which could be investigated by an IB implementation, but which could not be readily studied with previously presented frameworks such as VMs.

VMs, which treat the apical surface of simple epithelia as a tessellation of polygons, have been successfully used to probe many questions in developmental biology [47]. We aim to simulate an initially small group of cells that undergo repeated rounds of division, using our IBM framework. Specifically, we begin each simulation with nine cells represented by hexagonal IBs arranged in a honeycomb packing. We permit cells to follow a cell cycle model where they can grow and divide a fixed number (five) of times. The duration of the G1 (growth) phase in the cell cycle model is drawn randomly from an exponential distribution. We finish each simulation when all proliferation has finished, with approximately 500 cells. We choose to implement cell divisions, as described above, as occurring on a time scale faster than that of the bulk tissue mechanics, and therefore that they occur during a single time step. Figure 2.6a shows a snapshot of one such simulated tissue resembling, at least qualitatively, the apical surface of the wing imaginal disc epithelium.

Between different simulations, we vary two parameters: the internal and external spring constants (κ_{int} and κ_{ext}). We collect the polygon class distribution (PCD), the distribution of cell neighbour numbers, as a simple summary statistic that can be quantitatively compared to living tissues as well as other simulation studies, and which constitutes one simple readout of the tissue morphology. It is not straightforward to calculate the PCD for a population of IB cells, and detailed method for matching a PCD between vertex and IB populations lies outside the scope of this chapter. It is, however, explored in detail in Section 4.2.1 where a robust algorithm for this purpose is described. Figure 2.6b shows the variation in PCD for a fixed value of κ_{int} while κ_{ext} is varied. Using such summary statistics, we can hope to relate parameters between different models to allow like-for-like comparisons.

In addition to the PCD, Farhadifar and colleagues present the area distribution as a summary statistic quantifying the behaviour of the model in different parameter regimes in their VMs. The area distribution is the average area of each polygon class (squares, pentagons, hexagons, etc): in the *Drosophila* wing imaginal disc, experimental work shows a linear relationship in area distribution, with squares being on average roughly a third the size of octagons. The details of this are presented in Figure 2 of [42]. This linear distribution is recapitulated for certain parameter regimes

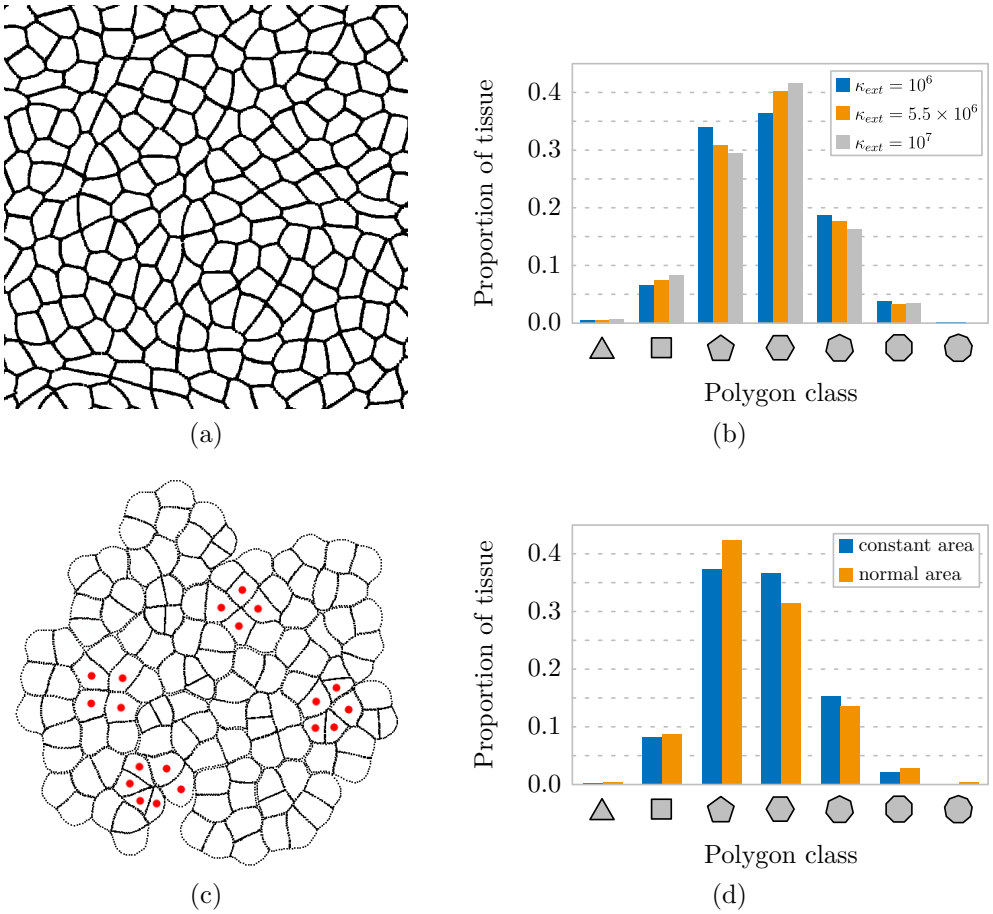


Figure 2.6: **Simulated epithelial tissues.** *(a) Snapshot of tissue morphology for one realisation with $k_{int} = k_{ext} = 10^7$, $N = 48$ nodes per boundary, with 256×256 fluid mesh points. (b) Polygon class distribution varies with κ_{ext}, for fixed $\kappa_{int} = 10^7$. (c) Snapshot of a growing tissue simulation, displaying several four-cell and higher-order junctions, highlighted with red dots. (d) Polygon class distribution varies with area distribution. Left hand bars: constant target area for each cell. Right hand bars: cell target areas drawn from a normal distribution. For details on how to obtain the code for these simulations, including all parameter values, see Appendix A.1.*

in the VM, but varies dramatically; 'soft networks' (Farhadifar's case III) have a flat distribution where polygons of each class are, on average, the same size.

The area distributions, however, are emergent properties of the VM, and cannot themselves be altered directly. By contrast, in the IBM the sizes of cells are precisely determined via fluid sources and sinks. This makes the IBM an excellent candidate for probing the effect of variability in cell size on the tissue morphology.

Figure 2.6d shows a dependence on polygon distribution, for fixed κ_{int} and κ_{ext}, on the choice of cell target area: in one simulation, each cell attains a globally fixed target area, while in the other, each cell's target area is is picked from a normal distribution.

This dependence highlights the potential to explore the role of regulated changes in cell size on tissue morphology, a question that cannot be directly probed using a VM. Moreover, this size control on the cellular scale could also be used to explore the effect of regulated cell death on tissue morphology.

Finally, an additional morphological feature present in developing epithelia, but not in standard vertex simulations, are multicellular rosettes [47]. Rosettes are structures in which a single central junction is shared by five or more cells. Settings in which rosettes are observed include epithelia in zebrafish and *Drosophila*, the vertebrate pancreas and the neural stem cell niche [62], and in the mouse anterior visceral endoderm, where they have been shown to play an important role in development [47]. While rosettes do not form naturally in VMs, Figure 2.6c shows several examples in a single snapshot of rosette and similar four-cell structures forming spontaneously. The IBM, therefore, presents itself as a natural framework within which to study the role of rosettes on tissue morphology.

While there is work left to do to allow a meaningful comparison between the two models, we nonetheless have identified several clear applications within epithelial morphology for which IBM frameworks appear well suited.

2.8 Discussion

In this chapter, we have presented a thorough description of the equations governing the IBM, and full details of a common discretisation approach and method of numerical solution. We have presented an efficient computational implementation, as part of a mature and thoroughly tested C++ library designed specifically for computational biology simulations. We have demonstrated numerically various parameter scaling properties of the IBM, and have demonstrated the convergence properties of our implementation. Finally, we have demonstrated, through a prototype simulation study, the potential utility of the IBM to investigate questions in epithelial morphogenesis. In this section, we return to several choices made during the formulation of our IBM framework.

2.8.1 Stokes or Navier–Stokes

The first such choice was whether to solve the full Navier–Stokes equations, or whether to solve the Stokes equations in the low Reynolds Number limit. To address this question, we first emphasize that the 'fluid' underlying the IBM need not have a direct physical correlate. It may be helpful to think of the fluid simply as a tool by which

the positions of the boundaries are updated, and which has certain 'nice' properties (such as volume preservation inside closed contours), although some authors have nevertheless sought to draw parallels between this fluid and the cell cytoplasm and extracellular medium [138]. A concrete example, though, of the difference between the IBM fluid and the fluid-like properties of the underlying biological system is in the case of a stationary circular boundary: if there is a resultant elastic force, there will be a non-zero body force in the IBM fluid and therefore an induced flow. As the fluid cannot be assumed to faithfully represent underlying biology, it is not obvious that modelling a biological situation with small Reynolds number necessarily means the Reynolds number in an associated IB problem need also be small. Rejniak and colleagues, for instance, derive a 'biological' Reynolds number of 10^{-9}, but use the value 5.9×10^{-5} for their simulations [139], a value chosen so as to recapitulate the relevant dynamics. This discrepancy demonstrates that the IBM fluid cannot be expected to adequately mimic the fluid-like properties of the underlying biology, and thus that we must take care in assuming an appropriate Reynolds number in IBM simulations need necessarily be very small. Further investigation is required to ascertain the relationship between 'fluid' properties *in vivo* and *in silico* for the IBM. Cutting experiments, for instance, where tissue is observed to recoil after ablation, could be used to fit an appropriate Reynolds number for the IBM in order to match *in vivo* dynamics. While IBM implementations based on Stokes flow do exist [17, 88, 165], we have chosen to implement the full generality of the Navier–Stokes problem. This keeps open the possibility of modelling situations where inertial effects cannot necessarily be neglected, while acknowledging that there are scenarios in which the reduced problem may be appropriate, and computationally much less expensive to solve.

2.8.2 Discrete delta function

We made a specific choice for the form of the discrete delta function. Peskin [124] derived the following form for ϕ, in contrast to that presented in Equation (2.10):

$$\phi(r) = \begin{cases} \frac{1}{8}\left(3 - 2|r| + \sqrt{1 + 4|r| - 4r^2}\right), & |r| \leq 1, \\ \frac{1}{2} - \phi(2 - |r|), & 1 \leq |r| \leq 2, \\ 0, & 2 \leq |r|, \end{cases} \tag{2.35}$$

While the functional form appears quite different, the numerical values taken by the different formulations of ϕ are very similar (differing by less than 0.008 at any point in

the domain). Given this remarkable similarity, using one form rather than the other may be decided by computational efficiency. In practice, we find the trigonometric function slightly quicker to compute during a simulation, which is probably due to difficult branch prediction of the 'if' statement necessary to compute ϕ using Equation (2.35). The proportion of the total simulation time spent evaluating the discrete delta function is, however, small enough that in practical terms the choice of ϕ is immaterial.

2.8.3 Intercellular interaction terms

Third, we will briefly discuss the choice of functional form for the intercellular interaction terms. The sharp cut-off represented by the interaction distance d_{ext} in Equation (2.12) may be unphysical, as it implies that when boundaries move apart, the opposing force linearly increases with distance until instantaneously becoming zero at distance d_{ext}. A different functional form may mirror the underlying behaviour more closely, and one such example is the Morse potential [111], which has a functional form $V(r) = \kappa \left(1 - e^{-a(r-l)}\right)^2$ where κ and a denote the depth and width of the potential well, respectively, r is the distance between the interacting nodes, and l is the equilibrium distance of the bond. The force between two IB nodes would, as a result of such a potential, be exponentially repulsive at short distances, have an attractive peak at a medium distance, and tail off at long distance. A cut-off at a value of d_{ext} would still be needed for computational reasons, but this cut-off would be at a low value of the force, rather than at the maximum value as is the case with linear springs. To what extent the choice of functional form influences IBM simulations is an open question, and a topic for further study.

2.8.4 Balancing sources

Fourth, in Section 2.4.7 we gave no precise formulation for the number of fluid sources, $M-N$, in excess of those associated with IBs. The purpose of these additional sources is to balance the net fluid creation due to processes such as cell growth, to ensure a constant fluid volume within the domain Ω. In our implementation we choose $M \approx 2N$, and initially place these equidistant along the boundary $y = 0$. Rejniak and colleagues [139] use a similar approach but do not specify the number of such additional sources, while Dillon and Othmer [34] use exactly four but do not specify their initial locations. The implications of such choices have not been systematically investigated, and to what extent these choices impact upon the results of simulations

is a topic for further study.

2.8.5 Constant viscosity

Finally, we address the assumption that the fluid viscosity, μ, is constant across the entire fluid domain. This choice is common but not ubiquitous in IBM studies, with Fedosov and colleagues [43], for instance, choosing different fluid viscosities inside and outside red blood cells. They found that the characteristic timescale of boundary deformation was dependent on the difference between the internal and external viscosities. This is an interesting result that came from domain specific knowledge of relevant viscosities. It is worth noting, however, that more recent studies in the same area (for instance by Krüger and colleagues [82]) do not choose different viscosities and this choice is made primarily for the substantial simplification of solving the fluid problem. Further work is needed to fully establish the extent to which non-constant viscosity drives dynamics in IBM simulations, for for the work presented in this thesis we shall keep μ constant.

2.9 Conclusion

With the availability of experimental biological data from molecular and live-imaging studies at ever finer resolutions, we need modelling tools able to represent that information on arbitrary length scales. This is a driving motivation for developing frameworks such as this, as they can better incorporate such data on spatial resolutions finer than existing models. In addition to incorporating such data, the IBM is one example of a more geometrically detailed cell-based modelling framework capable of representing cell geometries away from their equilibrium shapes. Both of these aspects are explored in Chapter 5 where we demonstrate the ability of this framework to incorporate spatially localised transmembrane protein expression in a model where cells display defined 'hook' shapes.

Furthermore, another major advantage to modelling populations of individual cells rather than confluent tissues (such as in the VM) is the ability to represent tissue viscoelasticity. Viscoelasticity is the property of exhibiting both viscous and elastic characteristics when undergoing deformation, and the ability for cells in the IBM to 'flow' past one another is a form of time-dependent strain that models such as the VM do not typically exhibit. Examples of such viscoelastic behaviour are explored in Chapter 4, where the ability for relative motion of cells is necessary in the context of cell sorting.

LBIBCell, (Tanaka and colleagues [164]), is another IBM for cell populations. In comparison to LBIBCell, our Chaste implementation is able to utilise existing extensive and highly-tested infrastructure for cell-based modelling. Along with built-in functionality for biological modelling such as cell cycle models, cell area modifiers, timestepping and solvers for coupled problems, there are two main strengths of implementing the IBM within Chaste. First is the software quality and reliability, discussed at length in Chapter 3, which lends confidence to the individual components underling the IBM implementation, but also enables the IBM implementation itself to slot into the exiting unit testing and continuous integration framework. Second, a particular strength of Chaste is the availability of a number of modelling frameworks in a single software package. This enables the most appropriate modelling framework to be selected when modelling the given biology and, furthermore, enables direct comparisons between different modelling frameworks on benchmark problems, which is described in detail in Chapter 4.

Finally, a strength of models such as the IBM is the ease with which cellular heterogeneity (for example, through patterning mechanisms) may be incorporated, and the consequences for tissue-scale behaviour be simulated and explored. The development of methods to efficiently explore the parameter space of such models, perform inference and model calibration against quantitative datasets, and analyse the tissue-level mechanical properties of such models remain avenues for future work in this area.

Chapter 3

Efficient implementation and exploration of cell-based models within an open source framework

Having now presented a detailed description of the IBM, and a careful numerical analysis of its implementation within Chaste, we turn our focus to the software engineering principles that underpin such research software, and research software in academia more generally. The work to develop and refine software best practices within Chaste, some of which is presented in this chapter, has led to a manuscript to be submitted to *J. Open Source Softw.*, on which I will be the first author.

3.1 Introduction

Software is an integral part of modern research. Indeed, the importance of investing in software development has recently been recognised by the EPSRC through their *Collaborative Computational Projects* and their research software engineer fellowship scheme [40]. A national survey across 15 Russell Group universities in 2014 found that research software is fundamental to 67% of researchers, and that 56% of researchers develop their own software [16]. A smaller survey conducted at the University of Oxford in 2018 reinforces these figures, with 71% of respondents reporting that they develop their own research software, and 70% reporting that it is vital to their work [26]. Ought we to be worried, then, that the majority of those developing software are self taught, or that few are confident in basic software engineering skills?

Research software ranges greatly in size and complexity, from short scripts used to process data or plot graphs, to large libraries for high-performance simulation.

Software is an indispensable component of this thesis. Evidence strongly suggests a need to ensure the quality, availability, and maintainability of research software, no matter the number of lines of code involved.

We suggest that all research software should adhere to three principles. First, code should be robust, correct and fit for purpose. Second, code used to generate results or process data ought to be published alongside the work itself. Third, code should be maintainable: minimal effort should be required to keep software working over time, and someone coming across the code in several years time should be able to use it. These three principles will be explored in greater detail shortly, but by way of motivation we begin with a cautionary tale.

3.1.1 Reinhart and Rogoff: a cautionary tale

In 2010, shortly after the global financial crisis, a paper was published by Reinhart and Rogoff, two influential economists with Rogoff a chair in public policy at Harvard University and former Chief Economist at the IMF. Their paper, *Growth in a Time of Debt* [133], investigated the relationship between a country's external debt as a percentage of gross domestic product (GDP), and its economic growth. Reinhart and Rogoff concluded the existence of a debt threshold above which GDP growth suffered significantly.

This paper influenced public policy. George Osborne, then UK Chancellor of the Exchequer, cited Reinhart and Rogoff's work prominently in his 2010 Mais Lecture *A new economic model* [117]; while Paul Ryan, then US House Budget Committee Chairman, delivered a budget *The Path to Prosperity* that cited *only* Reinhart and Rogoff [140]. Both the UK and the USA embarked on policies of austerity to reduce public debt and, while Reinhart and Rogoff were far from the only economists publishing similar findings, their paper was among the most stark and was certainly the most highly cited.

The paper, however, contained errors. In addition to methodological flaws (disputed by Reinhart and Rogoff [134]), there were a number of undisputed coding errors in the spreadsheet used for their analysis that directly affected the strength of the published results. These coding errors were identified after Reinhart and Rogoff allowed access to their original data and analysis to fellow economists Herdon and colleagues, these authors finding that "selective exclusion of available data, coding errors and inappropriate weighting of summary statistics lead to serious miscalculations that inaccurately represent the relationship between public debt and GDP growth" [65]. They concluded that average GDP growth was 2.2% not the

−0.1% published by Reinhart and Rogoff.

This is one example of a coding error identified in published research. While the instance of Reinhart and Rogoff influenced policy in a much larger way than most academic papers, several aspects to the story are widely applicable. First, in most instances, it would have been impossible to identify that coding errors were made. Perhaps only because of the magnitude of the impact that this paper had did it become necessary for the authors to make their original analysis available. This raises the (unanswerable) question of how many other coding errors underlie published results. As the majority of code written is *not* published alongside results, it would be impossible to ascertain the true magnitude of the problem, but there are certainly indications. At the University of Oxford, 63% of students and postdocs who develop their own research software have either not heard of, or are not confident with, unit testing (testing the correctness of individual code units), and the number is 29% even for version control (a system for tracking changes in source code), a fundamental and indispensable tool in software development (see Figure 3.1 for further details). This may not be surprising given that 70% of researchers who develop their own research software are self taught, but it does indicate the huge extent to which software is written in academia by people without formal training, and who are not confident in the basic tools necessary to generate good-quality software. These numbers indicate that a large proportion of published results are underpinned by poor quality code, even if that code is not necessarily wrong.

Second, problems are exacerbated by the analysis code not being made available during publication. This has two substantial impacts. First, large mistakes, such as those present in Reinhart and Rogoff's spreadsheet, would have a greater chance of being caught at the review stage. Even if not explicitly caught, knowing that code is available for reviewers to examine would probably increase the code quality and documentation by requiring researchers to have more confidence in it before submitting it for review. Second, making code available greatly helps reduce problems related to reproducibility, which is a growing concern [99]. It is not practical for every detail of an implementation or analysis to be presented in the text of a publication, but having code available enables those attempting to reproduce results to obtain every detail if necessary. Indeed, several of the pieces of work described later in this chapter invite the reader to look in the source code for full implementation details (instructions for accessing the source code are available in Appendix A.2). Third, code should be easy to maintain, allowing software to remain useful with the inevitable flux of people and resources available.

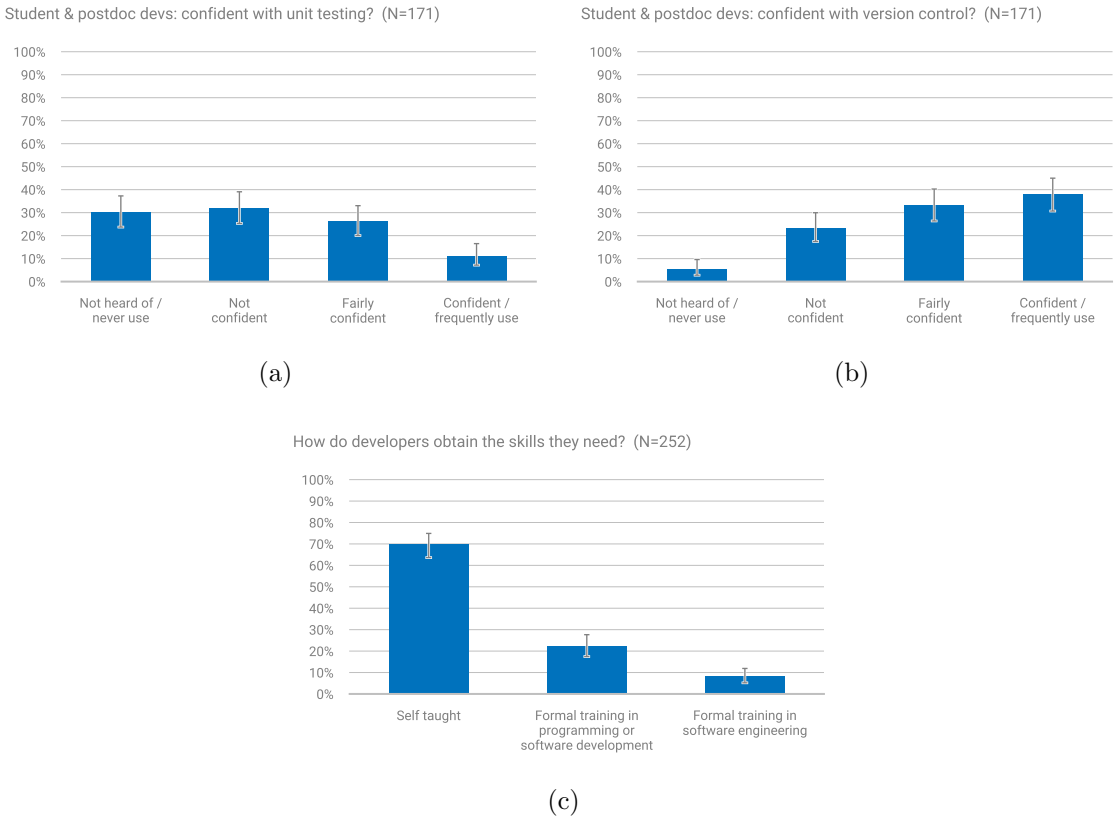


Figure 3.1: **Aspects of software development at the University of Oxford.** *(a) Proportion of students and postdocs developing research software who are confident with unit testing. (b) Proportion of students and postdocs developing research software who are confident with version control. (c) How researchers obtain the skills necessary for developing software. (a-c) Error bars are 95% credible intervals, assuming a multinomial likelihood and an uninformative Dirichlet prior. Data are from the University of Oxford Research Software Engineering survey [26].*

Software errors can lead, at best, to retractions if caught [106], and at worst misinform future research and public policy. These issues, illustrated by the Reinhart and Rogoff affair, inform the three principles outlined above. It is precisely because of the current problems around code quality in academia that we advocate strongly for the application of software best practices when writing code, and of spending the time necessary to ensure the quality, robustness, availability and maintainability of code.

It is in this context that we frame the work presented in this chapter, which is split into three sections: first, general contributions I have made to the Chaste libraries; second, specific contributions I have made to the Chaste libraries; and third, a general pipeline that I have developed for running simulations and presenting

results to collaborators.

Section 3.2 deals with adding to the quality, robustness, and maintainability of Chaste as a high-quality piece of research software, focussing on the software engineering of the project itself rather than particular features of code. In Section 3.3 we focus on specific contributions to Chaste, and detail several algorithms and additions that I have made to the core Chaste libraries that underpin modelling and simulation work presented in this thesis, but which do not clearly fit into other chapters specifically. Section 3.4 introduces a software tool for running simulations and presenting the output to colleagues in a generic manner, avoiding the need for access to any specific software. While this tool interfaces specifically with Chaste, the concept is applicable more widely.

3.2 General contributions to the Chaste libraries

The Chaste project is an excellent example of software that has been well engineered, and conforms to a number of software best practices. Chaste [107, 127], as introduced in Chapter 2, has extensive test suites ensuring 100% statement coverage (every statement in the source code is hit during unit testing), employs test-driven development to ensure error-free code, and employs continuous integration to verify that codebase changes have no unintended consequences. In addition, as part of the continuous integration, Chaste is tested with several operating systems and combinations of dependencies to ensure maximum compatibility and maintainability over longer periods of time. Because of the lengths gone to in ensuring the codebase is well engineered, Chaste is a stable and reliable piece of software that has existed for over 13 years.

In spite of this, over time as technologies and best practices change, additions and alterations are necessary to keep a codebase at the leading edge and to maintain its credibility as a well-engineered product that can be relied upon into the future. This section contains details of a number of additions and improvements that I have made to Chaste, and these collectively improve the foundations that underpin the research Chaste enables.

Chaste is a large C++ project. C++ is a natural choice of language for a project such as Chaste as it is both object-oriented (allowing for high-level abstractions that make it easier to reason about, and organise the structure of, individual components of the library), and fast. Downsides of C++ include that it is complicated, often requires the manual allocation and deallocation of memory, and is a difficult language for

newcomers to write well [156].

Since Chaste was started in 2005, the C++ programming language has undergone significant changes. Three new ISO standards have been published which fundamentally change a number of paradigms and best practices recommended for using the language. In addition, new tools are now available that, hand-in-hand, help to identify subtle software flaws and highlight coding errors or bad practice when writing code.

The ability to utilise these new features and tools is important for two reasons. First, more confidence can be placed on the quality and reliability of Chaste as a tool, which is important if we are to trust the results generated with it. Second, the maintainability of Chaste as a library is improved. Language features are evolving to make it much easier to write correct code, and tools that allow the automatic verification of new code, particularly written by non-expert C++ coders, will help ensure Chaste can be maintained in the future with minimal effort, and that Chaste can evolve towards language features and paradigms that will be used in the future.

3.2.1 Using modern C++

The C++ programming language was first developed in 1979 by Bjarne Stroustrup at Bell Labs. It was standardised in 1998 as ISO/IEC 14882:1998, and since then the standard has been updated in 2003, 2011, 2014 and 2017. Further changes to the standard are expected every 3 years. The standardisation in 2011, in particular, made a number of major changes to the language, and these changes enable developers to write code that is more robust, less error-prone, shorter and easier to read [105].

I worked towards the adoption of C++11 in Chaste, which required substantial alterations to the build system, source code and interface with dependencies, and version 2017.1[1] was the first C++11 version of Chaste to be released. Now that Chaste supports C++11, work is ongoing to adopt, where appropriate, the new features available.

3.2.2 Static analysis

While thorough unit tests backed by continuous integration provide a substantial indication that a piece of software is correct, they can only spot errors that have been explicitly tested for. Edge cases, poor use of C++ features, and even some copy-and-paste errors can be overlooked and end up in the software repository. Static

[1] `https://github.com/Chaste/Chaste/releases/tag/2017.1`

analysis is a term that refers to automated processes for identifying common errors and oversights in source code that cannot necessarily be caught during normal unit testing.

One such static analysis tool is `clang-tidy`[2], an extensible tool for analysis of C++ code that can find errors related to performance, readability, portability, security and others. Of particular interest to Chaste are checks against the C++ core guidelines [157], a set of rules aimed at making it easier to adopt C++ best practices.

I have added infrastructure to Chaste that runs `clang-tidy` on every commit to the Chaste repository. The results of these checks are uploaded to a publicly accessible website, allowing developers to check the warnings produced from each source code file. One such example can be found here: `https://chaste.cs.ox.ac.uk/buildbot/Continuous%20Clang_Tidy/clang_tidy126/`. In the future, it is hoped that these warnings can be addressed, at which point further commits to the repository will be rejected if new warnings are introduced. The introduction of these tests gives Chaste developers easy access to the latest coding best practices, which are constantly being updated, and constitutes a step forward in the reliability and maintainability of Chaste.

3.2.3 Monitoring test performance over time

Unit tests are the basic tool used to ensure correctness of individual functions within a codebase. There are two key purposes of unit tests. First, they ensure that specific functionality is correct; for example if you write code to compute the n^{th} prime number, then it should find the 854^{th} prime number to be 6619. Second, unit tests verify that any changes are made in a specific function do not have any unintended downstream consequences. To ensure this, the entire suite of unit tests must be re-run each time a change is made to the source code. This is a core component of the software engineering practice of continuous integration.

A key issue that unit testing does not address, though, is how the performance of a codebase changes over time. Whenever functionality is added to a C++ library, more machine code is generated, and this can mean slowdowns in simulation runtime. Much additional functionality is independent of existing functionality, and in these cases no runtime overhead is necessarily felt, but the problem comes when code refactoring or enhancements are made to existing functions. In this case, simulation runtime may increase. This needs to be measured, as even small increases may compounded over

[2] `http://clang.llvm.org/extra/clang-tidy/`

a period of years to have significant effects on the efficiency of the code, and all the while every unit test will still pass.

This is a surprisingly difficult issue to tackle. Measuring the time taken by an executable to run depends on a large number of factors. On a run-by-run basis, the time will fluctuate based on what else the machine is doing at the time, which is extremely difficult to control for. On the software level, the time will fluctuate based on the operating system, the kernel version, the compiler version, the compiler, and the version of any libraries linked against. Finally, on the hardware level, no computer lasts forever, and upgrading the machine will compound all of the above.

There is no sensible metric that can give an unambiguous value for the length of time it takes to run a specific simulation. The best we can do is to take measurements, monitor them over time, and identify any worrying trends that may indicate that a specific change, or a set of changes, have had a detrimental impact on code performance. To improve this monitoring for Chaste, I have implemented a Python interface, as part of the existing continuous integration framework, that provides comprehensive monitoring of simulation performance over time. This implementation uses the existing suite of tests that are run for code profiling purposes, and compiles historical information into an easy-to-access web interface, providing a tabulated view of each simulation showing 'sudden increase' and 'gradual increase' warnings (Figure 3.2a), as well as links through to graphs showing historical runtime performance for visual inspection (Figure 3.2b).

In more detail, the web interface provides a warning if there has been a sudden or gradual increase in simulation runtime. A sudden increase is defined as whether the most recent run exceeded the average of the previous 10 runs by > 1.96 standard deviations. A gradual increase is defined as whether the average of the most recent 10 runs exceeds by $> 10\%$ the average of the 10 runs 35 runs previously.

The graphs present the actual runtime in seconds to allow a Chaste developer to identify potentially worrying trends. The graphs include information such as the specific version control revision number so that it would be obvious if there was a specific spike after a certain revision.

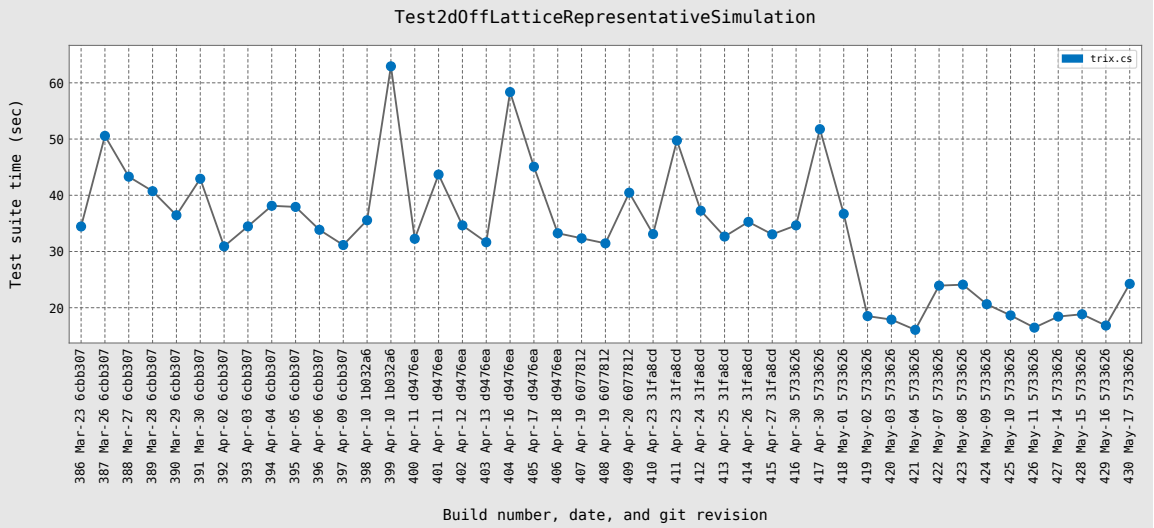
The impact of this performance monitoring is to provide a convenient and automated way of determining how code performance changes over time. This is of particular relevance in large long-running projects, such as Chaste, where small increases may compound over time, and having an extensive archive of historical run times gives at least the possibility of identifying where slowdowns may have been introduced. Having this resource available represents a significant asset to Chaste,

Test Suite Name	Profiling Output	Test Suite History	Sudden Increase	Gradual Increase
CompileTime	-	∞ 45	-0.49σ	-40.76%
Test2DMeshBasedCryptRepresentativeSimulation	svg output	∞ 45	-0.31σ	-54.77%
Test2DVertexBasedCryptRepresentativeSimulation	svg output	∞ 45	+1.23σ	-63.02%
Test2dOffLatticeRepresentativeSimulation	svg output	∞ 45	+1.86σ	-49.17%
Test2dVertexBasedSimulationWithFreeBoundary	svg output	∞ 45	-0.06σ	-58.48%
Test3dBidomainProblemForEfficiency	svg output	∞ 45	-0.69σ	-60.47%
Test3dBidomainProblemForEfficiencyWithFasterOdes	svg output	∞ 45	-0.62σ	-54.93%
Test3dBidomainProblemWithMetisForEfficiency	svg output	∞ 45	-0.55σ	-56.44%
Test3dBidomainProblemWithPermForEfficiency	svg output	∞ 45	-0.49σ	-55.60%
Test3dOffLatticeRepresentativeSimulation	svg output	∞ 45	-0.58σ	-49.04%
TestLongPostprocessing	svg output	∞ 45	-0.44σ	-60.15%
TestRepresentative3dNodeBasedSimulation	svg output	∞ 45	+0.23σ	-45.86%
TestRepresentativePottsBasedOnLatticeSimulation	svg output	∞ 45	-0.44σ	-53.75%

The "Sudden Increase" column measures how many standard deviations the most recent build time was above the mean of the previous 10 runs. The warning threshold is 1.96 standard deviations.

The "Gradual Increase" column compares the average run time of the most recent 10 builds, to the 10 builds 35 runs previously. The warning threshold is a 10% increase.

(a)



(b)

Figure 3.2: **Interface for tracking simulation performance over time.** *(a) Example web interface. Each row is a single profiling simulation, and columns provide warnings on runtime increases and links to further information. (b) Example profiling history graph. This graph shows the time taken to complete the previous 45 simulation runs, and the x-axis marks show the build number, date, and Git version control revision number.*

improving the long-term maintainability, reliability and credibility of the library, on which many results in this thesis are based.

3.3 Specific contributions to the Chaste libraries

We have already introduced the IBM implementation in Chaste (Chapter 2). For that implementation, and other parts of this thesis, to have been made possible, a number of additions, developments, and innovations have been necessary. In this section, we turn to my more specific contributions to Chaste, which have had direct relevance to the results in this thesis.

3.3.1 Voronoi vertex mesh generator

Every simulation of tissue dynamics that is run using VMs must start with an initial configuration of cells. Typically, simulations begin with a perfect hexagonal lattice of cells [157], and these cells then undergo dynamic changes such as division or rearrangements, with the assumption that the initial geometry and cell packing are unimportant.

The clear benefit of a regular hexagonal pattern is the ease with which it can be implemented. It is straightforward to calculate the vertex locations and whether or not any given vertex is on the boundary of the tissue. Importantly, it lends itself well to the generation of periodic domains, as a regular hexagonal lattice tessellates perfectly.

Given, however, that many biological systems do not exhibit perfect hexagonal cell packing, we would prefer to begin simulations from realistic initial conditions. The summary statistic, PCD, has been studied in detail, and distributions are well characterised in a number of developing epithelia [42, 52, 142].

Work by Sánchez-Gutiérrez and colleagues [142] demonstrates the applicability of Lloyd's algorithm [91] in generating realistic geometries for a variety of developing epithelia. Lloyd's algorithm partitions a finite region of Euclidean space into disjoint non-overlapping subsets as follows:

Input:

- An integer number N of randomly located initial seed points
- An integer number n of 'relaxation steps'

Output:

- A partition of space into N non-overlapping regions

Algorithm:

1. `for i = 1...n`

- Calculate the Voronoi diagram of the seed locations
- Move the seed locations to the centroid of each Voronoi domain

This algorithm iteratively calculates the Voronoi diagram, reseeded at each iteration with the centroid of the Voronoi domains from the previous iteration. Sánchez-Gutiérrez and colleagues demonstrate that applications of Lloyd's algorithm, with differing numbers of relaxation steps, have polygon and cell-area distributions in common with a variety of developing epithelia. The *Drosophila* prepupal wing imaginal disc epithelium, for instance, corresponds well to a tissue generated with five relaxation steps of Lloyd's algorithm.

Implementation within Chaste This work indicates that such an approach has the potential to generate biologically realistic initial geometric configurations for simulations of epithelial dynamics. In this section, we describe an implementation of Lloyd's algorithm within Chaste that allows robust identification of boundary nodes, generation of periodic domains, and arbitrary cell areas. This implementation makes use of the Boost.Polygon library[3] for computing the Voronoi diagrams.

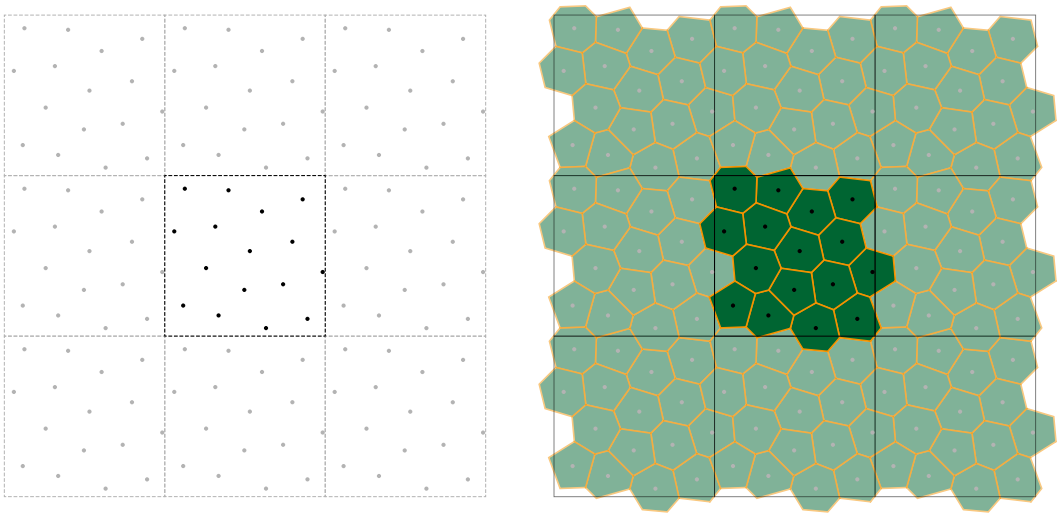
We first state the scenario more rigorously. Given N randomly located seeds in $[0,1] \times [0,1]$, we wish to find N Voronoi domains in $\mathbb{R}^2$ such that their union forms a periodic pattern with unit total area. A Voronoi domain about a given seed is defined as the region of the plane that is closer to that seed than to any other seed, i.e. given a seed s_i, the i^{th} Voronoi domain V_i is the set defined as

$$V_i = \{(x,y) \in \mathbb{R}^2 : \mathrm{d}(s_i,(x,y)) < \mathrm{d}(s_j,(x,y)) \quad \forall j \neq i\}, \tag{3.1}$$

where $\mathrm{d}(\cdot,\cdot)$ is the Euclidean distance between two points. The first issue is that the Voronoi domain corresponding to any seed on the convex hull of the set of seeds will extend to infinity. We must add additional seeds in order to keep each Voronoi domain finite.

The key insight is that if we create a 3×3 tiling of the original N seeds, distributed now in $[-1,2] \times [-1,2]$, the Voronoi domains corresponding to the original seed locations in the larger Voronoi diagram fulfil these requirements. Figure 3.3a shows the 3×3 tiling of 16 initial seed locations in the unit square. Figure 3.3b shows the Voronoi domains generated from the 16×9 initial points. For any Voronoi domain corresponding to one of the original 16 seeds that straddles the boundary of the

[3] `http://www.boost.org/doc/libs/1_66_0/libs/polygon/doc/index.htm`



(a) (b)

Figure 3.3: **Conceptual overview of mesh generation algorithm.** *(a) Initial seed points (black dots) are generated in the unit square (black dashed lines), and are then tiled eight times into a three-by-three square (grey points and dashed lines). (b) When the Voronoi diagram is calculated, the Voronoi domains corresponding to the initial seed locations are finite, form a periodic structure, and their areas to sum to unity.*

unit square, the tiling guarantees that a corresponding seed elsewhere belongs to a Voronoi domain of identical shape that encroaches into the unit square an identical amount, on the opposite side. Thus, after discarding all Voronoi domains that do not correspond to an original seed, the resulting geometry is perfectly periodic. We can obtain any desired average cell surface area by scaling the region by the square root of the desired size.

The following algorithm details the procedure, which is slightly more general than the conceptual overview presented above.

Input:

- Integer numbers X and Y of cells required in x and y directions
- An integer number n of Lloyd's relaxation steps
- A target average surface area a for the cells

Output:

- A Chaste vertex mesh object

Defined in:

- `VoronoiVertexMeshGenerator`

Algorithm:

1. Generate XY seeds in the box $\left[0, \frac{X}{\max(X,Y)}\right] \times \left[0, \frac{Y}{\max(X,Y)}\right]$ uniformly at random
2. Tile $9XY$ seeds into $\left[\frac{-X}{\max(X,Y)}, \frac{2X}{\max(X,Y)}\right] \times \left[\frac{-Y}{\max(X,Y)}, \frac{2Y}{\max(X,Y)}\right]$
3. Calculate the Voronoi diagram of all $9XY$ seed locations
4. Discard all Voronoi domains not corresponding to the initial XY seeds
5. Perform the relaxation loop, `for i = 1...n:`
   - Re-seed at the centroid of each of the XY Voronoi domains
   - Repeat steps 2 to 4
6. Identify boundary vertices
   - Loop over all vertices in the most-recently discarded Voronoi domains
   - Boundary vertices in the non-discarded Voronoi domains are exactly those whose locations perfectly coincide with a vertex in the discarded Voronoi domains: so, exhaustively compare each discarded vertex with each retained vertex to check for coincidence of position
7. If a periodic mesh is required, associate congruent vertices
   - The periodic domain is of size $\left[0, \frac{X}{\max(X,Y)}\right] \times \left[0, \frac{Y}{\max(X,Y)}\right]$, and a subset of boundary vertices will be outside this box
   - Relocate all vertices to be within this box by moving them modulo $\frac{X}{\max(X,Y)}$ in x and modulo $\frac{Y}{\max(X,Y)}$ in y
   - Some boundary vertex locations are now perfectly coincident with others. Iteratively remove these congruent vertices, replacing them with their counterparts, and removing their boundary vertex status
   - When no further boundary vertices exist, all congruent vertices will have been replaced
8. Rescale mesh to match target surface area
   - The current average surface area is $\frac{1}{\max(X,Y)^2}$, so the required linear scaling factor to apply to each location is $\sqrt{a}\max(X,Y)$.
   - Multiply each location by this factor leaving a domain of size $[0, \sqrt{a}X] \times [0, \sqrt{a}Y]$

Some implementation details are omitted here for brevity, and full details can be found in the source code (Appendix A.2). It should be noted that the efficiency of this algorithm could be substantially improved. Under certain circumstances, a full 3×3 tiling is clearly unnecessary, and reducing the tiling would reduce the size of the Voronoi calculations as well as the exhaustive search for boundary nodes. However,

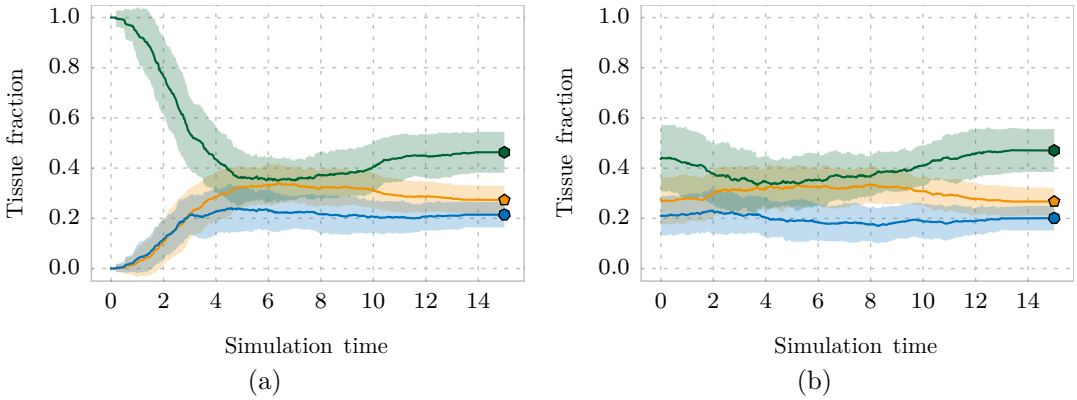


Figure 3.4: **PCD in simulations using different mesh generators.** *VM simulations were run starting with 36 cells, with each undergoing one complete round of cell division. There were 50 repeats each with (a) regular hexagonal initial conditions, and (b) Voronoi initial conditions with three Lloyd's relaxation steps. Solid lines represent mean tissue fraction of pentagons (orange), hexagons (green) and heptagons (blue), with the shaded area showing one standard deviation either side of the mean.*

as the Voronoi calculation is fairly cheap ($\mathcal{O}(N\log(N))$), and the mesh generation is only required once per simulation, the efficiency is unimportant and has not been addressed in the Chaste implementation.

With an implementation now in hand, we can analyse it in action. Figure 3.4 shows summaries of simple VM simulations initiated with a regular hexagonal initial condition and a Voronoi initial condition with three Lloyd's relaxation steps. The initial setup in each simulation was a 6×6 lattice, and the simulation was allowed to evolve over time with each of the 36 cells undergoing a single round of division.

While the distribution of tissue fractions ends up very similar in the two cases, there is a significant 'burn-in' time required in simulations using the regular hexagonal initial conditions (Figure 3.4a) whereas, in the Voronoi case, little or no 'burn-in' appears necessary. This is the case in spite of cell proliferation, which rapidly alters local cell–neighbour connectivity. In studies with no proliferation, it is common to start from a hexagonal packing of cells and run a simulation with either noise or proliferation added until it reaches some intermediate steady state before starting the 'real' simulation. An example of this approach is a VM of cell sorting [118]. Having the ability to generate realistic initial conditions for simulations obviates this need. Indeed, this functionality is used to reduce burn-in time during simulations of VM cell sorting in Section 4.3.1. During these simulations, burn-in was reduced from 10 time units to 1, with the simulation lasting 100 time units, representing a reduction in simulation time of approximately 8%.

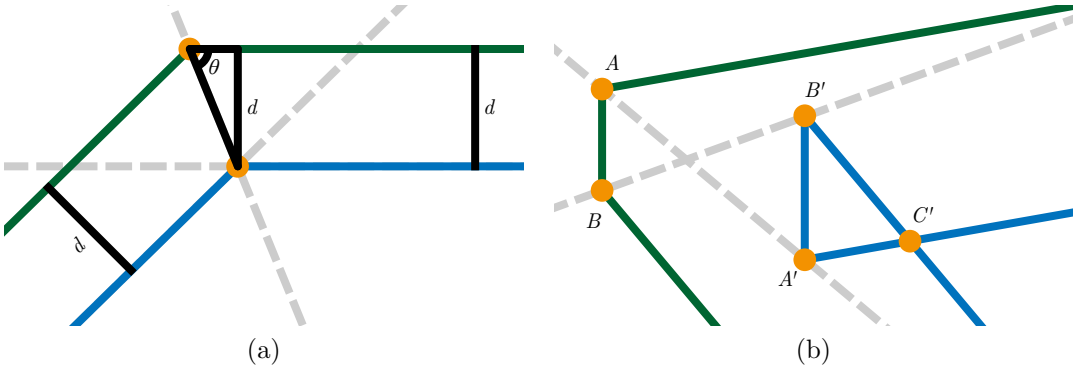


Figure 3.5: **Aspects of IB Voronoi mesh generation.** *(a) To shrink the original polygon (green lines) by a constant distance d, each angle must be bisected and the vertex repositioned a distance $d/\sin(\theta)$ along the line of bisection. (b) If edges are short compared to d, repositioning nodes could cause an inversion in the new polygon boundary. Here, A and B on the original polygon boundary move to A' and B', respectively, causing the new polygon to self-intersect at C'. In this case, the two new vertices A' and B' are merged into a single new vertex at C'.*

The paper by Sánchez-Gutiérrez and colleagues [142] demonstrates that two important summary statistics, namely PCD and cell area distribution, are successfully recapitulated for a number of biological scenarios, but further work is warranted to fully investigate the suitability of this method to produce realistic initial conditions. Nevertheless, this mesh generator is a good first step, and a marked improvement over the existing standard of regular hexagonal tiling.

Extension to IB cell populations This vertex mesh generation algorithm forms the basis of the IB equivalent. To turn a vertex mesh into an equivalent IB mesh requires two conceptual steps: (i) shrink each polygon to leave a fixed gap between elements; and (ii) discretise the boundary into a given number of evenly spaced locations.

An important difference between a VM representation of a tissue and an equivalent IB representation is that, in an IB representation, elements do not share edges: each IB is distinct from its neighbours. Given a convex polygon representing a Voronoi domain in a mesh generated as above, we must therefore shrink the polygon such that each new edge is parallel to, and a fixed distance d away from, the corresponding original edge.

How should each vertex move? To keep a vertex equidistant from two edges, it is clear that it must move along the line bisecting the angle between the edges. Furthermore, it should move a distance $d/\sin(\theta)$, where θ is the angle of bisection, as

seen in the schematic in Figure 3.5a.

Because, in general, the line of bisection through a vertex of a convex polygon does not go through the polygon's centroid, it is not guaranteed that the new shape will be simple. Figure 3.5b shows a diagram of how a self-intersection may occur. Given consecutive vertices A and B on the original polygon, if their distance is short with respect to d, the new polygon may self-intersect. In this case, A' and B' must be replaced by the point of self-intersection.

To avoid the complication of dealing with multiple intersections, we perform the polygon reduction in a number of steps, such that in each step the reduction distance d is no more than half the length of the shortest edge. The following algorithm details the procedure, which adds precision to the conceptual overview presented above.

Input:

- A Chaste vertex mesh object, as generated above
- A target spacing, $2d$, between cells
- A target spacing, s_t, between nodes in the discretisation

Output:

- A Chaste IB mesh object

Defined in:

- `VoronoiImmersedBoundaryMeshGenerator`

Algorithm:

1. `for each polygon in vertex mesh:`
    - Calculate $N = \lceil \frac{2d}{s} \rceil$, where s is the shortest edge length in the polygon
    - `for i = 1...N`
        - `for each vertex in polygon:`
            - Move vertex $d/N \sin(\theta)$ inward, with θ the angle of bisection at this vertex
        - Resolve any self-intersections
    - Calculate the number of nodes $M = \lceil \frac{p}{s_t} \rceil$, where p is the perimeter of the reduced polygon
    - Evenly space M nodes around the new polygon

Again, some implementation details have been omitted for brevity, and the full details can be found in the source code (Appendix A.2).

Together, these algorithms produce random realistic initial distributions of cell

shapes for use in modelling epithelial dynamics. They enable the use of these initial conditions in two explicit modelling frameworks, but it is clear that the same algorithm would easily extend to other modelling frameworks, such as the CPM, the SEM and the MVM. These implementations are directly relevant to work in this thesis, including work in Chapters 2, 4 and 5.

3.3.2 Fully periodic spatial decomposition algorithm

How can we keep track of which points in space are close to one another? To be more precise, given a number of points $\mathbf{x}_1, \ldots, \mathbf{x}_N \in \Omega \subset \mathbb{R}^n$, how can we produce a list of those points that are within a certain threshold distance, d, of other points? Our task is to produce such a list,

$$L = \{(\mathbf{x}_i, \mathbf{x}_j) \in \Omega \quad : \quad D(\mathbf{x}_i, \mathbf{x}_j) \leq d\}, \tag{3.2}$$

where $D(\cdot, \cdot)$ is a suitable distance function, assumed here to be simply the Euclidean distance.

Before addressing the problem of efficiently calculating such a list, we first ask why this is an important problem. The answer is that it is fundamental to the simulation of almost all off-lattice models. In all cases in which reactions occur between objects that are not connected by any other geometry such as a graph or lattice, the distance between points must be used to determine proximity for interactions. This is certainly the case for the IBM, in which cell–cell interactions are a key constituent component: determining the set of nodes that are involved in, say, cell–cell adhesion, is required at each time step. Because the IBM often involves a large number of discrete points per cell, it is of great importance that L can be calculated efficiently.

Spatial decomposition (box collection). A naïve method of calculation is to test every pairwise combination of points. This is a prohibitive $\mathcal{O}(N^2)$ task, for a potentially very large N and clearly we need a better way.

For a fixed maximum threshold distance d, a substantial improvement is to decompose Ω into d-sized boxes (hence the Chaste nomenclature of 'box collection'). Thinking, now, specifically of a square domain in $\mathbb{R}^2$ decomposed into smaller d-sized squares, given a point in a given square, interactions can only occur with points in the 3×3 grid of neighbouring squares. Thus, it is sufficient to assign each point to the box that contains it, and only check proximity to those points in neighbouring boxes.

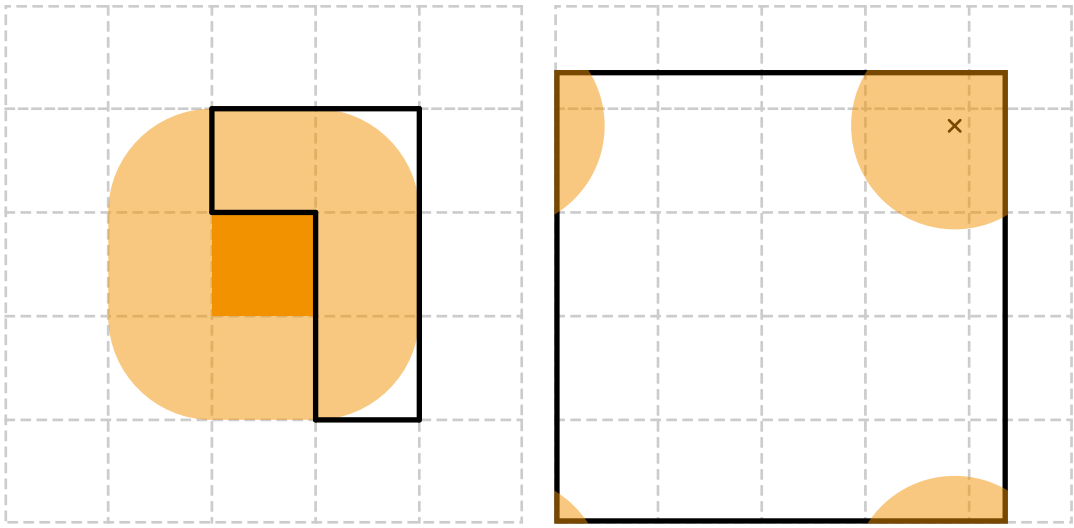


Figure 3.6: **Aspects of the spatial decomposition algorithm.** *(a) The dark orange square (centre) represents an arbitrary box of size $d \times d$ in the box collection (dashed grey lines). The lighter-orange rounded square surrounding it is the locus of all points within distance d of anywhere in the central box. The black inverted L-shape are those neighbouring boxes that must be checked to ensure all possible interactions are found. (b) The black square represents a domain covered by 5×5 boxes (dashed grey lines), where the box size does not perfectly divide the domain size. The black $\times$ in the penultimate row and column of boxes is within interaction distance of the orange shaded region which, due to the domain being periodic, is four disjoint regions.*

The algorithm can be further refined by noting that, if the node located at $\mathbf{x}_i$ interacts mechanically with the node located at $\mathbf{x}_j$, then the reverse is also true, and the reaction need only be identified in one direction. With reference to Figure 3.6a, this means that for each given box in the collection, only half the neighbours need be considered. There are multiple options for which neighbours to select, and not every selection of half of the neighbours is acceptable.

My specific contributions to the Chaste implementation have been to determine systematically which half of the neighbouring boxes should be selected, and to extend the algorithm to work with full periodicity. Both of these tasks are implemented in one, two, and three dimensions.

Selecting the correct neighbouring boxes. Each box in the domain needs to have a set of neighbouring boxes. When we iterate over all boxes in the domain, for each box we will check for interactions in all boxes that neighbour it. Therefore, we want the number of neighbouring boxes to be as small as possible with the proviso

that every interaction is identified. Interactions between two points within a single box are always, of course, possible, and so a box is always considered to be its own neighbour, but we are focussed now on selecting the appropriate other neighbours of the box.

In one spatial dimension there are two non-trivial neighbours: one to the left and one to the right. Let us label three consecutive boxes a, b and c. It is clear that, by selecting the neighbour as just the box to the right, every interaction involving any point within box b is caught: interactions between points in b and c will be caught when the neighbour of box b (box c) is searched, and interactions between points in a and b will be caught when the neighbour of box a (box b) is searched.

In two dimensions there are eight non-trivial neighbours, and in three dimensions there are 26. It is much less clear than in one dimension how to identify the appropriate neighbours for a given box, and I have developed a more systematic approach. First, we note that selecting at least half of the neighbouring boxes is necessary. Every adjacent box must be 'covered'. If we have selected B neighbours then every neighbouring box is certainly covered, and the box in question is a neighbour of each of B other boxes. At most $2B$ adjacent boxes are therefore covered, and so at least half the adjacent boxes must be neighbours. If there is no overlap between those adjacent boxes that are neighbours, and those that have the box in question as a neighbour, then the cover is exact and we have found a minimal and sufficient set of neighbours. The final insight is that, if a neighbour is chosen, then the box in question is a neighbour of the 'opposite' adjacent box: if the box to the right of a is a neighbour, then a is a neighbour of the box to the left and, if the box up-and-right is a neighbour of a, then a is a neighbour of the box down-and-left. Tables 3.1 and 3.2 show how this information can be encoded in two and three dimensions, respectively.

Four contact edges, in the following two pairs:		
`+x`	`-x`	`(1, 0)`
`+y`	`-y`	`(0, 1)`

Four contact vertices, in the following two pairs:		
`+x+y`	`-x-y`	`(1, 1)`
`+x-y`	`-x+y`	`(1,-1)`

Table 3.1: Box collection neighbour-pairs in two dimensions

Picking all adjacencies from the left-most column of Tables 3.1 and 3.2 as being those adjacent boxes selected as neighbours guarantees that all possible interactions

Six contact faces, in the following three pairs:		
`+x`	`-x`	`(1, 0, 0)`
`+y`	`-y`	`(0, 1, 0)`
`+z`	`-z`	`(0, 0, 1)`
12 contact edges, in the following six pairs:		
`+x+y`	`-x-y`	`(1, 1, 0)`
`+x+z`	`-x-z`	`(1, 0, 1)`
`+x-y`	`-x+y`	`(1,-1, 0)`
`+x-z`	`-x+z`	`(1, 0,-1)`
`+y+z`	`-y-z`	`(0, 1, 1)`
`+y-z`	`-y+z`	`(0, 1,-1)`
Eight contact vertices, in the following four pairs:		
`+x+y+z`	`-x-y-z`	`(1, 1, 1)`
`+x+y-z`	`-x-y+z`	`(1, 1,-1)`
`+x-y+z`	`-x+y-z`	`(1,-1, 1)`
`+x-y-z`	`-x+y+z`	`(1,-1,-1)`

Table 3.2: Box collection neighbour-pairs in three dimensions

will be identified, while ensuring that precisely half the adjacent boxes are chosen.

It is worth nothing that either of the adjacencies in each row could be selected, and that we choose an x then y then z ordering. This choice agrees with our choice in the one-dimensional case, and gives the inverted-L shape seen in Figure 3.6a for the two-dimensional case. This inverted-L corresponds to the left-hand column of Table 3.1 in that we have selected the four boxes to the right (`+x`), up (`+y`), right-up (`+x+y`) and right-down (`+x-y`) as being the neighbours of each box.

Accounting for periodicity. In principle, accounting for periodicity is straightforward. If the neighbour of a box on the edge of the domain is outside the domain, simply consider that neighbour to wrap around to the start of the domain in the appropriate dimension. In a perfect world, the size of a box would perfectly divide the size of the domain, and this would be the end of the story. The size of the box is simply the threshold distance for interactions, and so in practical terms it is never the case that the box size will divide the domain size. There will always be a small quantity of box 'left over' past the end of the domain, as illustrated in Figure 3.6b. The problem occurs when a point is in the penultimate box in any dimension and, specifically, when it is within one box-width of the edge of the domain. The $\times$ in Figure 3.6b illustrates this scenario. The locus of points within interaction distance

of the $\times$ includes three additional distinct regions, in boxes that are not adjacent to the box containing the point. Penultimate boxes in each dimension, therefore, need additional neighbours. The required additional neighbours are precisely those that are neighbours of the ultimate box in that dimension.

My work on this implementation can be found in the class `ObsoleteBoxCollection`. This class is named 'Obsolete' simply to distinguish it from the previously implemented `DistributedBoxCollection` [64] which extended the existing functionality to enable parallel computing. The `ObsoleteBoxCollection` is far from obsolete; it underpins the entire IB implementation.

Other methods of efficiently calculating pairwise interactions. When a pre-determined fixed interaction distance is known, and provided the points are distributed reasonably uniformly in space, the spatial decomposition described here is highly efficient. The approach described here is an example of a 'fast neighbour list', variations on which have been used for decades [67]. Due to the requirement of checking all neighbouring boxes, these methods are slightly conservative and improvements are possible. One such example is the 'Verlet list' [167] which reduces search time at the expense of an increased memory overhead.

In the case that no pre-determined interaction distance is known, methods for efficiently calculating pairwise interactions typically involve tree searches. Examples of such methods binary trees [67] and R-trees [61], and many advancements to these approaches continue to be made [4]. Such tree searches may also outperform neighbour list methods in the case of unevenly distributed points, even in the case of a fixed interaction distance. Neighbour searches are not prohibitive in current Chaste applications though, and therefore the relatively simple spatial decomposition described in this section is certainly adequate.

3.4 Pipeline for running simulations and presenting output

In this section, we describe a pipeline that I have developed for passing parameters to a simulation program (an 'executable') and processing the results from such simulations in such a way as to make them easily navigable in as portable a way as possible. This pipeline has been developed to efficiently utilise a Chaste executable to make use of multi-core computers, and automatically package results in such a way as to allow an experimental collaborator to interact with the output of parameter sweeps

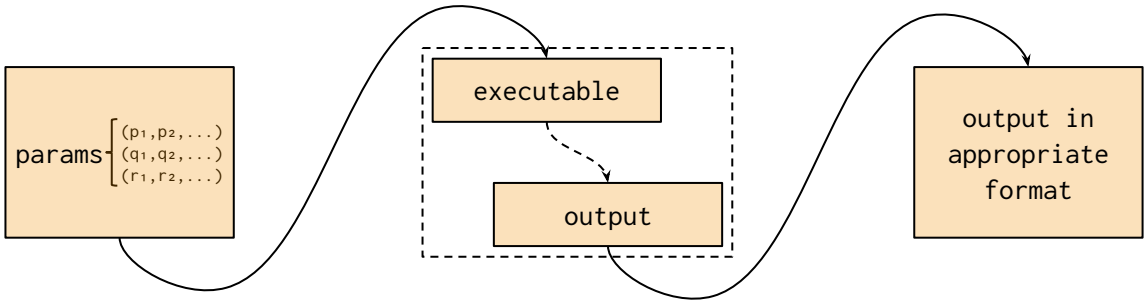


Figure 3.7: **Schematic of executable–output pipeline.** *Parameters sets are fed into an executable, which produces a certain output. This output is seldom in a convenient format, and so this output needs to be transformed into a more appropriate format. The solid black arrows represent stages in the pipeline that we have full control over.*

without needing specialist software. The work for which this pipeline has developed is presented in Chapter 5, but this approach is applicable in a wider sense and so is presented here in a general form.

To understand the specific need, we consider a single bespoke executable, such as a Chaste simulation, compiled from source code. This executable takes command line arguments, performs some computation, and records results in output files saved to disk. Considering cell-based simulations, these output files often contain some form of geometric information necessary to visualise the simulation, together with whichever statistics are required to summarise the results of the computation performed.

The diagram in Figure 3.7 shows a basic schematic of the pipeline, with a number of sets of parameters on the left hand side being fed into a 'black box' executable in which we have limited control over the output, followed by a post-execution step of converting the output into a more useful form. The two specific problems that this pipeline addresses are represented by the two black arrows in Figure 3.7, and are:

- how to efficiently allow the execution in parallel of many instantiations of a single-thread executable;
- how to allow the output from such parameter sweeps to be viewed by others, regardless of the operating system they are using, or the availability of any specialist software.

These issues are addressed in the following sections.

3.4.1 Infrastructure for parallel execution

It is often necessary to run a large number of simulations at once. Two examples where this is required are parameter sweeps, and to build up summary statistics from a large number of stochastic simulations. Parameter sweeps entail the varying of different parameters, and it is necessary to understand in what way each parameter influences the simulation output. Doing so necessitates running the simulation with many different sets of parameters, which simply means running the executable many times with different inputs. In addition, it is common to include noise in simulations, and it is often therefore necessary to run many instantiations of identical simulations in order to generate an average summary statistic.

These two scenarios are both examples of 'embarrassingly parallel' applications, a term coined by Moler [110] to refer to a workload where no cooperation is required between different processors in order to carry out the parallel computation. To accomplish the necessary infrastructure, we use the `Boost.Program_options`[4] library for parsing command line arguments within Chaste C++ applications, along with a custom Python wrapper using `multiprocessing`[5] to allow the distribution of simulations on multiple processors.

The `Boost.Program_options` library simplifies the passing of $(name, value)$ pairs on the command line to the executable. This produces an executable expecting parameters to be passed using the following syntax:

```
1 --VAR1 VAL1 --VAR2 VAL2 --VAR3 VAL3
```

A specific example of this can be found in the executable `PipelineExample.cpp` (Appendix A.2) which expects parameters `--ID`, `--FS`, `--RL` and `--TS` representing a unique simulation ID, the force strength, rest length, and number of time steps, respectively, for a toy example of an ellipse relaxing towards a circle.

Within the corresponding Python script `PipelineExample.py`, a dictionary defines the desired parameters:

```
1 command_line_args = {
2     'FS': {
3         'name': 'force_strength',
4         'vals': [1e5, 1e6, 1e7]
5     },
6     'RL': {
7         'name': 'rest_length',
```

[4] `http://boost.org/libs/program_options`
[5] `https://docs.python.org/2/library/multiprocessing.html`

```
        'vals': [0.1, 0.2, 0.3]
    },
    'TS': {
        'name': 'num_time_steps',
        'vals': [250]
    },
}
```

with the Cartesian product of all `'vals'` lists giving each combination of possible commands. Thus by simply filling in the required values in the `command_line_args` dictionary, the following set of commands are generated:

```
path/to/PipelineExample --ID 0 --RL 0.1 --FS 1000000.0 --TS 250
path/to/PipelineExample --ID 1 --RL 0.1 --FS 10000000.0 --TS 250
path/to/PipelineExample --ID 2 --RL 0.1 --FS 100000000.0 --TS 250
path/to/PipelineExample --ID 3 --RL 0.2 --FS 1000000.0 --TS 250
path/to/PipelineExample --ID 4 --RL 0.2 --FS 10000000.0 --TS 250
path/to/PipelineExample --ID 5 --RL 0.2 --FS 100000000.0 --TS 250
path/to/PipelineExample --ID 6 --RL 0.3 --FS 1000000.0 --TS 250
path/to/PipelineExample --ID 7 --RL 0.3 --FS 10000000.0 --TS 250
path/to/PipelineExample --ID 8 --RL 0.3 --FS 100000000.0 --TS 250
```

These commands are then distributed across multiple processes using the Python multiprocessing module, allowing this infrastructure to scalably make best use of the available computational resources, be it a laptop or a node on a supercomputer.

It is worth briefly mentioning two additional notes. First, the unique simulation ID assigned to each simulation can be conveniently used if a differing seed for a random number generator is required between simulations and, second, in order run an identical simulation a number of times, `'vals'` for any parameter can simply be a list of the same number n times, thus generating a list of identical simulations with differing IDs.

3.4.2 Presenting simulation output

A problem with bespoke software tools, such as Chaste, is that flexibility of the output format is often limited. The C++ programming language has good support for sending information to the console and to files, but there is (deliberately) no built-in support for graphical output such as images or videos. While libraries do exist with this kind of functionality, none could be considered canonical and, because usability is so closely related to the number of external dependencies a project relies upon, it is small wonder few scientific libraries use them.

Chaste has the default capability to output raw information to data tables which can later be interpreted and, optionally, to output geometric information to a niche file format produced by the software VTK[6]. This geometric information can be loaded into a software called ParaView[7] which, for an expert user, is an excellent tool for interacting with the information.

This, however, presents the main problem. In an ideal world the output from a simulation would be easily viewable by anyone with an interest in it. This can include, for instance, experimental collaborators with very little computational modelling expertise who are unlikely to have software such as ParaView installed, and who may not have the time or inclination to learn how to use it effectively.

What would be the ideal format for presenting the output of simulations to a wide audience? Is there a set of technologies that exist on all major operating systems, that would be useful summarising the output of a set of simulations? The answer we choose to build our solution around is the web browser.

Here, we present a pipeline for taking the output of a Chaste simulation and generating an HTML file consisting of a sortable data table displaying the parameters and summary statistics for each simulation, as well as hyperlinks to videos of the simulations viewable within the browser. Using this solution, once a set of simulations have been run, all that is necessary to share the output with anyone in the world is to share a directory with them where only a single `index.html` file need be interacted with.

An example of this solution can be seen in Figure 3.8. The left-most column displays hyperlinks to video representation of the simulation. The next four columns are the parameters passed to the executable, with non-varying parameters displayed in a lighter grey as a visual aid. The right-most column is the chosen summary statistic, here the ESF for the ellipse in the simulation.

The table is sortable, courtesy of the jQuery javascript library[8] and the tablesorter jQuery plugin[9], both of which are copied directly into the data directory in order to maximise compatibility by avoiding the need for an active internet connection to interact with the data. Having the table sortable allows easy identification of the parameter leading to most (or least) extreme summary statistics, and allowing easy navigation to the video corresponding to that extrema.

[6] `https://www.vtk.org/`
[7] `https://www.paraview.org/`
[8] `https://jquery.com/`
[9] `http://tablesorter.com/`

mp4 Name	Simulation Id	Rest Length	Force Strength	Num Time Steps	Esf
00.mp4	0	0.1	1000000.0	250	1.483337
01.mp4	1	0.1	10000000.0	250	1.359096
02.mp4	2	0.1	100000000.0	250	1.025988
03.mp4	3	0.2	1000000.0	250	1.485133
04.mp4	4	0.2	10000000.0	250	1.372507
05.mp4	5	0.2	100000000.0	250	1.035092
06.mp4	6	0.3	1000000.0	250	1.486935
07.mp4	7	0.3	10000000.0	250	1.386428
08.mp4	8	0.3	100000000.0	250	1.047715

Figure 3.8: **Example HTML index page.** *An example HTML file generated by the Python script* `PipelineExample.py`*, which invokes the executable compiled from* `PipelineExample.cpp`*. The main display is a sortable table displaying hyperlinks to videos of simulations, parameters, and summary statistics. Combining only HTML, javascript and mp4 video files, it is ultra-portable and viewable on almost any modern computer without need for any non-standard software.*

Producing the simulation videos. As already described, C++ does not lend itself well to the output of graphical information. First, we must identify the required format. The HTML5 standard supports only three video formats: MP4, WebM and Ogg. Of these, only MP4 is supported by all major web browsers [132]. Approximately 95% of all web browser use is accounted for by the top five most widely used browsers [5], all of which support MP4 playback, and therefore to ensure wide compatibility we must present videos in MP4 format.

How best to output data from a C++ executable and convert it to an MP4 video with minimal dependencies as far as C++ is concerned? The pipeline developed consists of the following steps, shown in Figure 3.9.

First, scalable vector graphic (SVG) files are written out from the C++ executable. SVG is an XML dialect for describing two-dimensional vector graphics [169], and constitutes a human-readable text-based markup language for drawing images. Primitives such as rectangles and circles are defined as in the following snippet, which

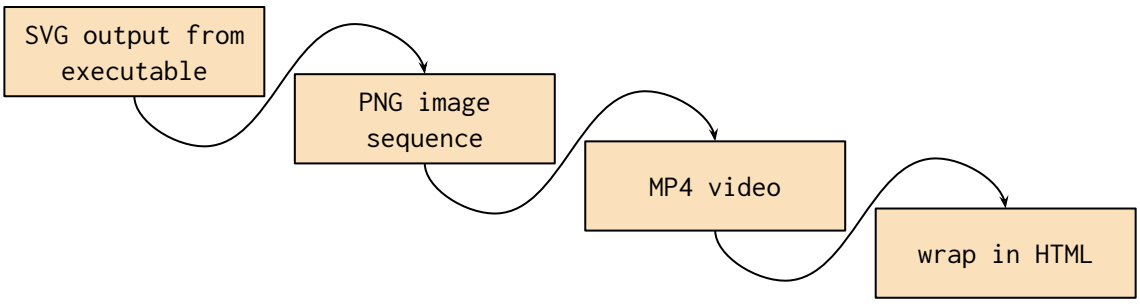


Figure 3.9: **Schematic of video generation pipeline.** *SVG files written out from the C++ executable are converted to a sequence of PNG files. This image sequence is converted to an MP4 video, and wrapped in HTML for inclusion in the index page.*

defines a 1920×1080 rectangle, on top of which is a circle centred at $(1120, 800)$ with a radius of 6.9.

```
<rect class="bg_rect" width="1920" height="1080"/>
<circle class="node" cx="1120" cy="800" r="6.9"/>
```

The 'class' argument allows CSS styling, such as setting fill colours, in the following manner:

```
<style type="text/css">
        .bg_rect{fill:darkgray;}
        .node{fill:#990000;}
</style>
```

Due to the simplicity with which one can programmatically construct customizable representations of simulation geometry in a universal format, the use of SVG output has the potential to be useful in a number of application areas. Here, functionality within Chaste is implemented in the class `ImmersedBoundarySvgWriter`, which writes image output at a customizable time sampling frequency.

The remainder of the process is handled within the custom Python script. SVG files are converted to a sequence of PNG files using the library CairoSVG[10], and this sequence is then stitched together into an MP4 video using FFmpeg[11].

In summary, the pipeline described in detail in this section is an automation built around a C++ executable. This automation efficiently utilises the available computational resources to run a set of simulations given a list of parameters. It results in a summary of the simulations presented in a cross-platform format viewable

[10] `http://cairosvg.org/`
[11] `https://www.ffmpeg.org/`

by anyone with a web browser. Once fed with the desired range of parameter values, the process is entirely automated.

While each individual step in the pipeline is conceptually straightforward, when taken together this solution vastly improves the efficiency and usefulness with which visualisations of simulations studies can be shared with collaborators and others. This is achieved mainly by tackling two problems: removing the dependency on any specialised software, and automating the process of converting simulation output to a cross-platform format.

3.5 Discussion and outlook

We began this chapter by arguing for the need to ensure code is written well, is available alongside published results, and is maintainable over a period of time. A specific example, that of Reinhart and Rogoff [133], was presented to frame the importance of ensuring high software quality within academia, and to highlight issues surrounding the lack of formal software engineering training for academics.

Section 3.2 details a number of important contributions I have made to Chaste infrastructure, which have increased the reliability and maintainability of the project, and move it into a position from which it can continue to evolve towards modern C++ and software engineering best practices. Section 3.3 details a number of specific contributions to Chaste that underpin other work in this thesis. These contributions underpin work on the IBM, already introduced in Chapter 2, and the comparison work and simulation studies presented in Chapters 4 and 5. Finally, Section 3.4 presents a pipeline for running executables and presenting simulation summaries to colleagues in a generic manner requiring no specific software to be installed.

This chapter has explored the rationale for the time and care needed when implementing academic software, and had highlighted a number of important contributions I have made to Chaste and its surrounding ecosystem. This work builds naturally on the IBM implementation presented in Chapter 2, as it is the infrastructure surrounding such software that lends it the credibility needed for the results generated by it to be trusted. Now that we have an IBM implementation, built on the solid foundation of Chaste, the next step is to gain better insight where the IBM fits into our toolkit of available computational models.

Chapter 4

Comparing individual-based models of cell surface mechanics

Our implementation of the IBM within Chaste (Chapter 2) enables the exploration of biological questions including those involving geometrically detailed descriptions of cell shapes. Subsequently (Chapter 3), we argued for the importance of research software engineering and the necessity of frameworks such as Chaste that emphasise the long-term sustainability and reliability of these implementations. However, we cannot unleash the IBM on a biological problem without verifying its suitability and assessing its relative merits compared to other common modelling approaches. The next step therefore is to gain a more thorough understanding of where the IBM fits into our toolkit of available cell-based models.

4.1 Introduction

We start by reviewing the literature on comparing cell-based models. Such models are being used to an ever greater extent, and the number of models and implementations is ever increasing (see [45] for an overview in the context of epithelial morphogenesis). As the number and range of available models increases, so too does the number and range of biological problems that can be sensibly addressed, provided we are able to select our tools appropriately. Given such models' complexity, it is typically not possible to make general statements about their properties based on rigorous mathematical analysis. Nevertheless, we can compare competing models within a consistent computational framework as a first step towards identifying and understanding artefacts associated with different models, different methods of numerical solution, and different implementations of the same model.

Very little work has been done in this area except by Osborne and colleagues [118], who exploited the range of models available in the Chaste library to simulate a number of benchmark problems with each model. This work substantially aids in understanding the applicability of each model, and guides potential future modelling applications. For example, the authors identified qualitative differences in the ability of each model to exhibit complete cell sorting (defined below). To date, no such work carefully compares the IBM, or any similar geometrically detailed cell-based modelling framework, to any competing framework. This chapter addresses this problem, improving our understanding of the IBM and the biological scenarios under which it can best be utilised. Our chosen strategy is to compare the IBM to a well characterised competing model, the VM, by applying both to simulate cell sorting as a benchmark model of tissue self-assembly.

4.1.1 Details of the VM

We first briefly summarise the VM, and fill in the remaining gaps in the description presented in Section 1.2.1. In the VM, cells are represented by polygons comprising a number of vertices. The motion of vertices is governed by a force law that assumes overdamped dynamics (Equation (1.3)), and the motion is supposed to be driven by the minimisation over time of a 'free energy' (Equation (1.4)).

As the vertices move, some cell edges inevitably shorten. Vertex models make choices of how to resolve particularly short edges, and one systematic description of possible vertex restructuring operations is presented by Fletcher and colleagues [46] who comprehensively describe the rules governing junctional remodelling in the Chaste VM implementation. The prototypical restructuring operation is the neighbour exchange, or T1 swap (Figure 4.1). A T1 swap is initiated when an interior edge reduces in size below a threshold, the 'cell rearrangement threshold' (d_{min}), in which case the short edge is replaced with a new edge perpendicular to it. The new edge is made longer by a factor of r_{cell}, the cell rearrangement ratio. For further discussion of the precise implementation of T1 swaps, and the stability of four-way vertices, see [151].

Taken together, the equations of motion, the free energy formulation, the method of numerical solution and the rules specifying the full range of restructuring operations define a VM implementation. How does this compare to the IBM? Several key differences are apparent. First, in the IBM, two neighbouring cells do not share a common edge, whereas in the VM they do. This explicit representation of cell boundaries in the IBM is key to its detailed representation of cell shapes, but no work

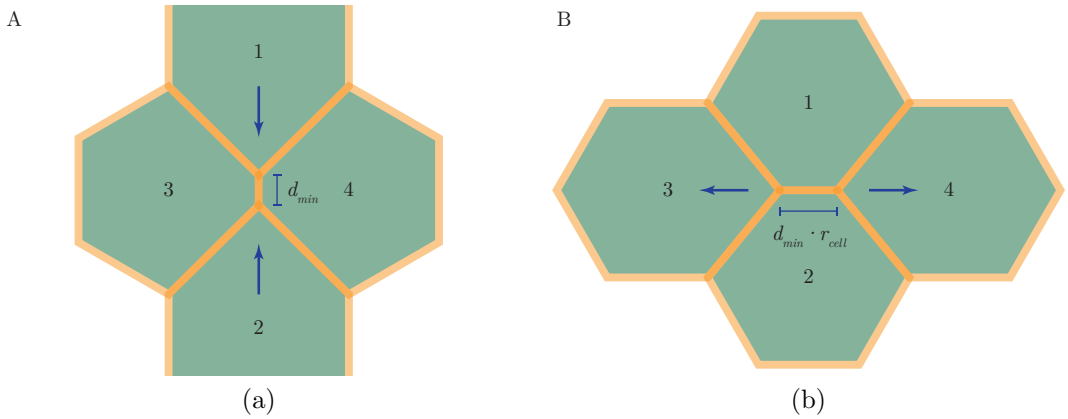


Figure 4.1: **A T1 swap in the Chaste VM implementation.** *(a) A short edge between cells 3 and 4 drops below the threshold d_{min}. (b) A new edge, of length $d_{min} \cdot r_{cell}$, replaces the short edge and separates cells 3 and 4.*

to date explores the impact that such an explicit representation has on simulations. Of particular interest is cell–cell adhesion: this is naturally incorporated into the IBM as a direct interaction between nodes in adjacent boundaries whereas, in the VM, adhesion is subsumed into the line tension term of Equation (1.4). Second, the rules of restructuring operations exist in the VM by necessity. To what extent do the choices made in the implementation of such rules impact simulations, and does the lack of such rules in the IBM constitute a relative advantage over the VM? Third, cell size is emergent in the VM but explicit in the IBM. These differences may well tailor the different models well to different biological scenarios. Given such a range of differences between the models, a careful comparison of them is particularly important in order to understand how key parameters might be related to biological quantities.

4.2 Extensions to existing IBM and VM descriptions

We next define a benchmark problem for simulating with both frameworks. Our chosen benchmark is a biologically inspired simulation of cell sorting, presented by Osborne and colleagues [118] (full details of which are presented in Section 4.3), which has been abstracted to remove extraneous details and allow us to focus on the essential behaviours and differences between the models. Before we begin exploring cell sorting as a model system to compare the two frameworks, we need the IBM to be capable of simulating epithelial dynamics. There are several fundamental processes that require implementation that are not part of the basic IBM as described in Chapter 2.

4.2.1 Cell neighbours in IBM simulations

In the VM it is unambiguous which cells are neighbours and which cells lie on any boundary of a tissue, as cell–cell interfaces are defined explicitly. In the IBM, things are not at all as clear, and knowing such properties is necessary when calculating summary statistics and implementing certain update rules. Consider a short edge about to undergo a T1 transition in the VM. Translating this geometry into an IBM simulation, using the algorithm described in Section 2.5.3 for generating an IB mesh from a vertex mesh will result in four IBs with gaps between them. A schematic of such a situation is shown in Figure 4.2a: it is far from clear by glancing at the geometries which pair of cells ought to be considered neighbours. How can we best convert the geometry of the IB nodes into an unambiguous description of cell neighbours?

The first approach one might think of is to tune a threshold distance and record all IBs within that distance as being neighbours. Inevitable fluctuations in the distance between IBs will, however, precludes this approach, which would be far from robust in the presence of short edges, and it certainly does not provide a unique characterisation of neighbours. A threshold distance would probably both over- and under-estimate the number of cell neighbours, for different cells and at different times in a single simulation.

We therefore propose a more sophisticated method for determining cell neighbours, inspired by the algorithm developed in Section 3.3.1 to generate an IB mesh from a vertex mesh. This algorithm shrinks a vertex element by a specific distance in such a manner as to generate a pre-defined constant gap between all generated IB elements. The reverse operation is to find the locus of points around the IB element that are closer to that element than any other element, i.e. the Voronoi domain of the IB element. As the IB element is comprised of a large number of nodes, to a good approximation the Voronoi domain of the entire element is the union of the Voronoi domains of every node in the element, which we define as the element's 'Voronoi superdomain'.

For a given geometry comprised of a number of IBs, Voronoi superdomains completely partition the simulation domain, and the boundaries between these Voronoi superdomains provide an unambiguous line defining cell neighbours. An example of this method is shown in Figure 4.2b. Four orange IBs, representing four nearby cells, are discretised by the blue nodes. Taking the union of the Voronoi domains for each blue node in each IB divides the square into four Voronoi superdomains, the boundaries of which are displayed as black lines. Using this method, the upper left and lower right cells may be considered neighbours, a fact

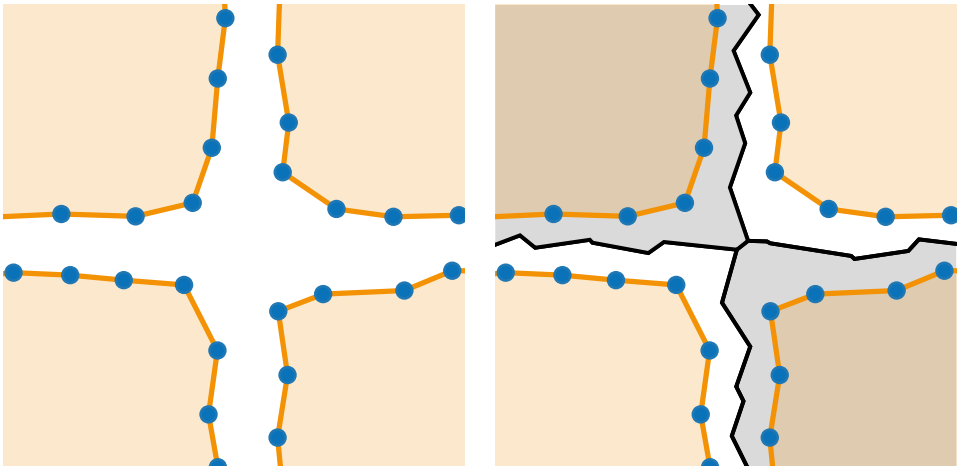


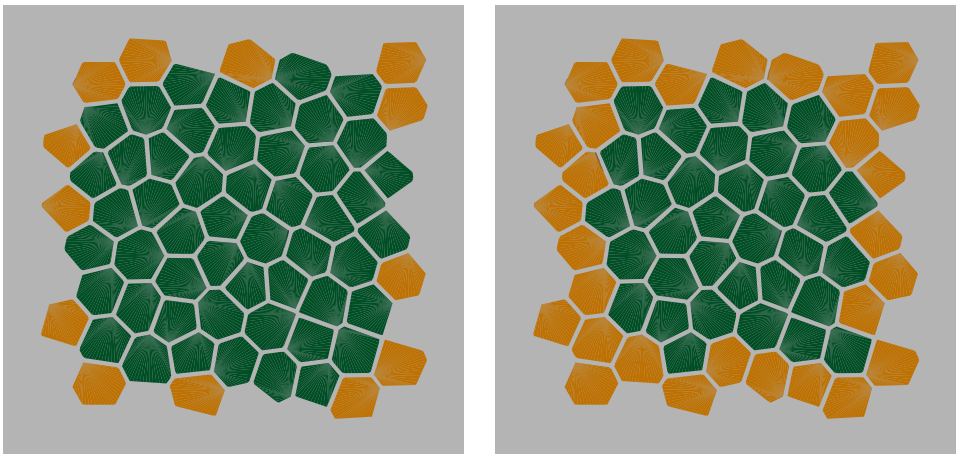
Figure 4.2: **Voronoi superdomains disambiguate cell neighbours.** *Blue points represent the discrete node locations on parts of four IBs (orange lines), representing four nearby cells. (a) It is not clear which pair of cells ought to be considered neighbours in this scenario. (b) Voronoi superdomains (black lines) superimposed over the schematic unambiguously categorise the upper-left and lower-right cells as neighbours.*

that is not clear at all in Figure 4.2a.

Given this unambiguous method for determining cell neighbours we can calculate the distribution of cell neighbour numbers (the PCD), a common measure of epithelial cell packing [42] and, indeed, it becomes straightforward given this total information about cell neighbours to determine the occurrences of T1 transitions in the IBM.

This approach also permits robust identification of which cells lie on the edge of an IB cell population. This is an important characterisation in a variety of circumstances: in the VM, nodes on the boundary must necessarily behave differently, for instance when involved in neighbour exchange events, and in both the VM and IBM, summary statistics such as the PCD must omit cells on the edge of the population. In the VM, nodes are unambiguously on the edge of the population, or not, and those cells on the edge of the population are exactly those containing edge-nodes. In the IBM, because cells do not share edges, no node is unambiguously on the 'edge' of the cell population.

The problem, in fact, reduces to that of identifying which of a set of randomly placed points in the plane are on the 'edge'. This can be explained well by the concept of an 'alpha-shape' [39]; consider rolling a ball of radius α around the edge of such a random set of points. Defining the points on the outside as those that the rolling ball touches, the number of points determined as being on the boundary will vary



(a) (b)

Figure 4.3: **Determining cells on the edge of an IB population.** *(a) Cells marked in orange have infinite Voronoi superdomains, and are certain to be on the edge of the population. (b) We iteratively include other cells whose Voronoi superdomain perimeters far exceed their actual perimeters.*

depending on α. Since this number varies with the radius of the ball, there is no unambiguous method for choosing which IB nodes lie on the 'edge' of a set of IBs.

We can, however, unambiguously determine a subset of elements on the boundary. Using the Voronoi diagram generated from every node, as described above, every infinite Voronoi domain will be on the convex hull of the set of points and so every infinite Voronoi superdomain is certain to correspond to a boundary element in the population (Figure 4.3a). Working iteratively from this subset, any neighbouring boundary elements will contain some length of 'free' edge and thus will have Voronoi superdomain perimeters much larger than their actual perimeter, and this can be used to determine which other cells are on the edge of the cell population (Figure 4.3b).

4.2.2 Allowing IB cells to modulate their size

In the VM, cells can change their surface area. Each cell has a target surface area, which feeds into Equation (1.4), allowing the cell area distribution to be an emergent property of the simulation. By contrast, the volume of fluid contained in an IB element is conserved, unless there are sources or sinks present within the IB.

To more faithfully compare the IBM with the VM we need to allow the surface area of IB elements to vary over time, with feedback from the geometry. In the VM, a cell under normal circumstances will be driven to increase its surface area up towards a defined target area, and this desire to expand is inhibited by the perimeter

contractility and line tension along cell edges. For a cell to shrink in surface area, it must be energetically favourable for its vertices to come closer together. This happens, in the VM, when line tension and perimeter contractility dominate the target area term in Equation (1.4), i.e., when the cell is squeezed, or crowded-out, by its neighbours.

To allow cells to modulate their size in the IBM, we seek to emulate this behaviour. There are numerous ways this could be achieved but, guided by parsimony, we choose the following *ad hoc* algorithm inspired by the VM case (see Section 4.4 for a discussion of this choice). As a measure for how crowded a cell is, we take the ratio of two perimeters: the length of the boundary itself (P_B) and the perimeter of the Voronoi superdomain corresponding to the boundary (P_V).

The closer P_V/P_B is to unity, the less freedom the cell has to expand, and we therefore assume it experiences a contractile pressure (down to a threshold minimum cell surface area). The larger P_V/P_B is, the freer the cell is, and we therefore assume it will inflate to fill the available space (up to a threshold maximum cell surface area). To implement this concept in the IBM, for each timestep of a simulation the ratio P_V/P_B is calculated for each cell. This generates a distribution of this ratio, which has a mean and standard deviation. The size of the deviation away from the mean crowding maps directly to the strength of the fluid source used to modulate the cell's surface area in the IBM, so that a cell experiencing exactly the mean crowding will not change in size, whereas the further from the mean crowding a cell is, the greater the magnitude of its associated fluid source. Full details of the implementation for this update rule can be found in the class `ImmersedBoundaryTargetAreaModifier`.

4.2.3 Adding noise to simulations

Biological processes are noisy. Sources of this noise include temperature and active biological processes such as growth. This noise is often intrinsic to the biological system; indeed, some processes such as neighbour exchange events seem to be driven by stochastic noise [30]. Because it is often impossible to identify the precise nature of all noise in any biological system, it is necessary to abstract this away and lump all noise into a random motion imposed on simulations.

In the case of VMs, the addition of such noise has previously been described by simple addition of Gaussian noise in the following manner: a random force $\mathbf{F}^{\text{random}}$ given by

$$\mathbf{F}_i^{\text{random}} = \sqrt{\frac{2\xi}{\Delta t}}\boldsymbol{\eta}^i, \tag{4.1}$$

is added to each node i in the simulation, at each timestep, where ξ controls the magnitude of the random perturbation, Δt is the length of each timestep in the simulation and $\left(\eta_x^i, \eta_y^i\right) =: \boldsymbol{\eta}^i$ are independent normally distributed random variates with zero mean and unit variance [46].

The time-dependence in this noise model formulation is introduced as a simple mechanism to temporally correlate noise independent of the size of the timestep, and is inspired by the motion of a particle undergoing diffusion. Under the dynamics described in Equation (1.3) (with η taken as 1), the i^{th} vertex with position $\mathbf{r}_i$ at time t under the sole action of $\mathbf{F}_i^{\text{random}}$, will move such that

$$\mathbf{r}_i(t + \Delta t) = \mathbf{r}_i(t) + \Delta t \mathbf{F}_i^{\text{random}} = \mathbf{r}_i(t) + \sqrt{2\xi\Delta t}\boldsymbol{\eta}^i, \tag{4.2}$$

which is the same expression as for the position of a random walker undergoing diffusion with diffusion constant ξ. The factor of $(\Delta t)^{-\frac{1}{2}}$ is included in the formulation of the force so as to ensure that the diffusive dynamics are independent of the choice of numerical timestep, and the force is constructed so as to incorporate a purely diffusive motion of each vertex.

This is the simplest implementation of 'noise' one could incorporate, but it is not at all clear that this choice is appropriate: why, for instance, should the random motion of two nearby vertices necessarily be independent of one another, particularly if those vertices represent junctions on the same cell? Before expanding on this point, let us consider adding a random force in the same manner to an IBM simulation. Consider an IB represented by N nodes with some node spacing h. The maximum motion a node could undergo without locally inverting the boundary is $h/2$ as, if the random perturbation strength on two adjacent nodes were perfectly opposite in direction, the motion would cross the nodes over, which is not permitted in the IBM. Any random perturbation, of strength ξ, capable of generating movements of length larger than $h/2$, therefore, cannot be permitted.

Consider the same IB this time represented by $2N$ nodes, with a node spacing of $h/2$. A perturbation of strength $\xi/2$ is now capable of locally inverting the boundary, so the maximum permissible strength of the perturbation is halved. This is undesirable: the magnitude of noise ought to be independent of the resolution at which we have chosen to represent the IBs. After all, the IB is simply a discretisation of a continuous membrane, and we ought to be able to add random perturbations of

a given magnitude to such a membrane in a manner that converges with h.

The previous method of simply adding independently sampled noise from a normal distribution to each node in the simulation is, therefore, inappropriate in the case of the IBM. This limitation also applies, though, to the VM. Consider a polygon with a short edge, which occurs frequently in motifs such as T1 transitions. The random forces added to the two nearby nodes could act in opposite directions, causing edge inversion and model degeneracy. Furthermore, two nearby nodes represent the locations of two material points that are also close together, and two nearby points on, for instance, the same cell membrane, clearly do not move independently of one another.

These factors point to the missing ingredient in the appropriate addition of random noise being a lengthscale over which it is correlated. We therefore seek a method to draw random noise from a normal distribution in such a way as to have spatial correlations on a given lengthscale: given M points, $\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_M \in \mathbb{R}^2$, we wish to draw an M-dimensional sample from a multivariate Gaussian distribution $N(\boldsymbol{\mu}, C)$, where $\boldsymbol{\mu} \in \mathbb{R}^M$ is a mean vector (usually, for this purpose, $\mathbf{0}$), and $C \in \mathbb{R}^{M\times M}$ is a covariance matrix representing how the random numbers should correlate with respect to each other.

Gaussian random fields. One such method is to generate instances of a discrete Gaussian random field (GRF). We first present formal mathematical definitions, before describing their usage and efficient computational implementation.

Consider a domain $D \subset \mathbb{R}^d$ and a probability space $(\Omega, \mathcal{F}, \mathbb{P})$. A random field Z is a set of random variables $Z(\mathbf{x}, \omega) := \{Z(\mathbf{x}) : \mathbf{x} \in D\}$, with the property that for each $\mathbf{x} \in D$, $Z(\mathbf{x}, \cdot) : \Omega \to \mathbb{R}$ is a random variable. For a fixed $\omega \in \Omega$, the deterministic function $f : D \to \mathbb{R}$ given by $f(\mathbf{x}) := Z(\mathbf{x}, \omega)$ is an instance of the random field Z.

Given a continuous random field Z and points $\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_M \in D$, we define a discrete random field $\mathbf{Z} := [Z(\mathbf{x}_1, \omega), Z(\mathbf{x}_2, \omega), \ldots, Z(\mathbf{x}_M, \omega)]^T$ with mean $\boldsymbol{\mu} = \mathbb{E}[\mathbf{Z}] \in \mathbb{R}^M$ and covariance $C = \mathbb{E}\left[(\mathbf{Z} - \boldsymbol{\mu})(\mathbf{Z} - \boldsymbol{\mu})^T\right] \in \mathbb{R}^{M\times M}$.

We call such a discrete random field *Gaussian* if $Z \sim N(\boldsymbol{\mu}, C)$, and an instance of it is simply a sample from the corresponding multivariate Gaussian distribution. How instances of such a field are generated depends, in general, on the form of the covariance matrix.

Choosing an appropriate covariance function. The covariance matrix C represents the extent to which the noise is correlated between points in the discrete

GRF and can be calculated by some covariance function C such that $c_{ij} = C(\mathbf{x}_i, \mathbf{x}_j)$. We prescribe a length scale $l > 0$ to control the distance over which noise will be correlated, and want this correlation to be some function of the distance between the points, implying that $C(\mathbf{x}_i, \mathbf{x}_j) = C(\mathbf{x}_j, \mathbf{x}_i)$. One suitable function is the Gaussian covariance function with unit variance, which takes the form

$$C(\mathbf{x}_i, \mathbf{x}_j) = \exp\left(-\frac{||\mathbf{x}_i - \mathbf{x}_j||_2}{l^2}\right), \tag{4.3}$$

where $||\cdot||_2$ is the vector 2-norm. Using this Gaussian covariance function C is real and symmetric; a fact that we shall use later.

Sampling from a GRF. There are several standard methods for sampling from GRFs and here we present a method based on spectral decomposition. Given points $\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_M$, we now have a known covariance matrix C. Let us now choose a mean $\boldsymbol{\mu} = \mathbf{0}$. How do we sample from $N(\mathbf{0}, C)$? First, draw M independent random numbers $\xi_i \sim N(0, 1)$, which can easily be done using many readily available computational tools. Then, $\boldsymbol{\xi} := [\xi_1, \xi_2, \ldots, \xi_M]^T \sim N(\mathbf{0}, I)$, where I is the $M \times M$ identity matrix.

Next, consider C. Because C is real and symmetric, by the spectral decomposition theorem, C can be factorised as $C = Q\Lambda Q^T$, where Λ is a diagonal matrix whose elements are the ordered eigenvalues $\lambda_1 \geq \lambda_2 \geq \ldots \geq \lambda_M \geq 0$ of C, and Q is an orthonormal matrix whose columns are the associated eigenvectors.

Finally, consider $\mathbf{Z} = Q\Lambda^{1/2}\boldsymbol{\xi}$. Then,

$$\mathbb{E}[\mathbf{Z}] = \mathbb{E}\left[Q\Lambda^{\frac{1}{2}}\boldsymbol{\xi}\right] = Q\Lambda^{\frac{1}{2}}\mathbb{E}[\boldsymbol{\xi}] = \mathbf{0}, \tag{4.4}$$

and

$$\mathbb{E}\left[\mathbf{Z}\mathbf{Z}^T\right] = \mathbb{E}\left[Q\Lambda^{\frac{1}{2}}\boldsymbol{\xi}\boldsymbol{\xi}^T\Lambda^{\frac{1}{2}}Q^T\right] = Q\Lambda^{\frac{1}{2}}\mathbb{E}\left[\boldsymbol{\xi}\boldsymbol{\xi}^T\right]\Lambda^{\frac{1}{2}}Q^T = Q\Lambda^{\frac{1}{2}}I\Lambda^{\frac{1}{2}}Q^T = C, \tag{4.5}$$

from which we conclude that $\mathbf{Z} = Q\Lambda^{1/2}\boldsymbol{\xi} \sim N(\mathbf{0}, C)$. Sampling the correlated noise we want, then, is a simple matter of applying the following algorithm:

Input:

- M points $\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_M$ at which we want to generate correlated noise
- A lengthscale l governing the distance over which the noise will be correlated

Output:

- $\mathbb{R}^M \ni \mathbf{Z} \sim N(\mathbf{0}, C)$, a field of random noise with given covariance

Algorithm:

1. Calculate the covariance matrix C using Equation (4.3)
2. Factorise C as $Q\Lambda Q^T$
3. Draw M random numbers ξ_i from a unit Gaussian distribution to form $\boldsymbol{\xi} \sim N(\mathbf{0}, I)$
4. Calculate $\mathbf{Z} = Q\Lambda^{1/2}\boldsymbol{\xi}$

Examples of GRF instances are shown in Figure 4.4. Each field was generated with the same 800 points $\mathbf{x}_i$ in the domain $[0, 20] \times [0, 20] \subset \mathbb{R}^2$. The points represent one instance of a randomly generated doubly periodic vertex mesh. Each row shows three instances of the GRF for a given lengthscale l where, here, l is the proportion of the domain width. As required, the undulations on the field occur on a lengthscale governed by the parameter l.

Efficiency. For the aforementioned algorithm for generating discrete GRFs, calculating the eigenvalues is an $\mathcal{O}(M^3)$ operation, thus scaling poorly with problem size. This all but ensures that GRFs cannot be calculated at every timestep of off-lattice simulations, as the calculation of random noise will dominate total simulation time. We therefore exploit the following additional factors that make this process fast enough to be applicable.

Regular grids. While the cost of calculating the eigenvalues of C is $\mathcal{O}(M^3)$, if C does not change between timesteps then this calculation is only required once. C does not change if and only if the points $\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_M$, and the lengthscale l, do not change. Treating l as a constant, per simulation, seems reasonable, and we can ensure each point $\mathbf{x}_i$ remains the same by fixing their locations, for instance by using a uniform grid.

For a constant C, the eigenvectors can simply be calculated once and reused as required. This turns the process from $\mathcal{O}(M^3)$ to $\mathcal{O}(M^2)$, which is still required every timestep in order to calculate the linear combination of eigenvectors. This approach can be applied in general to any off-lattice simulation framework, including the VM, by generating a suitably large uniform grid and performing an interpolation step between the off-lattice locations and the uniform grid on which the GRF is calculated. For the IBM, a uniform grid already exists for solving the fluid problem, and adding random noise to this grid removes the need for an interpolation step altogether.

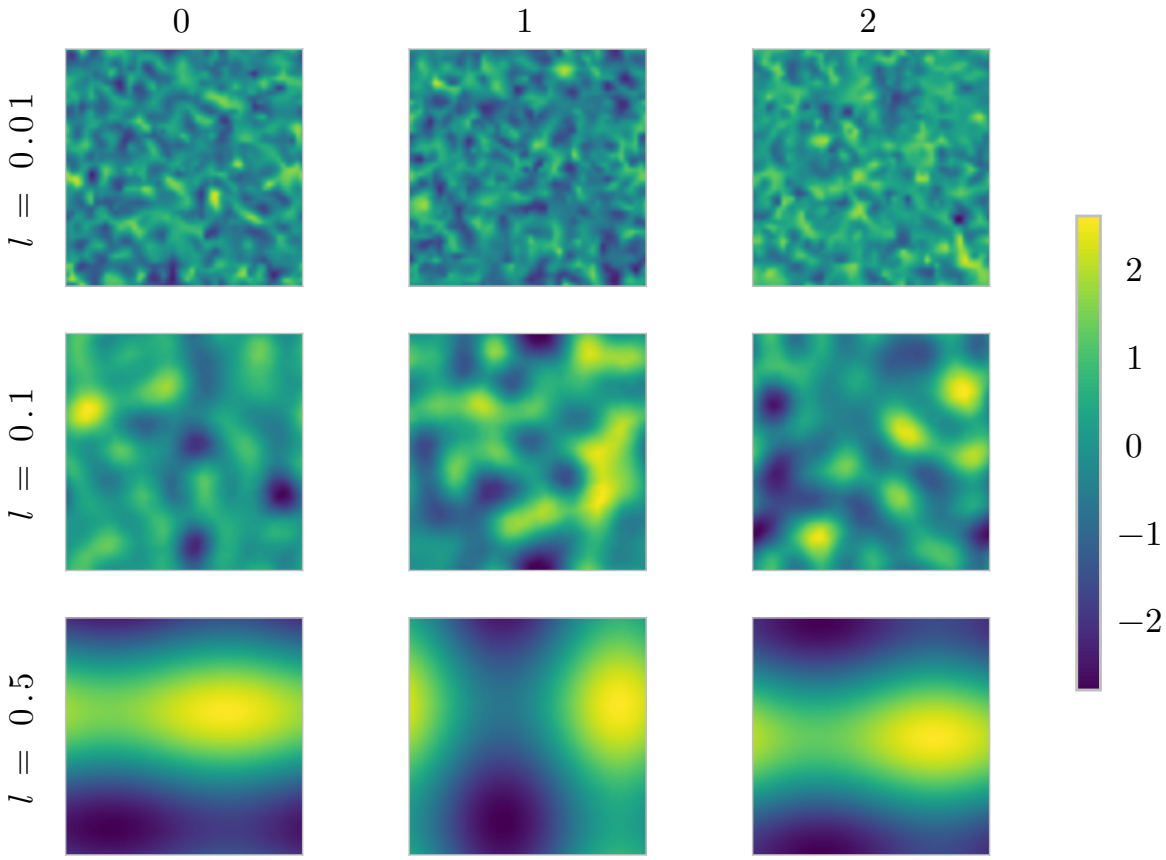


Figure 4.4: **Instantiation of GRFs.** *Nine GRF instantiations are shown for a doubly periodic square domain consisting of* 800 *randomly placed points. The field is interpolated away from these points for the purpose of visualisation. The lengthscales, l, are in proportion to the width of the domain, and three instantiations are shown for each value of l.*

Using uniform grids is the single largest saving in terms of computational cost, but does have one drawback. Creating a uniform grid of fine-enough resolution to allow suitable interpolation necessitates using many more points $\mathbf{x}_i$ than the original number of points in the off-lattice simulation. Thus, the M in the new $\mathcal{O}(M^2)$ problem is much larger than in the old $\mathcal{O}(M^3)$ problem, a drawback that we now tackle.

Partial eigenvalue decomposition. The concept of a partial eigenvalue decomposition for the approximate sampling of GRFs is existing work. To my knowledge, however, the work presented in this section is the first to quantify the relationship between the trace proportion and the closeness of the approximation, and the first to quantify the number of eigenvalues necessary given a specific lengthscale. Conceptually, the algorithm described above starts with a single sample

with covariance I, and reshapes this covariance to match C. This reshaping process is a linear transformation of the original vector of individual samples: the eigenvector transformations are rotations adding correlation to the points, and the eigenvalues apply the necessary scaling, weighting the eigenvector contributions by importance. The resultant discrete GRF can equivalently be written as

$$\mathbf{Z} = Q\Lambda^{1/2}\boldsymbol{\xi} = \sum_{i=1}^{M} \sqrt{\lambda_i} Q_i \xi_i, \tag{4.6}$$

where Q_i is the i^{th} column of Q, and this is just a linear combination of the eigenvectors.

One option for reducing the computational cost of instantiating this GRF is to simply truncate this sum, ignoring some quantity of eigenvectors with the smallest eigenvalues. We know that $M = \text{Tr}(C) = \text{Tr}(Q\Lambda Q^T) = \text{Tr}(\Lambda) = \sum_i(\lambda_i)$. We also know that the eigenvalues tell us about the weight of each eigenvector when reshaping the covariance matrix. What if we calculate eigenvalues until their sum exceeds some fixed proportion of the trace of the matrix? To investigate this, we can generate GRFs for different length scales and truncate the sum above once the calculated eigenvalues exceed a given proportions of the trace. For each field, we can generate many instances and, for each of the samples generated, plot a histogram to see whether they are indeed normally distributed.

Figure 4.5 summarises this investigation. Each column shows the truncation at different proportions of the trace. Looking at the first column, including eigenvalues up to just 50% of the trace produces very poorly distributed samples compared to the unit Gaussian distribution, for a wide range of different lengthscales l. Increasing the trace proportion to 80% vastly improves the distribution, but across all length scales the distribution is still slightly taller and narrower than the unit Gaussian distribution. By 95% of the trace proportion, the samples fit the unit Gaussian distribution very well.

The performance impact of truncating the eigenvalue sum depends significantly on the length scale. Figure 4.6 shows this dependence for nine distinct length scales (1%, 5% and 9% highlighted), for trace proportions between 60% and 98%. As can be seen, at $l = 9\%$ we need only calculate $\sim 20\%$ of the eigenvalues to achieve the 95% trace proportion (corresponding to a roughly fivefold decrease in computation time), whereas at $l = 1\%$ we require well over 80% of the eigenvalues, giving only a modest (but still useful) 20% speed up.

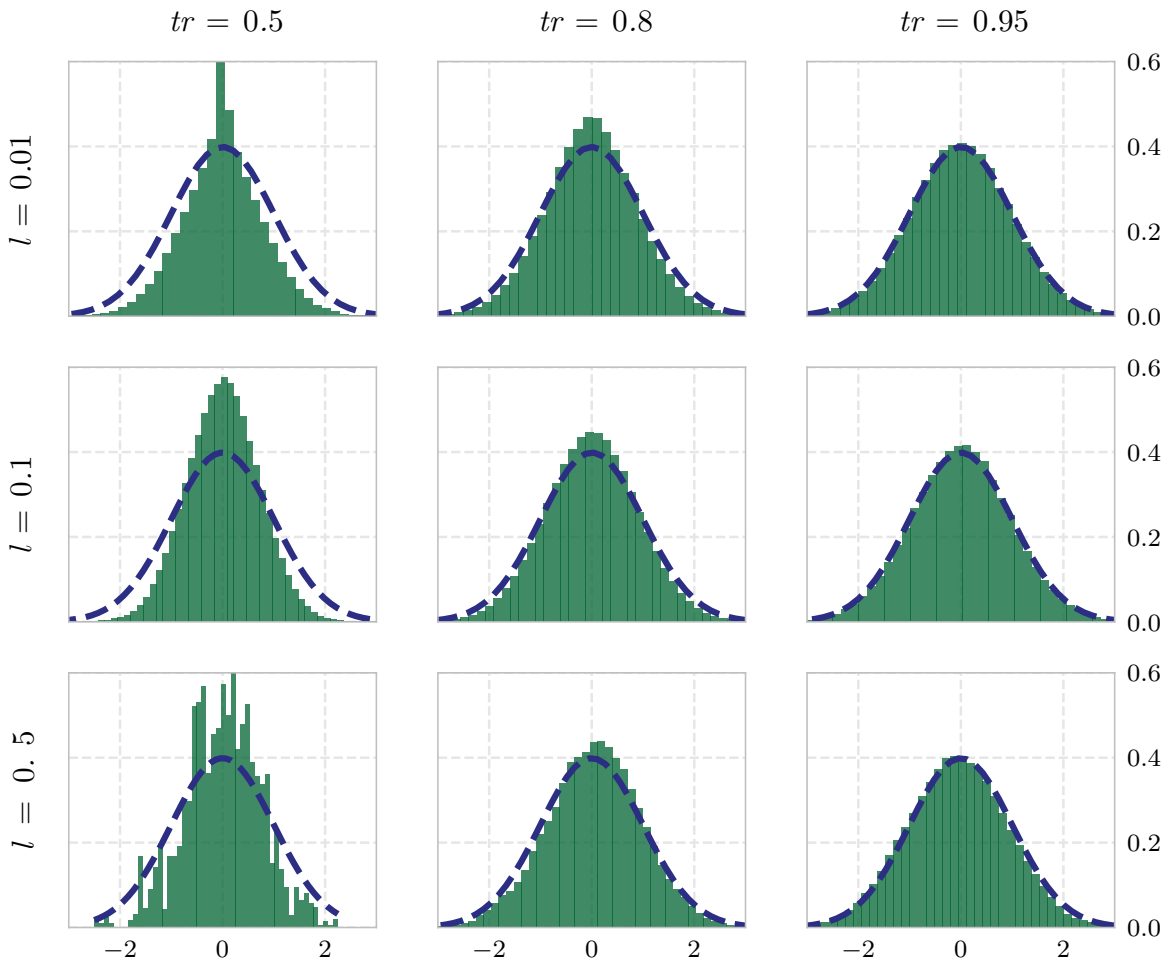


Figure 4.5: **Effect on field distribution of varying trace proportion.** *Sampling distributions are shown for a randomly generated doubly periodic vertex mesh containing* 800 *nodes. Each column shows a trace proportion, tr, and each row shows a correlation length, l. Each histogram represents* 500 *instances sampled from the GRF (*$800 \times 500 = 400000$ *individual random numbers in total), where the number of eigenvalues calculated is precisely chosen to ensure their sum minimally exceeds the proportion t of the trace of the covariance matrix. The dotted lines show the unit normal distribution,* $N(0, 1)$*. As can be seen, using too few eigenvalues from the covariance matrix results in a non-normal noise distribution. The value* $tr \approx 0.95$ *appears appropriate to faithfully reproduce normally distributed noise.*

We now have a strategy for calculating appropriately correlated random noise for simulations. We recall that this is vital for IBM simulations, for which uncorrelated noise simply does not make sense, but that in finding a solution to this problem, we see that the same problem is apparent in other off-lattice simulations as well, albeit hidden by the fact that nodes are seldom spaced close enough together to cause model degeneracy.

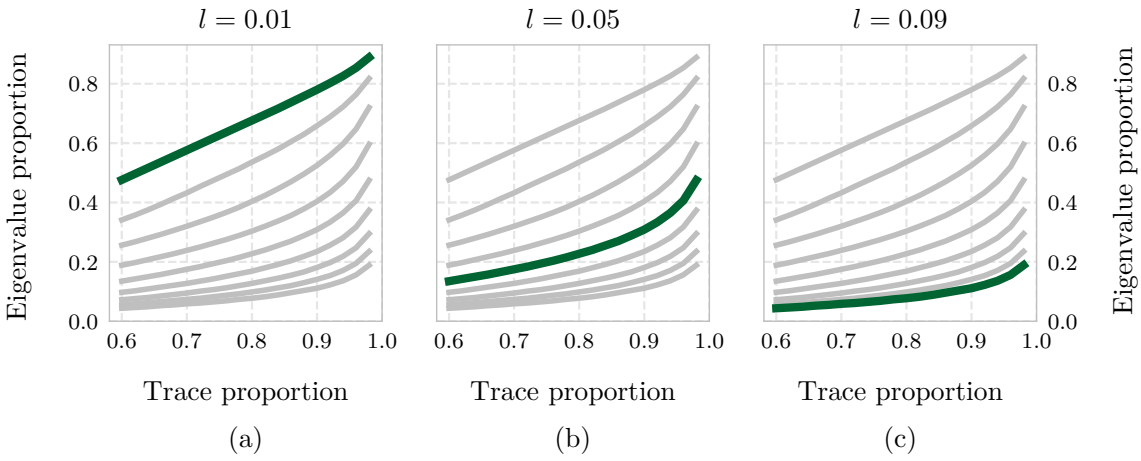


Figure 4.6: **Effect of correlation length on the proportion of eigenvalues needed to exceed a given trace proportion.** *As the correlation length proportion, l, reduces, the proportion of eigenvalues that must be calculated in order to achieve a given trace proportion, t, increases.*

4.2.3.1 Computational implementation

Full details of how these algorithms are implemented can be found in the class `OffLatticeRandomFieldGenerator` (details on how to obtain the source code can be found in Appendix A.3).

Factorising the covariance matrix. The matrix representation of C is stored using `Eigen`[1], a C++ library for linear algebra computation. Factorising C is performed by the `Spectra` C++ library[2] which is built on `Eigen` and is capable of sparse eigenvalue computation. These libraries exploit the sparseness of the covariance matrix as well as highly optimised algorithms.

Constructing the linear combination of eigenvectors. The library `Eigen` is, again, used here as it offers highly optimised linear algebra computation. This is of most relevance when using GRFs on a regular grid, as the one-off $\mathcal{O}(M^3)$ operation no longer dominates the time taken to compose the GRF, but the size of M is large and efficient linear combination of eigenvectors is key.

Taken together, the cell neighbour identification, the modulation of cell size and the implementation of random noise develop the IBM to a point where it is capable of performing the same types of computational experiment as the VM. The ability

[1] `http://eigen.tuxfamily.org/`
[2] `https://spectralib.org/`

to robustly identify cell neighbours allows direct comparison of processes such as T1 transitions and statistics such as neighbour edge lengths. A mechanism to alter the size of cells in simulations is also important. Finally, finding an appropriate method for adding noise to the IBM has led to a new and more general mechanism for adding random noise to any off-lattice models, and having a robust method for adding noise to the IBM is certainly a necessary step for adequately comparing the frameworks.

We are now in a position to directly compare the two methods on a particular biological problem, with the aim of understanding better the strengths and weaknesses of the IBM.

4.3 Cell sorting as a model system for comparison

Cell–cell motion is fundamental to a variety of developmental processes and includes a wide range of cellular rearrangements and sorting. Embryonic tissues consisting of cells of different histological types can mix, form checkerboard patterns and sort, and there are wide variety of biological examples and hypotheses for such behaviour, many of which are summarised in an review by Brodland [19].

Cell sorting describes the phenomenon of a cell population segregating into two or more distinct regions, and observed behaviours include the rounding up and merging of individual cell types, and the complete engulfment of one cell type by another. Experimental work on such phenomena date back to the 1700s with work on sponges, but systematic studies were undertaken in the 1960s [19]. Thereafter, two leading theories were developed that explain cell sorting: the differential adhesion hypothesis [154, 155] and the differential interfacial tension hypothesis [18] and its precursors [63].

In the differential adhesion hypothesis, cell sorting is driven by differences in the affinity of cells for adhering to one another. This hypothesis posits that a given cell type has a higher adhesive affinity for cells of its own type than for cells of a different type, and that therefore cell types will coalesce together, minimising their interaction boundary with differing cell types, leading to sorted cell regions. Experimental work by Steinberg [154] demonstrated that heterogeneous populations of cells evolved towards specific steady states regardless of the initial geometric configuration of cells, and furthermore that engulfment of one cell type by another occurred hierarchically in the presence of multiple cell types and in accordance with their relative affinities for one another.

Simulation studies have been effective at elucidating the mechanisms of cell sorting

under the differential adhesion hypothesis, and one such example is by Zhang and colleagues [174]. Brodland reviews other cell sorting simulation studies, many of which have been successful in advancing the understanding of cell sorting Brodland [19].

Increasingly, there is growing recognition of the need to carefully compare established modelling frameworks on benchmark problems in order to understand artefacts associated with different models, different methods of numerical solution, and different implementations of the same model. Given the biological relevance and the existence of previous simulation studies, cell sorting due to differential adhesion is an excellent candidate for one such benchmark, and this is a benchmark chosen by Osborne and colleagues [118], one of the few studies to date that addresses the need for such careful comparisons.

To date, no comparison has been made between the IBM and any competing frameworks, and we address this by building on the work of Osborne and colleagues. We first recapitulate their cell sorting results in the case of the VM. We then add random noise in this context to demonstrate the impact of a GRF noise implementation on the existing model. Finally, we simulate cell sorting in the IBM and, by comparing these results to the VM case, learn more about the strengths and limitations of the IBM as a tool for cell-based simulation.

Details of the previous simulation study. Osborne and colleagues simulated cell sorting due to differential adhesion in a simplified abstraction that removes extraneous details and focusses on the essential behaviours of the computational models being compared. They simulate a population comprising two cell types A and B in a monolayer that does not undergo any proliferation or respecification. The interaction energy between two neighbouring cells is governed by a term γ, the interaction energy between cells (corresponding to the line tension Λ defined in Section 1.2.1), such that $\gamma(A, A) = \gamma(B, B) < \gamma(A, B)$. This encodes a 'preference' for two cells of different types to minimise their interaction length. In addition to this inequality, the authors choose $\gamma(A, void) < \gamma(B, void)$ so that cells of type A will preferentially exist on the boundary of the simulated monolayer, resulting in the engulfment of cell type B by cell type A.

In addition to the update rules for the VM described in the introduction to this chapter, the 'basic' random force (see Section 4.2.3) is applied to each node in the simulation: this random force facilitates a significant increase in the T1 transitions required for cell sorting. The diffusion strength, ξ, modulates the quantity of perturbation added to the simulation.

A quantitative comparison between different models is made possible by computing the evolution over time of the 'fractional length' (based on the 'heterotypic boundary length' introduced by Zhang and colleagues [174]), defined as the total length of the boundary between cells of different types, normalised by the length of that boundary at time zero. Formally, the heterotypic boundary length at time t is defined as

$$\mathrm{HBL}(t) = \sum_{b \in \mathcal{B}(t)} l(b)(1 - \delta_{AB}), \tag{4.7}$$

where $\mathcal{B}(t)$ is the collection of all cell-cell boundaries in the configuration of the cell population at time t, l is the length of a given boundary, and A and B are the identities of the two cells forming the boundary. The fractional length, then, is given at time t by $\mathrm{HBL}(t)/\,\mathrm{HBL}(0)$. This measure therefore takes the value 1 initially and decreases as cells sort and reduce the total length of heterotypic cell–cell interfaces.

4.3.1 Recapitulating cell sorting in the VM

To facilitate comparison between the VM and the IBM, we make one significant change over the previous work by Osborne and colleagues by simulating on a doubly periodic domain. This prevents cells in the IBM from simply pushing apart during simulations (which prevents cell sorting), and this is discussed in Section 4.4. The simulation setup is as follows. We position 100 cells in a square, on a doubly periodic domain, using the Voronoi mesh generation method described in Section 3.3.1. Before randomly selecting cells to be labelled as cell type B, we run the simulation for a number of timesteps to allow the tissue to achieve mechanical equilibrium. Due to the random noise imposed on vertices in the simulation, there is no definitive notion of a mechanical equilibrium, and so this is achieved in an *ad hoc* manner by simulating the monolayer long enough for there to be little change over time. By utilising the Voronoi mesh generation method, this burn-in time is negligible, and we choose 1% of the overall simulation time rather than the 10% used by Osborne and colleagues.

Once close to mechanical equilibrium, we randomly label half of the cells as type B, where heterotypic interactions have the property that $\gamma(A, B) = 2\gamma(A, A)$. The simulation then proceeds for a fixed simulation duration and the fractional length is recorded over time. Figure 4.7 shows snapshots of a simulation with complete sorting by $t = 150$. All parameters are as published in [118], with a diffusion strength equal to 0.5, and full details are available in the source code (Appendix A.3). There are two observations in these snapshots that are worth highlighting. First, in all four images there are examples of non-convex cells. This is due to the imposition of a random

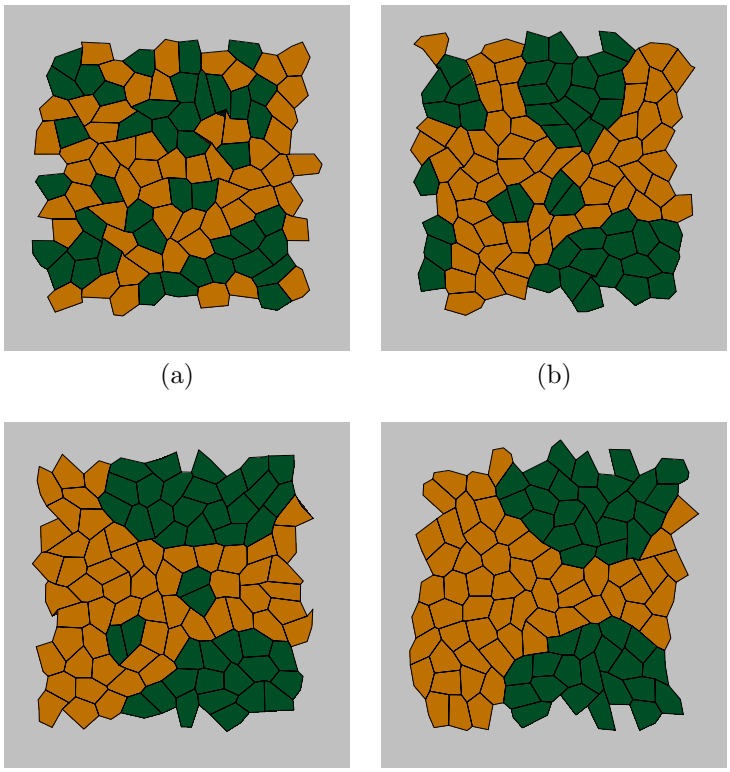


Figure 4.7: **Snapshot of VM cell sorting on periodic domain.** *Snapshots of a simulated monolayer on a periodic domain are shown at four selected time points. Two cell types A and B are shown in green and orange, respectively. Over time the two cell types fully segregate until cells of type A are engulfed by cells of type B. (a) Snapshot at* $t = 0$*. (b) Snapshot at* $t = 50$*. (c) Snapshot at* $t = 100$*. (d) Snapshot at* $t = 150$*.*

force on each vertex, allowing vertices to move in a volatile manner, and so the normal convexity observed during VM simulations is not present in these simulations. Second, there are black triangles visible in the interior of the tissue, particularly in the first of the four snapshots. In order to achieve a high measure of cell sorting, the value of the cell rearrangement threshold chosen by Osborne and colleagues was tenfold higher than the typical default value for VM implementations (and the default value in Chaste). For this reason, where there are several short edges in close vicinity and one is involved in a rearrangement such as a T1 transition, the VM implementation is unable to fully cope and a partial inversion of the polygon occurs. While the VM implementation is robust to this, it is worth highlighting that a substantially higher threshold than the default was used by Osborne and colleagues. In the next section, the choice of this threshold is explored further.

4.3.2 Extending the understanding of VM cell sorting

In order to understand cell sorting in the VM in more detail, we vary the aforementioned parameters to determine the relative effects of a change in the diffusion strength and in the cell rearrangement threshold. We begin by performing the following computational experiment. For two values of the cell rearrangement threshold (0.1, used by Osborne and colleagues, and 0.01), we run a number of simulations at differing values of the diffusion strength. The diffusion strength is varied from 0.1 to 0.9 in steps of 0.2, and for each value we average the summary statistics over 20 simulation runs. Figure 4.8 summarises the results of this experiment.

First, focussing on the results for the higher rearrangement threshold (Figure 4.8a), the green highlighted line (a diffusion strength of 0.9) compared to the orange highlighted line (a diffusion strength of 0.1) demonstrates that the diffusion strength plays a role in cell sorting in this VM regime. Broadly speaking, the larger the diffusion strength, the lower the resulting fractional length. This is to be expected, as allowing the vertices in the VM to move a greater distance through random perturbation allows rearrangements that are, in the absence of strong random fluctuation, very energetically unfavourable. The higher the level of random noise, the greater the likelihood of a vertex breaking out of a local energy minimum and causing a swap that ends up in a lower-energy global state.

The picture is not the same, however, for the default cell rearrangement threshold (Figure 4.8b). In this case we see immediately that the quantity of cell sorting is significantly reduced, and that there is no eventual difference in the quantity of cell sorting over the entire range of diffusion strengths. Indeed, the eventual trajectory of all diffusion strength values is near identical. The reason for this is more subtle. Cell sorting in the VM relies upon random motion driving vertices to within the cell rearrangement threshold. If an edge is similar in length to the rearrangement threshold, it is clear that random perturbation of vertices is easily capable of triggering a rearrangement, and indeed that the likelihood of triggering such a rearrangement will be proportional to the strength of the random perturbation. If, however, the edge is long in comparison to the rearrangement threshold, it is unlikely that the random perturbations will bring the two vertices together. Moreover, if the diffusion strength is large enough to make such an occurrence likely, the 'rearrangement zone' would be very small in comparison to the absolute vertex motion, and overshoot or large unpredictable position jumps would result in a high probability of the VM becoming degenerate. For this reason, there is little difference based on the diffusion strength

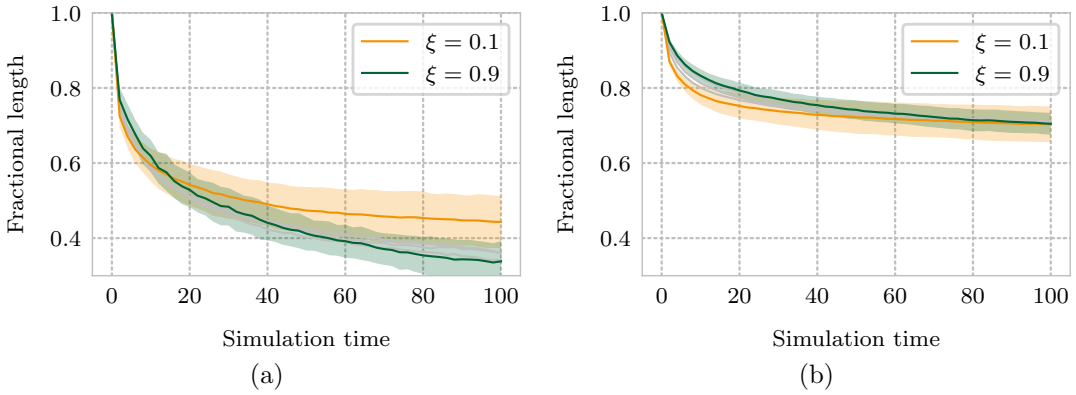


Figure 4.8: **Effect of cell rearrangement threshold and diffusion strength on VM cell sorting.** *(a) Fractional length over time for multiple simulations with a cell rearrangement threshold of 0.1, the value used by Osborne and colleagues. The lines represent the average of 20 runs with different values of ξ, the diffusion strength, with the orange highlighted line $\xi = 0.1$ and the green highlighted line $\xi = 0.9$. Grey lines are intermediate values between the two highlighted extremes. Green and orange highlighted regions denote a standard deviation either side of the mean for the two highlighted values. (b) Fractional length over time for multiple simulation with a cell rearrangement threshold of 0.01, the default value used for VM simulations in Chaste. The lines represent the average of 20 runs with different values of ξ, the diffusion strength, with the orange highlighted line $\xi = 0.1$ and the green highlighted line $\xi = 0.9$. Grey lines are intermediate values between the two highlighted extremes. Green and orange highlighted regions denote a standard deviation either side of the mean for the two highlighted values.*

and, indeed, running simulations with significantly larger diffusion strengths does lead to model breakdown.

In summary, in order to obtain robust cell sorting in the VM, the cell rearrangement threshold must be large enough so as to allow random perturbation to frequently trigger cell rearrangement events. In the absence of random perturbations, or in the absence of a sufficiently large rearrangement threshold, there is an insufficient quantity of rearrangement events for the VM to adequately escape the local minima of its energy function. This introduces a fundamental limitation on the VM for investigating differential adhesion. Not only is a random force required, but the choice of model-specific parameters must also be made carefully. Because the cell rearrangement threshold does not have any direct biological correlate (it is simply a computational necessity as, due to the discrete timestepping, edge lengths do not smoothly approach zero), any choice for this parameter is *ad hoc*. As a future avenue for further investigation, varying this parameter, together with the diffusion strength, may allow a more appropriate calibration of these parameters with biological data. That is, however, beyond the scope of this work, which is primarily focussed with

understanding cell sorting in the VM as a point of comparison with which to better understand the utility of the IBM.

4.3.3 Adding correlated noise to VM simulations

A further avenue for exploration is the addition of length-correlated noise to VM simulations. As detailed in Section 4.2.3, the impossibility of applying random perturbations to IBM simulations in the same manner as had previously been applied to VM simulations led to the development of a GRF technique for adding spatially correlated noise to arbitrary domains. Given that, in VM simulations, it is expected that vertices come within very small distances of one another (for instance, during a T1 transition) it is unclear that adding uncorrelated random noise is valid. Spatial correlation is therefore appropriate to consider in the VM case as well as in IBM simulations, and in this section we explore the extent to which adding spatially correlated noise to VM simulations affects summary statistics.

We make use of the uniform grid and interpolation method for adding random noise, so must ensure the lattice spacing is small in comparison to the average distance between vertices in the simulation. In this simulation study we use a conservative lattice spacing in order to reduce the likelihood of lattice spacing being a material concern in the interpretation of results. Given a square domain with approximately 200 evenly spaced vertices, one would expect approximately $\sqrt{200} \times \sqrt{200} \approx 14 \times 14$ vertices in each dimension, so choosing 64×64 lattice sites ought to give ample resolution. As an aside, operating on a doubly periodic domain in this instance is also a benefit as the underlying lattice will always perfectly fit the simulation domain.

To understand the impact on VM cell sorting simulations of altering the length scale over which random noise is correlated, we perform the following computational experiment. For fixed diffusion strength (1.0) and fixed cell rearrangement threshold (0.05), we perform five simulations runs for each of a variety of correlation length scales. The target surface area for cells in this experiment is 1.0, and the average surface area of elements is $0.5\sqrt{3} \approx 0.9$, so 1.0 is a rough 'lengthscale' of an individual cell. In order to encompass a useful range of lengthscales for the GRF force, we choose a range from 0.0 to 2.0 in steps of 0.4 and for each of these lengthscales we average the simulation results over five runs. Here, $l = 0$ means that the previous random perturbation method (with no spatial correlation) was used, and any non-zero lengthscale means the new GRF method was used.

Figure 4.9 shows the results of this computational experiment. The results match intuition well. We would expect the $l = 0$ ('most' random) case to result in the

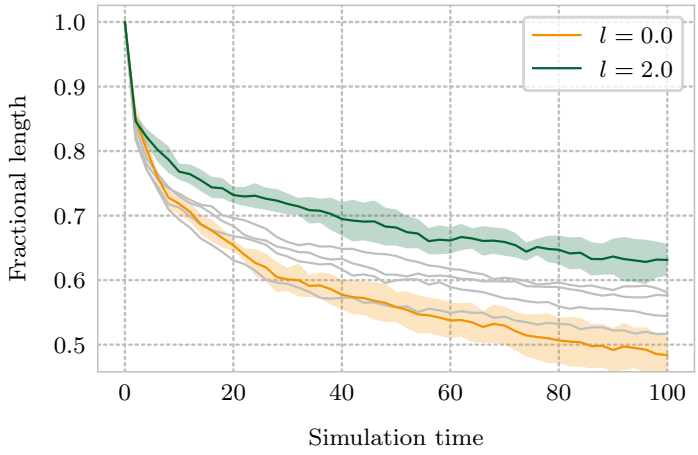


Figure 4.9: **Effect of noise correlation lengthscale on VM cell sorting.** *The lines represent the average of 5 runs with different values of l, the random noise lengthscale, with the orange highlighted line $l = 0.0$ and the green highlighted line $l = 2.0$. Grey lines are intermediate values between the two highlighted extremes. Green and orange highlighted regions denote a standard deviation either side of the mean for the two highlighted values.*

greatest probability of triggering cell rearrangement events. In the limit $l \to \infty$, the noise would be perfectly correlated over the entire domain, and for each timestep would simply result in an identical displacement being added to each vertex in the simulation, giving a simulation identical to having no random perturbation at all. As expected, $l = 0$ produces the greatest amount of cell sorting, and sorting gets progressively less good as the lengthscale increases.
This result highlights that, when including noise in vertex movement, correlation lengthscale ought to be carefully considered. In fact, the impact of lengthscale on summary statistics appears more pronounced and has a much tighter variance than the strength of the noise added, which has previously been modulated in published simulation studies [118].

4.3.4 Cell sorting in the IBM

This section details the attempts to recapitulate with the IBM the results for the VM above. The general setup is similar to the VM case. We start with a population of cells on a doubly periodic domain, allow them to relax for a short time, and then select half the cells to be labelled as 'type B'. The simulation then proceeds, with the addition of noise, for a fixed duration, and the fractional length is recorded over time.

Like the cell rearrangement threshold in the VM, there are also elements of model choice in the IBM that must be carefully considered for there to be any chance of cells sorting. Principle among these is the distance between cells. Due to velocity of the

fluid underlying the IBM being the mechanism for updating the IB node locations, for two specific nodes in the IB representation to move relative to one another, the nearby fluid must also have a relative motion. This enforces that for two boundaries to move relative to one another (to exhibit shear) they must be separated by a sufficiently large number of fluid grid spacings. As interpolation is based on a 4×4 subgrid, four fluid-grid-spacings seems a reasonable default minimum spacing between cells for there to be any chance of shear. This implies the need for a fine fluid grid which, in turn, implies the need for a large number of nodes per IB in order to satisfy a suitable spacing ratio (Section 2.6.1). This has the undesirable implication of a high computational cost. As a result, for this unit of work we reduce the number of cells in each simulation in order to keep simulation runtimes feasible on a desktop computer. We therefore simulate 36 cells on a square doubly periodic domain with 128×128 fluid grid points. We choose a gap of size $0.03 \approx 4/128$, and this geometry results in ≈ 5000 IB nodes in total.

While we initially specify a gap between cells, this is certainly not sufficient to prevent adjacent cells getting too close to each other to shear. We use an adhesion force containing a Morse-style potential as described in Section 2.8.3. Setting the interaction rest length to the distance between cells, two IB nodes at this distance experience no additional adhesive force. Two IB nodes below this distance experience a strong (and exponentially increasing) repulsion, preventing any cells coming too close together to prevent shear. Two IB nodes further apart than this rest length experience an attractive force that peaks before tailing off to zero at long distance.

Implementing differential adhesion in this context is straightforward: the strength of the heterogeneous interaction force is simply reduced compared to the homogeneous interactions by a factor μ_{het} where, in the following computational experiments, we choose a default value of $\mu_{\text{het}} = 0.25$. Osborne and colleagues choose $\mu_{\text{het}} = 0.1$ in the case of the overlapping spheres and Voronoi tessellation cell-centre models. To make the repulsion strength (interactions below the rest length) independent of the heterogeneous nature of the interactions, a separate value for the repulsive well depth, μ_{rep}, is used, which remains constant regardless of the interaction, giving the following functional form for the cell–cell force

$$\mathbf{F}(r) = \frac{2a\mu}{r} e^{-a(r-g_{\text{def}})} \left(1 - e^{-a(r-g_{\text{def}})}\right), \tag{4.8}$$

where r is the distance between two IB nodes, a is the well width and μ is either μ_{rep} (if $r < g_{\text{def}}$), μ_{def} (for homotypic interactions) or μ_{het} for heterotypic interactions.

In the following computational experiments, we use the default parameter values

given in Table 4.1, with multiples of those values being used as described for each specific experiment. Full details on the exact specification for each simulation can be found in the file `TestIbSortingWithNoiseLengthscales.hpp` which defines all IB cell sorting simulations presented in this section.

Parameter	Description	Value
μ_{def}	default cell–cell well depth	1.2×10^5
μ_{het}	heterogeneous cell–cell well depth	$0.25\ \mu_{\text{def}}$
μ_{rep}	repulsive cell–cell well depth	$10\ \mu_{\text{def}}$
g_{def}	default cell–cell gap	$0.03 = 3.84\ h$
ξ_{def}	default diffusion strength	5×10^8

Table 4.1: Default parameters for IB cell sorting simulations.

Figure 4.10 shows snapshots from a single IBM simulation with differential adhesion. The first and most obvious difference in comparison to the VM case (Figure 4.7) is that total cell sorting is not achieved. While the different cell types do aggregate over the duration of the simulation, and cell rearrangements do occur, the frequency of these events remains low. Further simulation studies in this section quantify the cell sorting extent, but what is clear is that the timescale over which cell sorting does occur is much longer than that of the VM case where, here, the timescale on which the noise acts is constant between IBM and VM simulations. Elaboration on the reasons for infrequent cell rearrangements is presented in Section 4.4 but, while cell sorting is incomplete it is far from non-existent, and is certainly instructive for our understanding of the IBM.

4.3.5 IB sorting and diffusion strength

The differential adhesion experiment performed by Osborne and colleagues assessed the impact of the random perturbation strength on the rate of cell sorting. Figure 4.11 shows the corresponding numerical experiment for the IBM. The diffusion strength is varied over an order of magnitude, with little difference in the quantity of cell sorting. While, in the complete absence of noise, no relative cell motion (and hence no cell sorting) occurs, Figure 4.11 demonstrates a significant parameter space in which cell sorting occurs, but occurs in a manner independent of the strength of the noise driving it. This result is strikingly similar to the corresponding VM simulations presented in Figure 4.8b, in which the quantity of cell sorting was independent of the strength of noise added to the system. In this case, altering the cell rearrangement threshold proved necessary to find a parameter regime where the diffusion strength did correlate

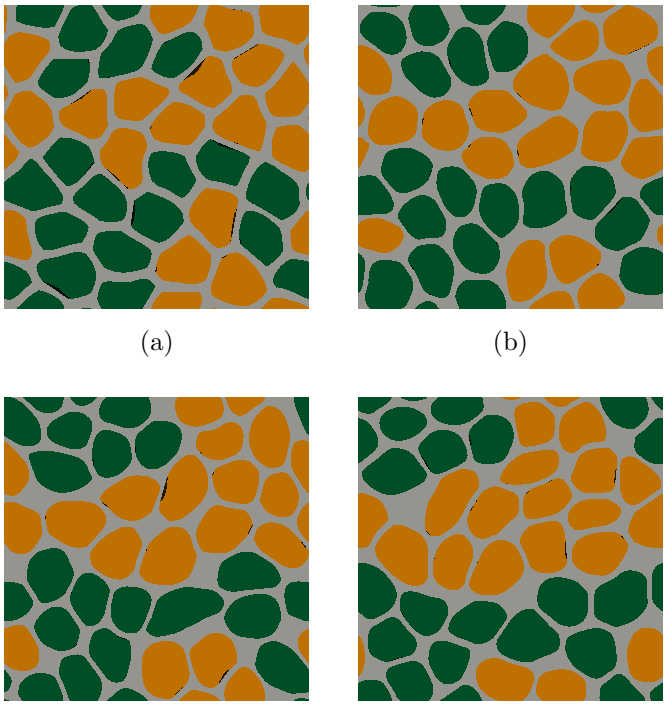


Figure 4.10: **Snapshot of IB cell sorting simulation.** *Snapshots of a simulated monolayer are shown at four selected time points. Two cell types A and B are shown in green and orange, respectively. Over time the length of the heterotypic boundary decreases but unlike in the VM case, total cell sorting is not achieved. Parameter values for this simulation are all as in Table 4.1. (a) Snapshot at* $t = 0$*. (b) Snapshot at* $t = 330$*. (c) Snapshot at* $t = 660$*. (d) Snapshot at* $t = 990$*.*

with the quantity of cell sorting. This begs the question of whether a similar model parameter has such an impact on the IBM, and this is explored in Section 4.3.7.

4.3.6 The impact of correlated noise on IB cell sorting

Section 4.3.2 extended the existing model of cell sorting using the VM, and investigated the impact of modulating the lengthscale l over which the random perturbations are correlated. The results closely followed intuition: the shorter the lengthscale of correlation, the more 'locally' random the perturbations become, and this increases the probability of triggering the cell rearrangement events that drive cell sorting. Figure 4.9 shows the characteristic reduction in cell sorting as l increases.

Is the same true for the IBM? Intuition, again, agrees that as $l \to \infty$, the random perturbations are perfectly correlated and simply act to translate the cells. We therefore expect cell sorting to decrease in the presence of a large value for l. In

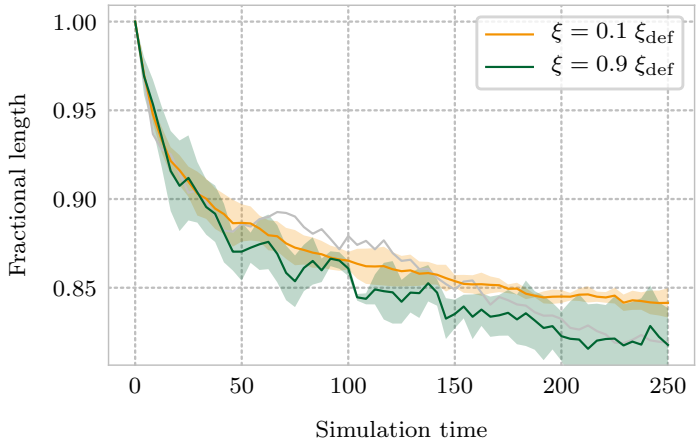


Figure 4.11: **Effect of noise strength on IB cell sorting.** *Lines represent an average of 3 runs with different values of ξ, the diffusion strength, with the orange highlighted line $\xi = 0.1\ \xi_{\text{def}}$ and the green highlighted line $\xi = 0.9\ \xi_{\text{def}}$. Grey lines are intermediate values between the two highlighted extremes. Green and orange highlighted regions denote a standard deviation either side of the mean for the two highlighted values.*

the case of a small lengthscale, the situation is more interesting. Unlike in the VM case, we cannot take the lengthscale to zero. This was the original motivation for developing a method of adding correlation: as the IB nodes can be arbitrarily closely spaced, a non-zero lengthscale is required for the addition of random noise to make any sense. Provided, though, that l is large enough compared to the distance between individual IB nodes for the simulations to be non-degenerate, what would we expect to observe? The more relevant size on which to consider this is that of the gap between cells which, as previously discussed, must be large compared to the fluid grid spacing: if l is small compared to this gap, we would expect little consistent relative motion between cells and, instead, simply a 'rippling' of the IBs themselves. As l increases, we would expect to see the noise contributing more substantially to relative motion between cells, and so would predict a sweet spot of correlation length that maximises quantity of cell sorting.

To test this hypothesis, we perform cell sorting simulations with varying values of l and record how the fractional length varies over time. Figure 4.12 shows the results. The orange highlighted value of $l = 0.1\ g_{\text{def}}$, and the green highlighted value of $l = 2.3\dot{3}\ g_{\text{def}}$ are the two most extreme simulated values of l, and both perform significantly less well than the three intermediate values (grey lines). This demonstrates the expected behaviour, and confirms that the correlation lengthscale is an important consideration in such simulations. The implications of this, and the difficulty of relating this fact to any biological correlate, is discussed at greater length

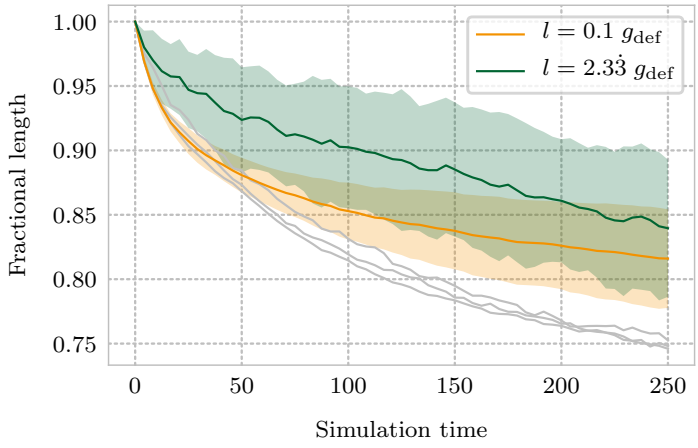


Figure 4.12: **Effect of noise correlation lengthscale on IB cell sorting.** *The lines represent the average of 40 runs with different values of l, the random noise lengthscale, with the orange highlighted line $l = 0.1\ g_{\mathrm{def}}$ and the green highlighted line $l = 2.3\dot{3}\ g_{\mathrm{def}}$. Grey lines are intermediate values between the two highlighted extremes. Green and orange highlighted regions denote a standard deviation either side of the mean for the two highlighted values.*

in Section 4.4.

4.3.7 The impact of cell gap on IB cell sorting

Finally, we return to the question posed in Section 4.3.5 of whether, in the IBM, a model parameter such as the cell rearrangement threshold substantially influences the quantity of cell sorting. While not directly related, a candidate for such a parameter is the size of the gap between cells. The reason for investigating this parameter is grounded in the IBM itself: location of boundaries are updated by the flow induced in the underlying fluid. Two IBs, therefore, that are close together will be carried along by similar fluid velocities, prohibiting the relative motion of boundaries that is necessary for cell sorting to occur. It is reasonable to assume that, if the gap between boundaries is small, cell sorting will not occur.

To test this hypothesis, we perform cell sorting simulations with varying values of the cell–cell gap, $g_{\mathrm{def}} = 3.84\ h$, and record how the fractional length varies over time. Figure 4.13 shows the results of this investigation. Reducing the cell gap, g, by a factor of three from its default value of $g_{\mathrm{def}} = 3.84\ h$ to $g = 1.28\ h$ is sufficient to completely prevent cell sorting. Enlarging the cell gap too far, by contrast, removes all semblance of cell shape, and g_{def} has been chosen for this study in an *ad hoc* manner so as to retain cell shape while enabling cell shear.

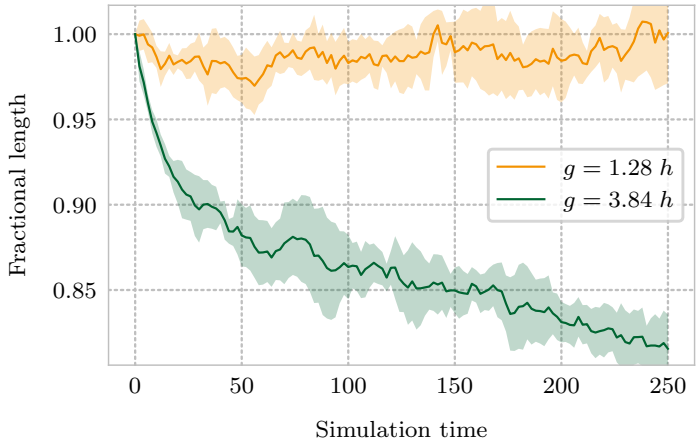


Figure 4.13: **Effect of differing cell gap on IB cell sorting.** *The lines represent the average of 4 runs with two different values of g, the cell gap, with the orange line $g = 1.28\ h$ and the green line $g = 3.84\ h$. Green and orange highlighted regions denote a standard deviation either side of the mean for the two highlighted values.*

4.4 Discussion and outlook

In this chapter we have undertaken a careful comparison of the VM and IBM by using them both to simulate a benchmark problem of cell sorting by differential adhesion. Doing this has required the development of a number of fundamentals that enable the IBM to simulate epithelial dynamics in a manner similar to the VM. The development of these fundamentals included the need to add noise to the IBM, and the subsequent implementation generalises previous attempts and applies equally well to other cell-based frameworks. This synergy has ultimately led to a more in-depth understanding of the VM, in addition to elucidating the strengths and limitations of the IBM for cell-based modelling.

We now briefly discuss several of the implementation choices made during this work. In order to incorporate changes in cell surface area in the IBM in a manner similar to that of the VM, we implemented a fairly *ad hoc* feedback between crowding and cell size (Section 4.2.2). It remains unclear the precise extent to which our implementation influences simulations of epithelial dynamics and, while out of the scope of this work, an avenue for further investigation would be to tune the algorithm to produce biologically realistic cell area distributions. Second, we chose to deviate from the study undertaken by Osborne and colleagues [118] by simulating cell sorting on a doubly-periodic domain. This choice was the only sensible way to enable any cell sorting: without some mechanism to keep IB cells constrained, the population simply rounds up and spreads apart. This is due to the membrane elasticity strength

dominating the cell–cell interaction strength in order to keep IBs coherent, and further work should seek to address this weakness in the model. For the present study, however, simulating on a doubly periodic domain is sufficient to constrain the cell locations, and has allowed simulations of cell sorting that enable useful comparison between the IBM and VM frameworks.

Turning our attention to the comparison itself, the investigation has revealed a number of points for discussion. First is the difficulty with which parameters can be related like-for-like between the frameworks. In the VM, differential adhesion is implemented by modulating the line tension parameter, and this has a non-trivial feedback with the motion of vertices. By comparison, the IBM has a more natural representation of differential adhesion, as explicit forces between cell boundaries are represented. There is no clear way to directly relate line tension to an explicit representation of forces and, therefore, models such as the IBM that incorporate such explicit representations may be better suited to modelling situations where a mechanistic understanding of the forces is known.

Furthermore, the difficulty with which the IBM was capable of recapitulating cell sorting sheds light on several of its important characteristics. The IBM struggles to display shear of cells past one another. Careful attention must therefore be paid to the gap between cells, and to the granularity of the underlying fluid grid, particularly when shear is expected. Finally, both the VM and IBM demonstrate a dependence of cell sorting on l, the correlation length for the random perturbations. This is an interesting and novel computational result, and raises the question of how values for this parameter should be chosen. Future work in this area might infer a value for l from biological observations.

In conclusion, in this chapter we have developed fundamentals of the IBM and advanced a method of adding noise in general to off-lattice cell populations. Through the application of both the VM and the IBM to simulations of cell sorting, we have undertaken the first study that compares one of the class of more geometrically detailed cell-based models to an existing framework. Through this study, we now better understand the capabilities and limitations of the IBM for studying cell populations, and are now in a position to utilise it on a suitable problem in developmental biology.

Chapter 5

A computational model of early placode morphogenesis

In this chapter, we apply the IBM to explore biophysical mechanisms of epithelial bending with application to early placode morphogenesis of the developing molar and the developing salivary gland. This work has been undertaken in collaboration with Professor Jeremy Green (King's College London).

5.1 Background and motivation

Epithelial bending and folding are ubiquitous morphogenetic deformations that play key roles in numerous developmental processes. During gastrulation, the first major morphogenetic transformation during development in most animals [77], a single-layered epithelium (blastula) is reorganised by coordinated bending and folding into a layered structure (gastrula). Neurulation, a critical stage in the development of the brain and spinal cord, also requires carefully controlled tissue folding. In such processes, coordinated cell shape changes and cell neighbour exchanges drive bending and folding. Because mechanical forces underlie these deformations, elucidating the mechanics of epithelial bending is necessary if we are to understand embryonic development and associated pathologies. To date, a number of bending mechanisms have been characterised:

Apical constriction. Apical constriction is responsible for, among many other processes, ventral furrow formation and dorsal closure in *Drosophila*, early stage neurulation in vertebrates [31], and gastrulation in *Xenopus* [77] and *C. elegans* [145]. It is characterised by a 'purse string' of apically localised actin circumferential in each

cell which, upon constriction, reduces the apical surface area of the cell. Coordinated constriction by a group of neighbouring cells generates a local invagination in the epithelium [152]. Progressive apical constriction in coordination with other cell shape changes can lead to complete folding [32].

Basal wedging. Basal wedging is geometrically similar to apical constriction in that the apical surface of cells becomes significantly smaller than the basal surface, but it occurs via a distinct mechanism. This phenomenon is observed in several pseudostratified epithelia including the chick neural plate [148]. In a pseudostratified epithelium, cells are narrow enough that they bulge around the location of their nucleus. The nucleus moves during the cell cycle in a process known as interkinetic nuclear migration, and is located basally during S (DNA synthesis) phase. During basal wedging, S phase is longer for cells forming the hinge about which bending occurs [149], thus a high proportion of the cell nuclei remain basal, ensuring the basal surface is significantly larger than the apical surface. This results in a bend in the epithelium.

Differential proliferation. The differential proliferation hypothesis posits that a stratified epithelium may bend by increased cell proliferation in a certain region of the tissue. During such bending, undulations are generated by increased proliferation perpendicular to the epithelium generating a 'down growth' responsible for shape change. An example of this is epithelial dysplasia, a pathology of abnormal growth and differentiation [7]. Such a bend is dependent on a number of factors including the stiffness of the underlying mesenchyme and the coordination of cell spindle orientation during division [122].

A new method of epithelial bending. The tooth and salivary placodes, similar in their early stages of development, have recently been shown to exhibit a bending mechanism that is quite different from those summarised above, and which remains poorly understood. The motivation for the work presented in this chapter is to create a computational model able to probe the mechanics of bending in these tissues.

5.1.1 Computational modelling of out-of-plane deformations

Before developing a suitable computational framework within which to investigate the mechanics of epithelial bending, we first review previous work in this area. Possibly the earliest example, by Odell and colleagues [116], is described in Section 1.2.2, with

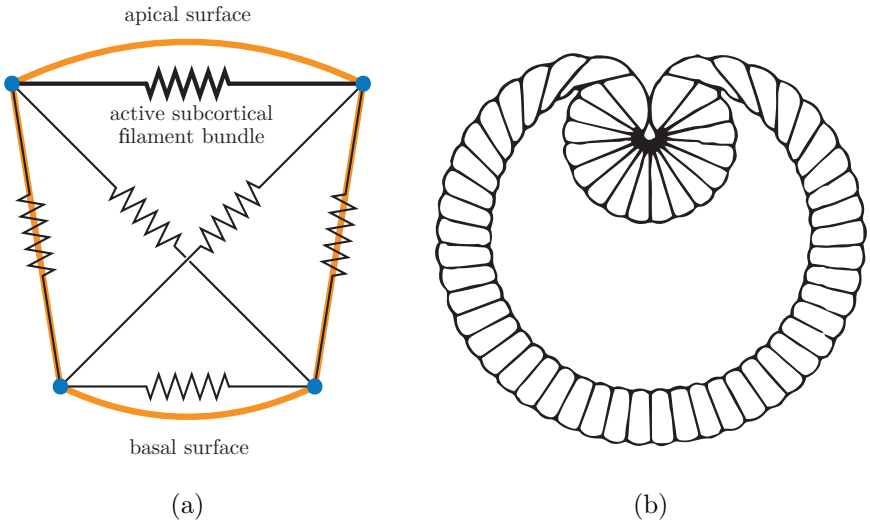


Figure 5.1: **Viscoelastic model of embryonic cross-section.** *(a) The mechanical elements comprising a single cell in the model by Odell and colleagues [116]. Linear springs join the corners, and an apical active element contracts in response to extension. (b) A circle of identical cells evolves to the pictured configuration during a simulation of ventral furrow formation in Drosophila. Images are reproduced from [45] (Figures 1a and 1b) with permission from the authors.*

further details in Figure 5.1. Using this model, the authors successfully recapitulated gross phenotypic observations of normal development (Figure 5.1b). A more recent study by Polyakov and colleagues [128] investigates *Drosophila* gastrulation in a similar manner using a cross-sectional vertex model, with a more detailed examination of the role of apical constriction based on current knowledge of its biological underpinning.

An alternative paradigm for modelling out-of-plane deformations in two dimensions is to model the apical surface, rather than a cross-section, of the tissue. An example is provided by Spahn and Reuter [150] who investigate *Drosophila* ventral furrow formation using an 'en face' VM. A strength of this work is the ease with which summary statistics can be compared to experimental observations which, in this system, are typically viewed using live confocal imaging of the apical surface. Figure 5.2 shows three snapshots from Spahn and Reuter's simulations of ventral furrow formation in which the invaginating (red) cells reduce in apical surface area.

A final example is a computational model of tooth placode morphogenesis at a late stage of development. Marin-Riera and colleagues [97] use the software EmbryoMaker (described in Section 1.3) to simulate the development of the tooth germ in three dimensions, representing mesenchymal and suprabasal cells as spheres and epithelial

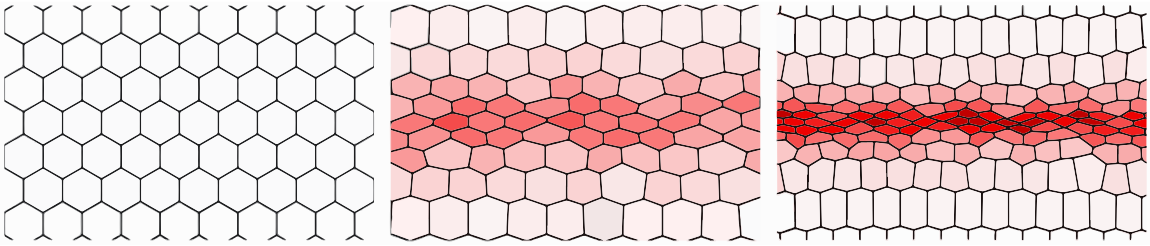


Figure 5.2: **Vertex model of *Drosophila* ventral furrow formation.** *Three snapshots of a simulated epithelium demonstrate Spahn and Reuter's model [150] recapitulating ventral furrow formation using a two-dimensional representation of cells' apical surfaces with red shading proportional to the size of the apical surface area of cells. Reproduced from [150] (Figure 2e) under the Creative Commons Attribution License.*

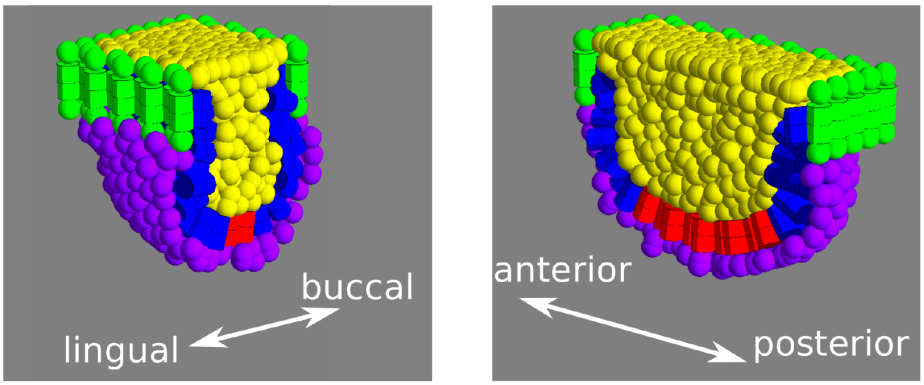
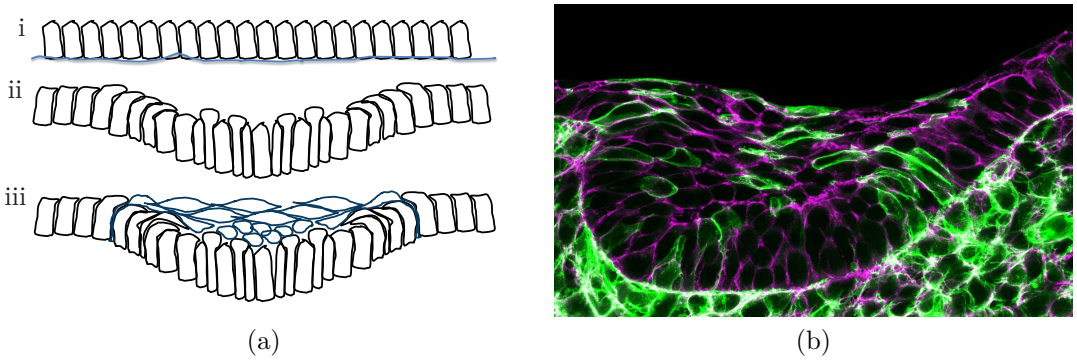


Figure 5.3: **Three-dimensional model of molar development.** *Initial conditions for a simulation by Marin-Riera and colleagues [97]. Blue and red represent the epithelial cells, yellow the supbrabasal cells, and purple the mesenchyme. Green cells and the topmost layer of yellow cells are fixed in space. Reproduced from [97] (Figures 2a and 2b) under the Creative Commons Attribution License.*

cells as cylinders defined by two end points. This model explores cell division and adhesion as drivers for significant morphogenetic deformation, finding that differential tissue growth and differential adhesion are sufficient to build the three-dimensional structure of the tooth germ. Figure 5.3 shows the initial geometry for a single tooth simulation, where each sphere or cylinder represents a single cell.

These examples showcase a variety of geometrical approaches for modelling out-of-plane deformation: cross section, top-down, and fully three-dimensional. While each is useful in specific circumstances, none consider detailed, emergent cell shape dynamics. This, as we will see, is an important component of the system that we are investigating, and requires a novel modelling approach.



(a) (b)

Figure 5.4: **Simplified schematic, and microscopy slide, of the tooth placode.** *The tooth formation system, as a cross section through the developing jaw, orientated with buccal–lingual (cheek–tongue) axis from left to right. (a) i Initially flat epithelial monolayer anchored to the basal lamina. ii Initial bending during early stages of invagination, characterised by cells beginning to form hooks directed towards the centre of invagination, followed by cells (blue) beginning to detach from the basal lamina (delaminating). iii After delamination, cell migration and intercalation results in arcs of elongated cells above the invagination, thought to be under high tension. (b) Image of jaw explant showing suprabasal arcs, showing E-cadherin (green) and F-actin (magenta). Images courtesy of Professor Jeremy Green.*

5.1.2 Characterisation of the model system

Professor Green's lab study placode morphogenesis using the mouse as a model system. Figure 5.4 shows a schematic of one particular stage of the system in question. The images represent thin (roughly 10μm) slices of tissue taken from the jaw of a developing mouse embryo. Remarkably, while *ex vivo* these explants continue to undergo normal morphogenesis for a time, allowing for detailed microscopy studies of the early stages of morphogenesis. Figure 5.4b shows a cross section through the tissue, with the epithelium comprising a monolayer of cells initially arranged in a 'palisade' above a basal lamina, under which is a mass of mesenchymal cells.

The process, characterised by Panousopoulou and Green [120], proceeds as follows. An initially flat epithelium invaginates to form a furrow (out of the plane of the images), in which teeth will later form. Several distinct stages are identifiable in this process. In the first stage (i to ii in Figure 5.4), cells take on a 'hook' shape oriented towards the centre of the invagination. This allows some cells to, seemingly, drag themselves away from the basal lamina and begin migrating towards the centre of the invagination (ii to iii in Figure 5.4). Intercalation (in which cells move past one another) then occurs in these suprabasal arcs, rapidly drawing together the shoulders of the invagination to crease the tooth furrow.

Of particular interest in this study is the very earliest stage, during which the

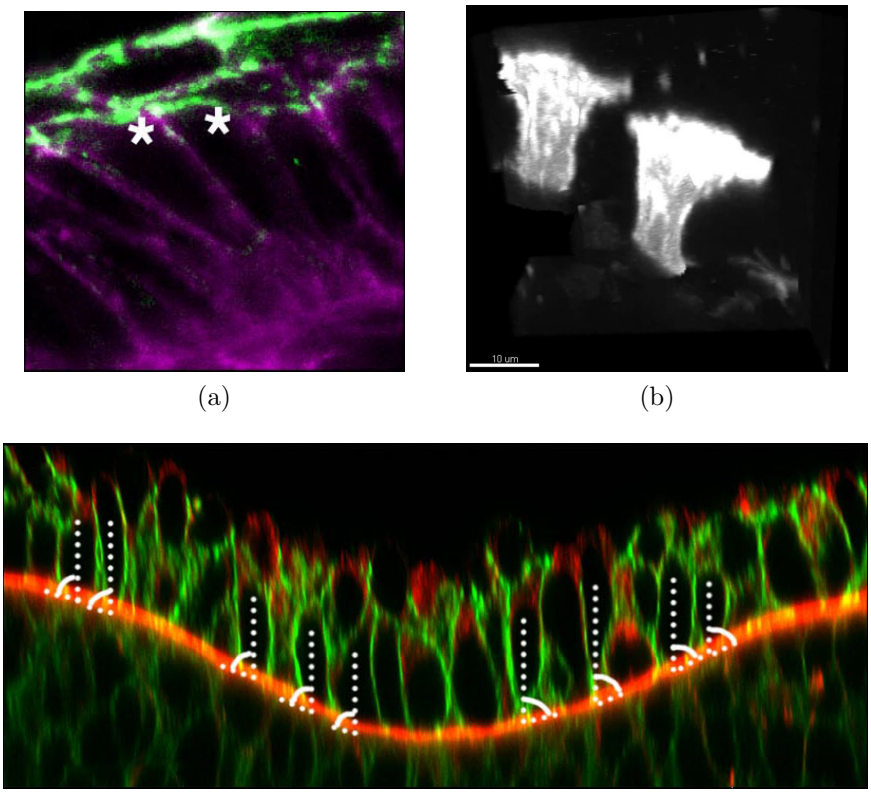
flat epithelium invaginates. This is characterised by (vertical) relative motion of cells past one another, and by hook-shaped cells pointing towards the centre of the invagination. Much detail concerning the mechanical interactions involved in these processes remains unclear. What is clear, however, is that the well-characterised processes of apical constriction and basal wedging are certainly not driving bending in this system. This has been conclusively demonstrated by Panousopoulou and Green [120]: cells with basal nuclei are not observed to be more abundant in the explants (ruling out basal wedging), and no apical enrichment of actin is observed (ruling out apical constriction). In this case, the biology is inconsistent with any previously reported methods of generating out-of-plate epithelial deformations.

The following specific observations succinctly characterise the main biological observations that have given rise to the mechanical hypothesis explored in this study.

Adhesive protein localisation. Fluorescent tagging has revealed that an adhesive transmembrane protein, E-cadherin, is abundant in the apical domain of cells in the early placode [120] (Figure 5.5a). The absence of this protein in lateral domains indicates that adhesion is probably a contributing factor in the observed deformation.

'Penguin' cell shapes. *Ex vivo*, a high proportion of cells in the placode take on a hook-shaped geometry, similar to the beak of a penguin, with the hooks consistently oriented towards the centre of the invagination. This has been characterised by Li and Green [56], with 75% of cells in the invaginating salivary gland displaying centrally directed protrusions in their apical domains. Examples of such cell shapes can be seen in Figure 5.5b. A working hypothesis is that these structures function in a similar way to lamellipodia, providing an anchor ahead of the cell against which to pull forward. Lamellipodia are protrusions of cytoskeletal actin, formed by epithelial cells, and are implicated in cell motility [1]. These shapes are also observed, but in fewer number and with a more slight beak, in fixed specimens, indicating the transient nature of the hooks and supporting a lamellipodial-like function.

Outward lean of cells. The cells in the placode are shown to remain largely vertical as the invagination deepens. This is characterised by an acute angle between the long axis of individual cells and the basal lamina, and is referred to here as the 'outward lean' (Figure 5.5c).



(a) (b)

(c)

Figure 5.5: **Elements of the model system.** *(a) E-cadherin (green) production is shown to be upregulated in the apical domain of cells in the early placode. Reproduced from [120] (Figure 4d) under the Creative Commons Attribution License. (b) Snapshot of a three-dimensional reconstruction of two individual cells displaying the 'penguin'-like hooked geometry. (c) Characterisation of cells forming an acute angle with the basal lamina, referred to as an 'outward lean'. Images (b) and (c) courtesy of Professor Jeremy Green.*

Vertical telescoping. As cells begin to hook towards the centre of what will later be the invagination, the epithelium bends into a shallow valley-shape, with central cells lower than the outer flanking epithelium. The region of bending, approximately 10-20 cells from shoulder to shoulder, is flanked by flat epithelium. This process appears to be characterised by a small relative motion between each pair of neighbouring cells, which cumulatively generates the observed bend over the width of the epithelium. This relative motion has been characterised by looking at imaging data from the tissue explants, and has been termed 'vertical telescoping' by Professor Green. This concept is best explained with reference to Figure 5.6. An analogy of this kind of motion is the steps of a rising escalator, with each cell moving vertically relative to its neighbour. When extended to a concentric arrangement about the centre of an invagination, this is akin to the sliding sections of a telescope. Cells

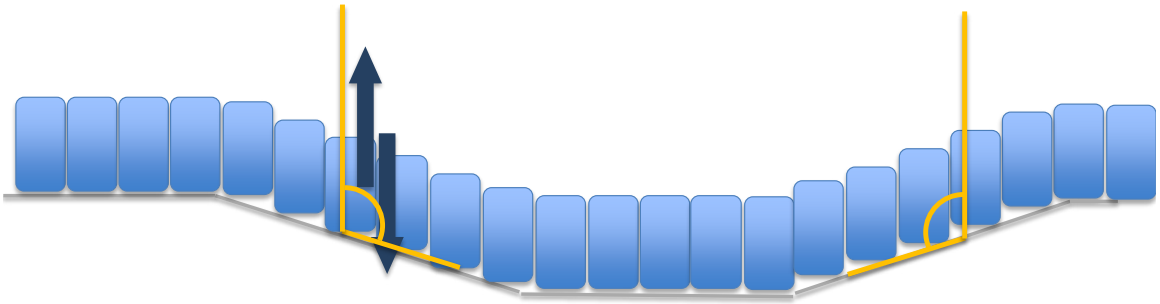


Figure 5.6: **Schematic of vertical telescoping.** *Cells appear to move with respect to their neighbours in such a way as to effect a bending of the epithelium. The motion appears to be relative, and there is some evidence [56] that cells 'lean' in the sense that their long axes remain upright with respect to the flanking epithelium, rather than remaining perpendicular to the bending basal lamina.*

remain upright as the epithelium bends, from the frame of reference of the basal lamina, and this outward lean has been quantified from experimental data. This process has been observed and quantified by Green and colleagues in the tooth germ epithelium, but this quantification is yet to be published.

5.2 Computational modelling of this system

Which mechanical interactions between neighbouring cells can suffice to give rise to the above observations, in particular (i) hook-shaped cells, (ii) invagination (bending of the basal lamina), and (iii) outward lean during vertical telescoping? To address this question, we will develop a novel computational framework within which we can probe the role of mechanics in generating this form of epithelial bending.

Figure 5.4b shows a huge variety of cell sizes, shapes and protein expression levels. The integrated impact of multiple biological factors almost certainly implies unintuitive outcomes of experimental perturbations, and the value of a computational model comes from the ability to quantitatively observe these consequences.

5.3 Methods

5.3.1 An IB model of early placode development

We first identify the key biological elements that must be accessible in a computational model of epithelial bending, in order to select an appropriate modelling approach. We wish to capture complex and dynamic cell shapes and relative movements. In addition, observed heterogeneities in cadherin expression levels around cell

membranes indicate that adhesion probably plays an important role in the process. For this reason, a framework giving us explicit control over the location and strength of cellc-cell interaction forces is required. The biological observations are from thin tissue explants, roughly a cell diameter thick. For this reason, a two-dimensional model is appropriate as the system itself, and observations thereof, are also essentially two-dimensional.

Several of the models outlined in Section 1.2.3 may be suitable for this study, including the many-vertex type model, the SEM, and the IBM. Of these, the high resolution representation of cell geometries, the ease with which explicit mechanics can be investigated, and the availability of a high quality implementation makes the IBM an appropriate framework to begin modelling the system.

The essential information that informs our computational design is summarised as follows:

- there exists a central region of cells that do not exhibit apical protrusions;
- cells to the left and right of the central region develop 'hook' shapes, hooking in the direction of the central region;
- while this is happening, the epithelium bends;
- experimental work demonstrates the presence of differing levels of transmembrane proteins, in particular an increase in E-cadherin in the apical domain.

Distilling these essential elements, we aim to assess to what extent the following three success criteria can be met by investigating mechanical hypotheses in an IB computational model:

1. quantifiable bend of the epithelium;
2. quantifiable 'hooking' shape;
3. quantifiable 'outward lean' of cells.

The basic framework. We simulate an epithelium made up of a small number of cells, initially in a regular columnar palisade. The early stages of epithelial bending in this system typically involves approximately 15 cells, so we choose to simulate this number. Each cell's membrane is represented by a single IB.

There are mesenchymal cells situated basal to these columnar cells that do not appear to be mechanically active during invagination. Instead of treating these cells explicitly, for simplicity we represent only the boundary between the columnar epithelium and the underlying mesenchyme (Figure 5.4b). This takes the form of a single IB which, due to the periodic boundary conditions, forms a closed loop partitioning the domain in two. We refer to this IB as the 'basal lamina' (white dotted line in Figure 5.7b).

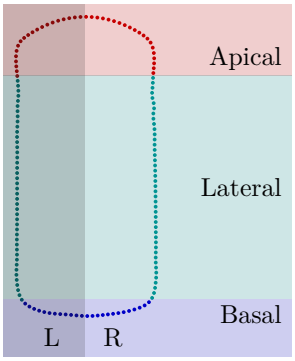
Three regions of cells. Based on the data presented above, we assume that the tissue comprises a 'passive' central region with 'active' regions on either side. Again, with reference to the biology, a sensible choice appears to be three cells in the central region and six either side (Figure 5.7b). We suppose that the mechanisms driving bending in this system are symmetric about the centre, so label the regions A (left), B (centre), and A' (right), such that any mechanical heterogeneities in region A are mirrored in region A'.

Six regions per cell. For each cell in the active regions, we introduce mechanical inhomogeneities to drive relative motion of cells. First, we represent epithelial apical–basal polarity via distinct apical and basal regions of each cell boundary. The remainder of each boundary, between the apical and basal domains, is denoted lateral. Finally, we label nodes as being either 'inner' or 'outer' by dividing them left and right along the long axis of the cell, through its centroid, with the 'inner' half being those closer to the centre.

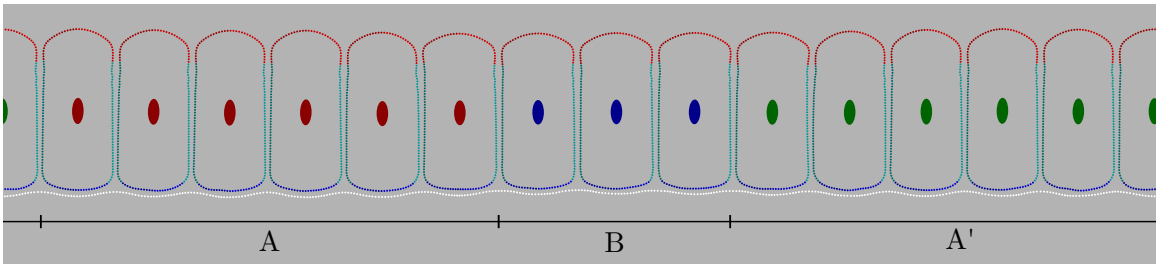
We thus have six regions, each possibly having different mechanical properties: left and right apical, lateral, and basal (Figure 5.7a). Figure 5.7b brings this geometric information together: the three regions of cells, each with their six regions and, in addition, the basal lamina.

Summary statistics. Defining suitable summary statistics is essential to extract meaningful information from a simulation and to be able to quantitatively assess how different hypotheses impact simulation output. In the case of tooth placode morphogenesis, some measure of shape or curvature of the combined group of cells is needed. In addition we are interested in just how 'hooked' individual cells are.

Bending quantification. There are myriad possible ways to quantify the amount of epithelial bending, in particular the bend of the interface between the



(a)



(b)

Figure 5.7: **Cell and tissue geometry in IBM simulations of epithelial bending.** *(a) Each cell is comprised of six regions: apical, lateral, and basal, each of which is either 'inner' (closer to the centre of the invagination), or 'outer', marked here as L (left) and R (right) in the absence of a center about which to orient. (b) Individual cells are represented by the 15 IBs. The white IB represents the basal lamina, the interface between the mesenchyme and epithelium. Three regions of cells are present: here, visualised with red ellipses (A: left region), blue ellipses (B: central region), and green ellipses (A′: right region). Each ellipse is centred at its corresponding cell's centroid; its aspect ratio and orientation are the aspect ratio and long axis of the cell, respectively (this is for visualisation only, and serves no mechanical or modelling purpose). Within each cell, six regions are present. These regions are (from top to bottom) apical (red points), lateral (teal points), and basal (blue points), with those to the left of the cell's long axis visualised with a darker colour.*

mesenchyme and the epithelial monolayer. In this framework, that boundary is modelled explicitly by the basal lamina: a single IB comprised of a number of discrete points in space.

A simple measure of bending is therefore the maximal height difference between any two locations in the lamina. More precisely, if the basal lamina is comprised of N nodes with locations $\mathbf{\Gamma}^n = \left(\Gamma_x^n, \Gamma_y^n\right)$ for $n = 1, \ldots N$, we define

$$\text{bend} = \max_n\{\Gamma_y^n\} - \min_n\{\Gamma_y^n\}. \tag{5.1}$$

This bend can be converted to an angle by noting that this height difference occurs over half the domain of (nondimensional) unit width, leading to the following

definition

$$\text{lamina angle} = \arctan\left(\frac{\text{bend}}{0.5}\right). \tag{5.2}$$

This metric does not distinguish between a bend 'upward' or 'downward'. However, visual inspection suffices to determine whether the lamina is behaving similar to the *ex vivo* system. A significant benefit of this measure is its simplicity and ease of implementation.

Asymmetry skewness quantification. We also wish to quantify cell shapes in simulations in order to assess apical 'hooking' as observed *in* and *ex vivo*. There are several commonly used summary statistics for a planar body's shape in two dimensions [48]. Perhaps the most common is aspect ratio, the ratio of the largest diameter to the smallest perpendicular diameter. A related but more robust measure is the ESF, described in detail in Section 2.6. These both attempt to capture the deviation from regularity. Other shape factors such as circularity (a function of surface area and perimeter) and waviness (a function of perimeter) attempt to capture the smoothness of the shape's outline. None of these measures can distinguish between a cell with a protrusion to the left or right. We therefore instead propose a measure based on Pearson's moment coefficient of skewness which, for a random variable X, is defined as

$$\text{Skew}\,(X) = \mathbb{E}\left[\left(\frac{X-\mu}{\sigma}\right)^3\right], \tag{5.3}$$

where μ and σ denote the mean and standard deviation of X.

Here, we consider the skewness of the mass distribution of each cell along the x axis. In this way, a cell hooking left has a negative skew whereas a cell hooking right has a positive skew. The following algorithm is used to compute the skewness measure, key steps of which can be seen in Figure 5.8:

Defined in:

- `ImmersedBoundaryMesh::GetSkewnessOfElementMassDistributionAboutAxis()`

Input:

- Discrete number of points representing a closed polygon
- An arbitrary axis about which the polygon's mass distribution is calculated

Output:

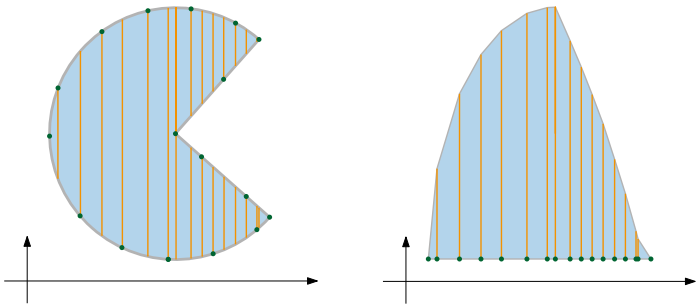


Figure 5.8: **Skewness measure of planar-body asymmetry.** *Planar-body asymmetry measures for understanding the shape of hook-like cells. Diagram showing key steps in the skewness calculation. Left: for each point in the discretisation of the polygon, calculate the length of the intersection of the vertical line through that point with the polygon. Right: those points form a mass distribution along the axis, which can be integrated directly to calculate the distribution's skew.*

- $S \in \mathbb{R}$, the skew of the polygon's mass distribution about the axis

Algorithm:

1. Reorient geometry
    - Calculate centroid of polygon and translate to the origin
    - Rotate polygon so the axis is vertical
    - The positive x-axis is now the direction along which the mass distribution is calculated
2. Calculate mass distribution of the polygon
    - Sort points representing the polygon from left to right
    - For each point, calculate the number of intersections between the polygon and a vertical line through that point:
        - Record all y-locations of intersection points: this should be even, unless two x-locations are perfectly coincident and one is an inflection point in the shape
    - Calculate the length of the intersection for each x-location: the linear interpolation between these lengths is the piecewise linear mass distribution of the polygon
3. Calculate the skew
    - Calculate the moments of the mass distribution by direct integration
    - Calculate the skew using the moments

Further details on the implementation of this algorithm can be found in Appendix B.1.

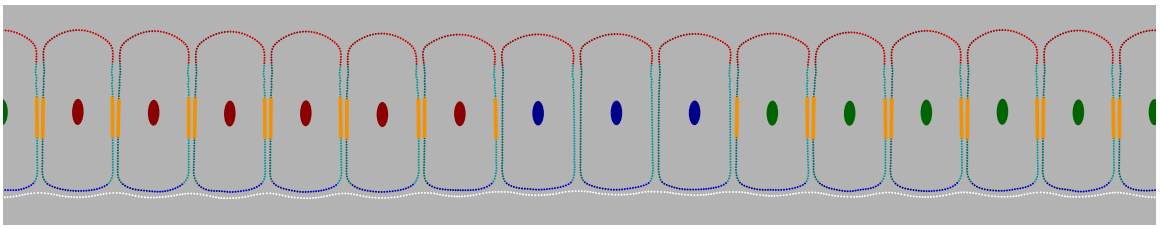


Figure 5.9: **Determining outward lean of cells.** *The middle third of each lateral domain in regions A and A' is displayed as an orange line. The average angle of these lines to the vertical is a measure of how upright a cell is, while the lamina angle defines how horizontal the placode is.*

With the ability to determine the skewness of a simulated cell about a given axis, this gives us ready access to the skewness distribution for, say, all cells in active regions. The mean and standard deviation of the absolute skewness of all cells in regions A and A' give insight into the degree of hooking observed in a given simulation.

Outward lean. To assess whether cells are exhibiting any outward lean we need a measure of how upright cells are in comparison to how horizontal the basal lamina is locally. For the former, a simple metric is to determine the average angle of the two lateral domains in each cell in regions A and A'. To reduce the influence of bending at the apical and basal ends of each lateral domain, we measure the angle that the middle third of the lateral domain makes with the vertical. Two lateral domains per cell, for 12 cells in regions A and A' gives 24 angles, the average of which we take as a measure of how 'upright' cells are in the active regions. Figure 5.9 shows a schematic of the 24 angles used in this calculation.
For the basal lamina, we take the lamina angle as a summary statistic of lamina bend. A new summary ratio, the 'lean ratio', of average-lean to lamina-angle is a statistic where 0 indicates that cells in the active region are on average perfectly vertical, regardless of the lamina angle, while 1 indicates that cells in the active region are on average perfectly perpendicular to the basal lamina.

5.3.2 IB region tagging

We next describe the algorithms used to assign nodes to their correct region (of the six regions defined in Figure 5.7). Region tagging must work robustly, as the dynamics of the simulations rely on implementing different mechanical rules in different regions. A naïve approach would be to set the node regions correctly at the start of a simulation and allow their locations to evolve over time. This produces an acceptable initial

configuration, but the relative motion of cells ensures that, after some duration of simulation, the regions no longer line up at all well between adjacent cells. Instead, we turn to the biology for inspiration: the basal domain of a cell is that which is anchored by integrins to the basal lamina, while the apical domain includes any free surface above the epithelial monolayer. Based on this, our algorithm for assigning nodes to regions is defined as follows:

Defined in:

- `ContactRegionTaggingModifier`

Input:

- An IB discretised by N nodes
- For each of the N nodes, a list of all nodes in other IBs within a specified interaction distance

Output:

- For each of the N nodes, the region (of six) that it belongs to

Algorithm:

1. Identify axes of cell
    - The short axis $\mathbf{s} = (s_x, s_y)$ of the cell is calculated as the dominant eigenvector of the shape's inertia matrix around its centroid [50]
    - Orient the short axis such that $s_x > 0$, i.e. `if` $s_x < 0$ `then` $\mathbf{s} := -\mathbf{s}$
    - If the ESF of the cell is within some tolerance of 1, the axis is unreliable: in this case, take $\mathbf{s} = (1, 0)$
2. Identify all basal nodes by determining which nodes are in contact with the basal lamina
    - For each $\mathbf{\Gamma}^n$ in the IB, identify all neighbours (all nodes within an interaction distance)
    - If any neighbour node is in the basal lamina, mark as basal and left or right depending on the sign of $\mathbf{\Gamma}^n \cdot \mathbf{s}$
3. Identify all apical nodes by identifying the 'free surface' at the apical domain of the cell
    - Proceed anticlockwise from the right-most basal-tagged node, and then clockwise from the left-most basal-tagged node, find the first node to not participate in any neighbour interactions
    - The 'free surface' is deemed to be all nodes contiguously between the two nodes identified above

- Tag these nodes $\mathbf{\Gamma}^n$ as apical and left or right depending on the sign of $\mathbf{\Gamma}^n \cdot \mathbf{s}$

4. Identify lateral nodes

Further details on the implementation of this algorithm can be found in Appendix B.2.

5.3.3 IB remeshing

In contrast to the examples of (Chapter 4) where forces are relatively homogeneous, in this setting the nodes representing the IBs can often be dragged around or bunched up in a manner that cannot be overcome by the cortical membrane force, which usually acts to keep the boundary nodes evenly spaced. Over many time steps, this can cause complete model degeneracy: if nodes do become crossed over, or move too far from their neighbours, the model can produce uncontrolled node movements, generating unphysical cell shapes. A second cause of 'node bunching' is the deliberate addition of mechanical inhomogeneity into the model (see Section 5.4.2). Over many time steps, this causes the nodes to take on an uneven equilibrium spacing, explored in greater detail in Appendix B.3. Nodes ought not bunch up at all, even in the presence of imposed mechanical inhomogeneity.

These two problems have one thing in common: nodes becoming unevenly spaced around the IB. It is worth remembering at this point that the nodes themselves are simply a discretisation of the boundary and do not represent, say, material points on the cell's cortex. We therefore propose the notion of 'remeshing' which, at given points in time, redistributes nodes around the IB in such a way as to restore equal node spacing. A simple algorithm for remeshing an IB is to linearly interpolate a closed path through every node in the IB, and reposition nodes to be evenly spaced along this linear parametric representation of the continuous shape:

Defined in:

- `ImmersedBoundaryMesh::ReMeshElement`
- `MeshUtilityFunctions.hpp (EvenlySpaceAlongPath())`

Input:

- The locations of N nodes forming a closed polygon

Output:

- For locations of N nodes, evenly spaced around the polygon

Algorithm:

1. Choose a random starting point

2. Undo any wrap-around due to the periodic domain
    - Replace each location with the location of the previous node plus the vector between them
3. Evenly space nodes around the polygon
    - Calculate a unit vector for each edge, and the partial perimeter sums
    - Calculate the desired node spacing, s, as total perimeter divided by number of nodes
    - Place the i^{th} node distance si along the linear perimeter of the polygon
4. Conform each location back to the periodic domain

Taken together, the summary statistics, region tagging and remeshing algorithms are enhancements on top of the basic IB functionality that enable the exploration of the mechanical hypotheses described in this chapter.

5.3.4 The IB framework for this study

Before applying the model to elucidate the mechanics of early placode development, we first recap the essential elements of the IB computational framework.

As in Chapters 2 and 4, we work with nondimensional parameters in a nondimensional doubly periodic unit domain $\Omega = [0, 1] \times [0, 1]$. This domain consists of a grid with 192×192 points on which the underlying fluid problem is solved, along with a number of off-lattice nodes, making up the IBs, between which specific forces act.

Those forces are categorised as being inter- or intracellular forces which define, respectively, the cell–cell adhesive and membrane-elastic interactions between nearby nodes. Intercellular forces act between all pairs of nodes, within a threshold distance, that are in different IBs. Intracellular forces act between adjacent nodes within a specific IB. Figure 2.1 shows a schematic summarising the geometric setup of the basic IBM.

In this study, forces are modelled as springs using the Morse-type potential introduced in Section 2.8.3. For two nodes that interact with each other, this gives rise to the following functional form for the intercellular forces,

$$\mathbf{F}_e(r) = \frac{2a_e\kappa_e\Phi_e}{r} e^{-a_e(r-l_e)} \left(1 - e^{-a_e(r-l_e)}\right), \tag{5.4}$$

and for the intracellular forces,

$$\mathbf{F}_i(r) = \frac{2a_i\kappa_i\Phi_i}{r}e^{-a_i(r-l_i)}\left(1 - e^{-a_i(r-l_i)}\right), \tag{5.5}$$

where subscript e refers to external (intercellular) and subscript i refers to internal (intracellular) parameters, κ and a denote the depth and width, respectively, of the potential well, r is the distance between the interacting nodes, and l is the equilibrium distance of the bond. Φ is a function that we allow to vary arbitrarily as necessary to test specific hypotheses in the simulations to come.

In addition to the prescribed inter- and intracellular forces between nodes, we apply a GRF force as described in Section 4.2.3.1. This force is added to each simulation with the purpose of perturbing the dynamics away from any extreme realisations, and gauging the uncertainty in specific summary statistics for given parameter sets.

5.4 Results

In this section, we detail the biological hypotheses that were incorporated into the computational model, and the extent to which the success criteria were met. Table 5.1 contains reference values for parameters that vary during simulations undertaken in this study. For each set of simulations, the values stated in their respective descriptions are the multiple by which they differ from the values in this table. Due to the computational implementation of the IBM within Chaste, all parameter values are nondimensional.

5.4.1 Increased apical adhesion

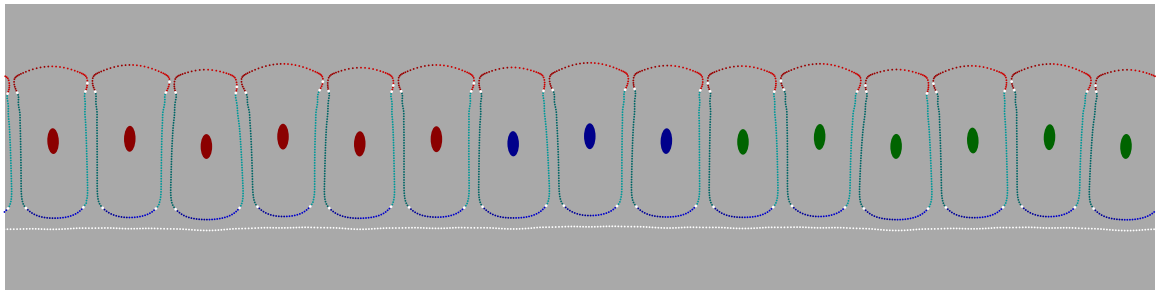
The increased apical adhesion hypothesis is based on the experimentally demonstrated upregulation of E-cadherin. The hypothesis posits that bending can be generated simply by means of altered adhesion strength in certain portions of the cell membranes. The basic outline is that cells would, in response to some as-yet unidentified spatial cue, know the relative location of the centre of the invagination and up-regulate the production of certain adhesion proteins in particular domains. This up-regulation would result in the 'inner' (towards the eventual invagination) apical domain of the cells having a greater adhesive affinity to their neighbours, which would cause the cortex of the cell to crawl over its neighbour, generating motion in a manner akin to a caterpillar track. One could imagine this hypothesis recapitulating the observed 'hook' shapes of cells, as well as the relative cell–cell motion describing the observed vertical telescoping.

Parameter description	Symbol	Ref value	Simulations
Intracellular spring constant	κ_i	2.5×10^8	Figures 5.10 to 5.17
Intercellular spring constant	κ_e	0.8×10^7	Figures 5.10 to 5.17
Apical adhesion multiplier	aam	1.0	Figures 5.10 to 5.17
Inner-apical stiffness multiplier	ias	1.0	Figures 5.11 to 5.17
Support spring strength	sss	0.006	Figures 5.12 to 5.17
Diagonal spring strength	dss	0.5	Figures 5.16 and 5.17
Cyclic frequency	f	0.125	Figures 5.14 to 5.17
Cyclic on-proportion	cop	1.0	Figures 5.14 to 5.17

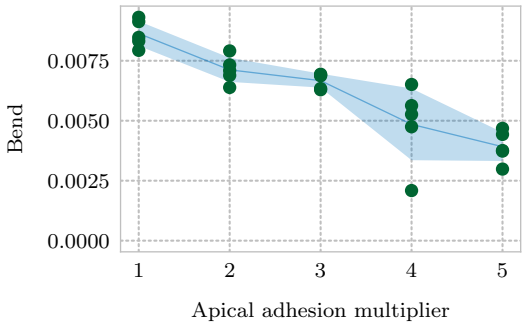
Table 5.1: **Reference parameter values for placode morphogenesis simulations.** *Unless explicitly stated, the parameter values for simulations in Section 5.4 are those in this table, or the stated multiples thereof. Apical adhesion multiplier is the factor by which apical adhesion is modified, for simulations incorporating the increased apical adhesion hypothesis. Inner-apical stiffness multiplier is the factor by which inner apical domain stiffness is reduced, for simulations incorporating the active cytoskeletal remodelling hypothesis. Support spring strength is the strength of additional tensile elements present in some simulations to represent explicit epithelial polarity, and is expressed as a multiple of the cell–cell spring constant. Additional diagonal spring strength is the strength of an additional diagonal tensile element applied corner-to-corner in the final simulations, and is expressed as a multiple of the support spring strength. Cyclic frequency is the frequency at which the inner apical domain is modified in simulations incorporating oscillatory behaviour, and the cyclic on-proportion is the fraction of a cycle during which this modification occurs. Details of all other simulation parameters not varied in this study can be found in the source code.*

We test this hypothesis using our model by increasing the strength of cell–cell interactions involving nodes in the inner apical region. Specifically, any right apical node in region A and any left apical node in region A' involved in any neighbour interaction is given an increased interaction strength compared to the baseline interaction strength of a node in the apical domain of region B. Specifically, the intercellular force is modified so that $\Phi_e = aam$ whenever a node involved in an intercellular interaction is tagged as 'inner apical', and $\Phi_e = 1$, otherwise. The value aam varies between simulations.

Figure 5.10a shows a representative example of such a simulation, with inner apical adhesion increased five-fold. We observe slight bulging (with no directionality) in the apical domains of several cells but no protrusions. In addition, no bend is observed in the basal lamina. These findings are preserved across a large range of values for the increased apical adhesion (Figure 5.10b): virtually no bend is observed. The explanation for this is clear: there is perfect symmetry in adhesive interactions. We simply have not yet introduced any mechanical heterogeneity to the system. Because no success criteria are met, we reject the hypothesis that increased apical adhesion alone is the mechanical basis driving epithelial bending in this context.



(a)



(b)

Figure 5.10: **Increased apical adhesion.** *(a) Snapshot of representative simulation with parameters as in Table 5.1, with aam = 5.0. No directed protrusions or bend are observed. (b) Quantification of bend with variable apical adhesion multiplier. Blue region is one standard deviation either side of the mean (blue line) of five individual simulation runs per parameter value, with random noise added to perturb away from any local extremes.*

5.4.2 Active cytoskeletal remodelling

Biological observations show clear and polarised hook-shaped protrusions of the cell cytoskeletons oriented towards the centre of invagination. The biology underlying these protrusions may involve lamellipodia-like structures acting as tractors to pull the leading edge of cells in the direction of the centre of invagination. Lamellipodia are thin sheet-like protrusions of cytoskeletal actin formed by epithelial cells, and they enable cell motility by providing an anchor ahead of the cell against which to pull forwards [1]. While these protrusions do not look like classic lamellipodia seen in cells migrating on a substrate, they are observed to collapse when treated with an inhibitor of actin branching [56], so it is likely that they play a similar role.

As an aside, the cause of the polarisation is currently unknown: we do not know the mechanism by which cells on the left of the invagination hook right, and those on the right hook left. For this study, we simply assume the existence of a left–right polarisation defining, for each cell, an 'inner' and an 'outer' face, with the inner face being the side of the cell closer to the centre of the invagination.

How can we best incorporate such active remodelling into the IB framework? It is tempting to come up with an elaborate algorithm that explicitly extrudes the cell membrane in the inner-apical corner towards the invagination; however, grounding this hypothesis in plausible biology seems difficult. We instead implement remodelling as a simple local reduction in the stiffness of each cell's representative IB, in the inner apical corner. We imagine that this active cytoskeletal remodelling is an increase in the willingness of part of a boundary to deform in response to adhesion with a neighbouring boundary. Specifically, the intracellular force is modified so that $\Phi_i = ias$ whenever a node involved in an intracellular interaction is tagged as 'inner apical', and $\Phi_i = 1$, otherwise. The value ias varies between simulations.

The idea is that this will allow turgor-related force to dominate in the weaker boundary regions, which is the effect of remodelling whose essence we hope to capture. Areas with a reduced stiffness will then react in a more extreme manner to any shape changes resulting from conservation of volume. In addition, in the absence of other factors, it increases the relative importance of the cell–cell interaction forces in that particular region, allowing the zipping together of nodes in a manner relatable to the biological zipping together of adhesive membranes.

Figure 5.11a shows a representative example of such a simulation. Some slight hook shapes are apparent, particularly in the inner-most cells of regions A and A'. These, however, are far from convincing, and the main observation is a clear bend in the basal lamina. The extent of this bend is characterised in Figure 5.11b in the presence of a fixed apical adhesion multiplier and variable inner-apical stiffness, and in Figure 5.11c in the presence of a fixed inner-apical stiffness and variable apical adhesion multiplier. It is clear that the reduced inner apical stiffness is the main driver of bend in the lamina during these simulations, with a clear relationship and low variability between identical simulations (up to random noise). Varying the apical adhesion has a less profound and highly variable impact on the quantity of bend.

The bend, in these simulations, is driven by a buckling about two cells: the outer-most cells of the central region. This is a major drawback of this regime as such behaviour is not observed in the live imaging experiments (see Figure 5.4), where cells maintain a very clear polarity along the apical–basal axis. This leads us to add explicit representation of this polarity within our modelling framework.

A method is required to stiffen certain cells to prevent their buckling. The simplest mechanism to add the necessary polarity to the cells is to identify the 'corner' nodes and add additional springs between them in order to stiffen the four sides of the rectangle. The biological justification for treating these corners as specific entities

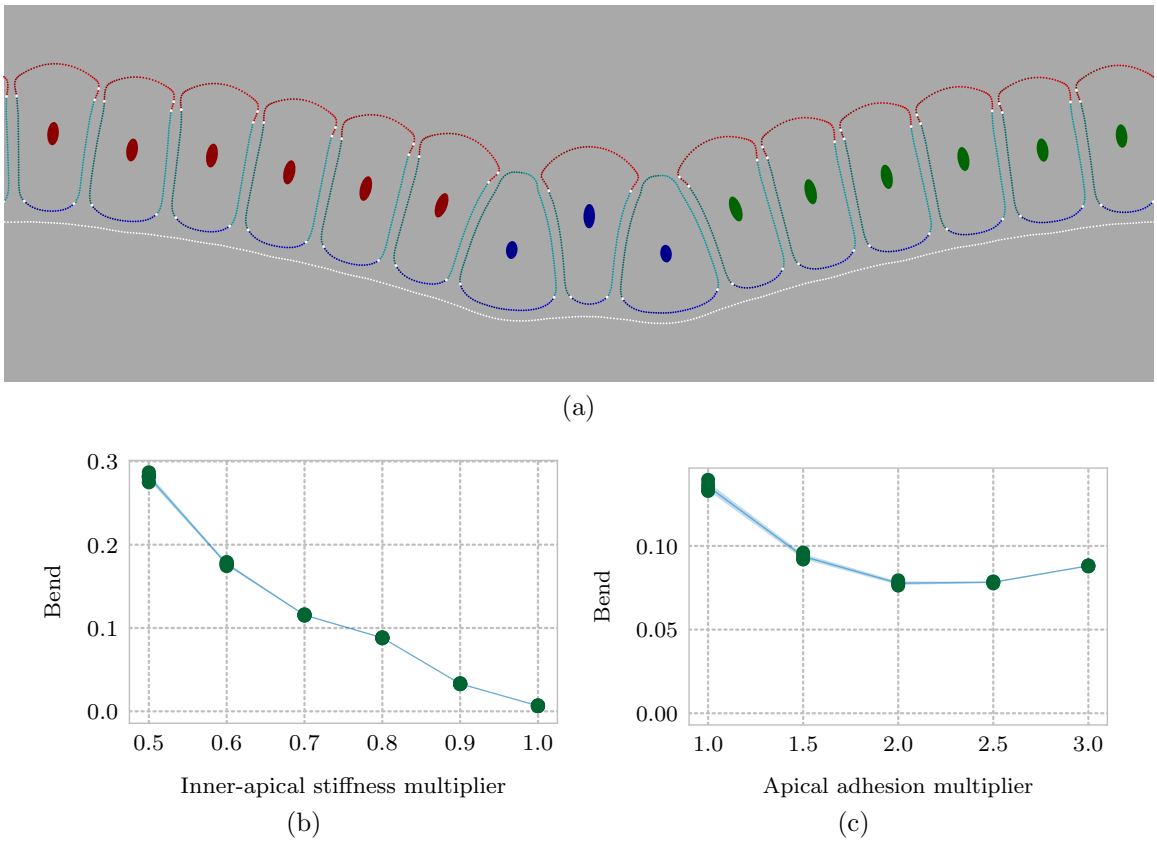


Figure 5.11: **Reduced inner apical stiffness.** *(a) Snapshot of representative simulation with parameters as in Table 5.1, with aam = 3.0 and ias = 0.8. Some small apical protrusions appear, directed into the centre, with a clear bend caused by buckling about central cells. (b)-(c) Blue regions are one standard deviation either side of the mean (blue lines) of five individual simulation runs per parameter value, with random noise added to perturb away from any local extremes. (b) Quantification of bend, with parameters as above, and variable inner-apical stiffness multiplier. (c) Quantification of bend, with parameters as above, and variable apical adhesion multiplier.*

in the model is that these epithelial cells are known to be highly apically-basally polarised, and that these corner nodes therefore allow us to represent the underlying biology (regardless of the mechanisms that generate such polarity). The obvious choice for 'corners' are the left- and right-most apical and basal nodes. An additional spring can then be added between the two apical corners, between the two basal corners, and between each of the two left and right corners, creating a rectangle of additional springs on top of the forces already calculated for the membrane stiffness of each cell. A schematic of this addition can be seen in Figure 5.12b.

This support-force between each pair of corner nodes is modelled as a Hookean spring,

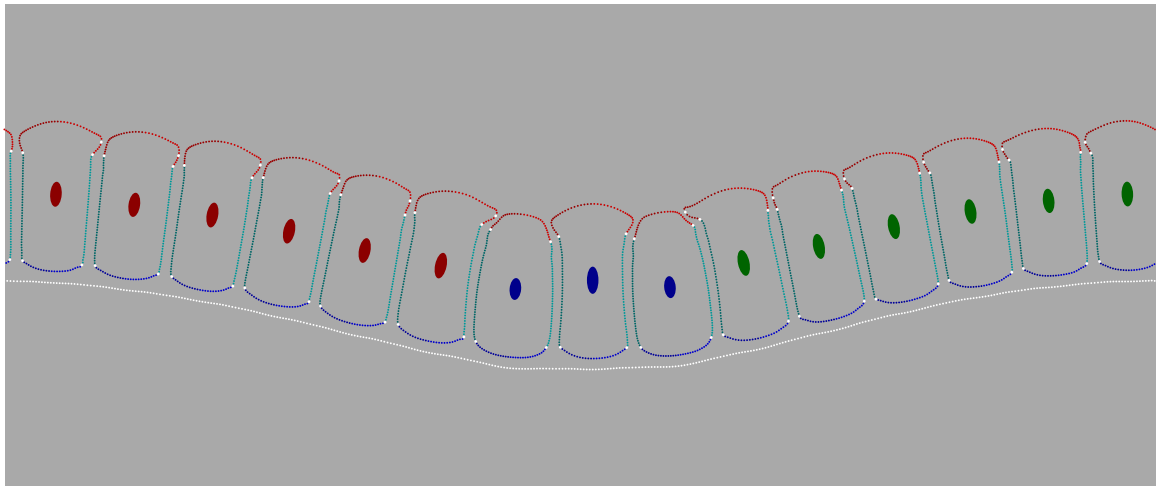
$$\mathbf{F}_s(r) = sss\ \kappa_i(r - l_s), \tag{5.6}$$

where r is the distance between the two corners, l_s is the rest length, κ_i is as in Equation (5.5), and sss is the variable multiplier modulating the quantity of support.

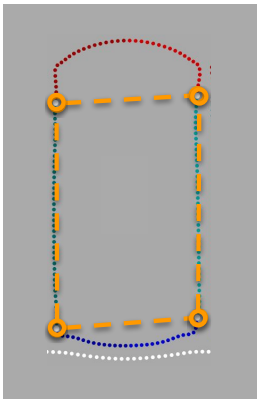
What is an appropriate rest length? Because the number of nodes tagged in each region may change between any two timesteps, choosing an absolute distance is not an option. Instead of a fixed value, there are two obvious distance measurements between two corner nodes: the straight-line distance between the two points in space, and the piecewise-linear distance along the IB between each corner. Setting the rest length to the straight-line distance would result in zero force as the spring would always be (by construction) under no tension. Setting the rest length to the piecewise-linear distance would bias the boundary toward being completely straight. Thus, taking a value for the rest length between these two distances gives a variable scale between no stiffening and complete co-linearity of nodes. This happens in a manner independent of the precise number of nodes tagged in the specific regions.

In total, this additional support adds an additional mechanical inhomogeneity into the system that, for a single cell in isolation, results in a non-round equilibrium shape and, in the palisade monolayer, protects the columnar polarity observed in epithelia. Figure 5.12c characterises the resistance to buckling introduced by the additional tensile elements. To determine whether buckling is present in a given simulation, we take the average ESF of the two outer boundaries in region B as a measure of their aspect ratio. The greater the buckling, the closer this ESF will be to 1 (perfectly round), whereas the starting ESF of all cells in the palisade is roughly 2 (rectangular; twice as tall as wide). As can be seen, with no additional support the ESF remains well below 2, but quickly recovers as sss increases away from 0. Based on this result, we choose the value 0.06 as the default value of sss, ensuring cells remain well away from the buckling regime in further simulations.

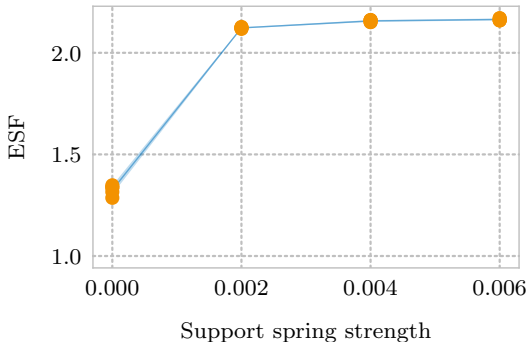
Figure 5.12a shows a representative simulation, with apical adhesion multiplier equal to 3.0, and inner apical stiffness equal to 0.2. There is a clear bend in the basal lamina, and examples of hooked cells directed towards the centre of the invagination. Importantly, no buckling is observed. The extent to which stiffness reduction and increased adhesion influence the bend and skew is characterised in Figures 5.13a to 5.13d.



(a)



(b)



(c)

Figure 5.12: **Explicit cell polarity resists epithelial buckling.** *(a) Snapshot of representative simulation with parameter values given in Table 5.1, with aam = 3.0 and ias = 0.2. Protrusions exist, directed into the centre, with an observed bend no longer hinged about two cells. (b) Schematic of the four tensile elements added to represent explicit cell polarity. (c) Quantification of buckling with parameters as above, and a variable sss. Blue region is one standard deviation either side of the mean (blue line) of five individual simulation runs per parameter value, with random noise added to perturb away from any local extremes.*

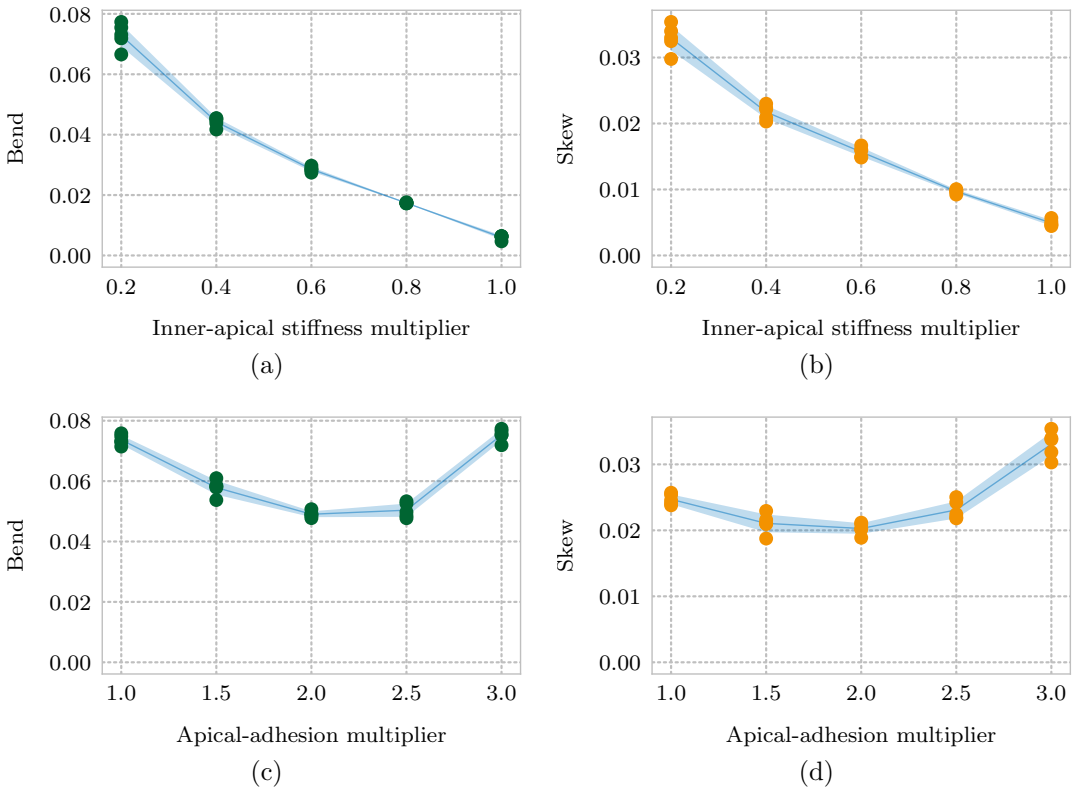


Figure 5.13: **Reduced inner apical stiffness with explicit polarity.** *(a)-(d) Blue regions are one standard deviation either side of the mean (blue lines) of five individual simulation runs per parameter value, with random noise added to perturb away from any local extremes. A fixed support spring strength of* $sss = 0.006$ *is added to each simulation. (a)-(b) Parameters as in Table 5.1 with* $aam = 3.0$ *and variable ias. (a) Quantification of bend with ias. (b) Quantification of skew with ias. (c)-(d) Parameters as in Table 5.1 with* $ias = 0.2$ *and variable aam. (c) Quantification of bend with aam. (d) Quantification of skew with aam.*

Figures 5.13a and 5.13b demonstrate that reducing the inner apical corner stiffness has a direct impact on the bend and skew of simulations in exactly the manner we predict. The greater the reduction in stiffness, the greater both the bend and skew. In biological terms, this implies that active cytoskeletal remodelling, in conjunction with increased apical adhesion, probably plays a role in both invagination and apical protrusions.

Conversely, for a fixed reduction in inner apical stiffness, increased apical adhesion has a less intuitive impact. Figures 5.13c and 5.13d characterise this relationship: epithelial bending is observed in the absence of increased apical adhesion. While an interesting and novel mechanism of bending a palisade of cells, it is worth noting that these simulations do not result in any 'hook' shapes. While, in the absence of increased adhesion, there is still a high measure of skew, this is an artefact of very slightly wedge-shaped cells which exhibit a skew without a hook shape. This is a limitation of the measure that we have adopted for describing hook-shaped cells but, as previously described, a simple visual observation can tell us whether hook shapes exist, and in simulations with hook shapes the skewness measure acceptably quantifies the protrusions for us. For high enough values of aam, both bend and skew measures increase.

In summary, for a fixed value of apical adhesion, reducing the inner apical stiffness increases both the quantity of epithelial bend and the size of apical protrusions. For a fixed value of reduced apical stiffness, increasing the apical adhesion increases the size of apical protrusions; however, increased apical adhesion is not required for simply recapitulating bend in the epithelium.

There are still two drawbacks to these simulations. The main drawback is a lack of any observed outward lean. As a key success criterion, this lack of recapitulation is a problem, one that we return to later. The other drawback is that, under this regime, the apical domains of hook-shaped cells can elongate excessively, and an example of a large elongation can be seen in Figure 5.14. This is at odds with the observed shape of hooked cells, examples of which can be seen in Figure 5.5b. In this case, the combination of reduced inner apical stiffness and increased apical adhesion allows the apical domains to elongate uncontrollably.

5.4.3 Cyclic cytoskeletal remodelling

To be more specific about the trend highlighted with the example in Figure 5.14, some cells protrude a narrow hook which increases the apical–apical area of contact, allowing further protrusion in a zip-like fashion. This can manifest as a small number

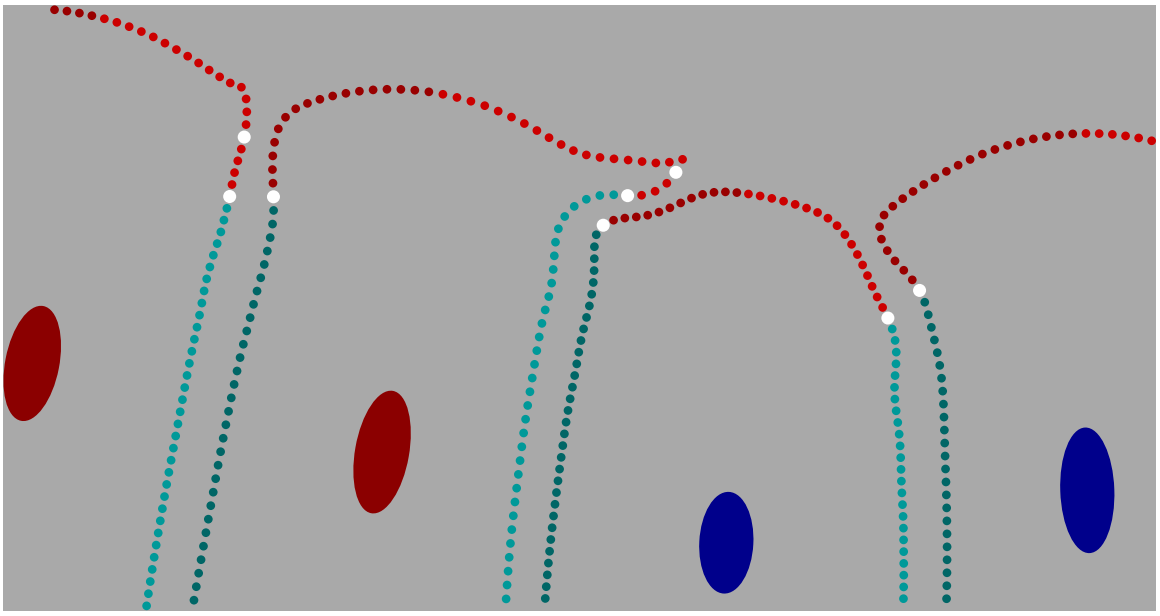


Figure 5.14: **Overextended apical protrusions.** *Example of a simulation where one specific apical protrusion has narrowly elongated. All parameters are as in Table 5.1.*

of cells quickly developing substantial and narrow hooks while neighbouring cells display no protrusive behaviour. To mitigate this deficiency in the model, we again turn to existing biology for cues. A number of biological systems generally, and epithelial morphogenesis in particular, are observed to exhibit oscillatory behaviour. Several morphogenetic processes in *Drosophila* are dependent on actomyosin pulses driven by a biochemical oscillator [12, 54]. A specific example of this is found in *Drosophila* axis elongation, in which the action of actomyosin oscillations are a key component driving cell intercalations [44]. Actomyosin pulses have been incorporated into computational models that have been used to understand the roles of T1 transitions and multicellular rosette formation and resolution on the rate of convergent extension [89, 85]. A further instance of oscillation being demonstrated to be important is mesoderm invagination, which can be rescued in mutants by applying periodic localized mechanical forces generated via magnetic nanoparticles [108]. An oscillatory ratchet has also been identified in the context of mesoderm invagination, driven by the protein Rab35 [74]. This illustrates the necessity of oscillatory contractions for some instances of the morphogenetic deformation.

Such oscillatory behaviour, we hypothesise, could be at play in this system, aiding the stability of the observed protrusive hooks. To investigate this, we introduce a cyclic component to the stiffness reduction. Formally, we suppose that each cell has an internal clock ticking with some frequency f which is fixed across the population.

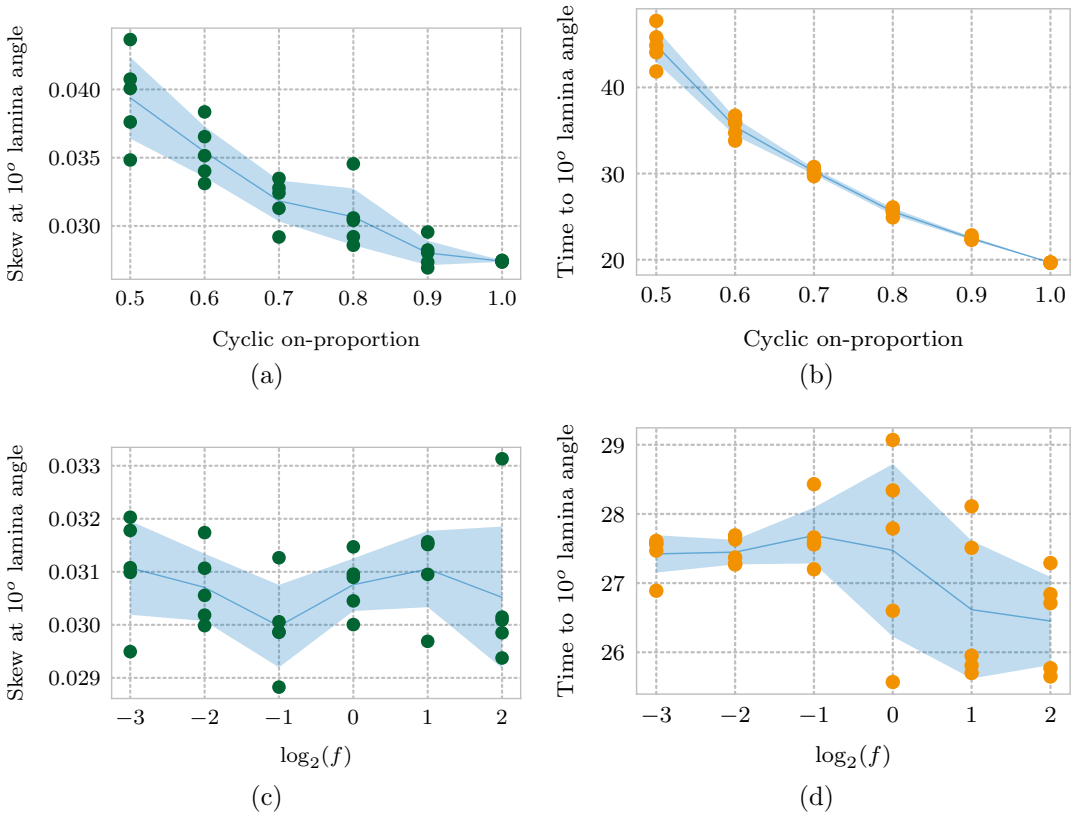


Figure 5.15: **Cyclic stiffness reduction.** *(a)-(d) Blue regions are one standard deviation either side of the mean (blue lines) of five individual simulation runs per parameter value, with random noise added to perturb away from any local extremes. Parameters as in Table 5.1 with aam* $= 3.0$ *and ias* $= 0.02$*. (a)-(b) cop varies as indicated, with* $f = 0.125$*. (a) Quantification of skew at* 10^o *lamina angle with cop. (b) Quantification of the time to* 10^o *lamina angle with cop. (c)-(d)* f*, varies as indicated, with cop* $= 0.75$*. (c) Quantification of skew at* 10^o *lamina angle with* f*. (d) Quantification of the time to* 10^o *lamina angle with* f*.*

We do not suppose any phase coupling between neighbouring cells, so we assign each cell a random phase ϕ in $[0, 2\pi)$. The quantity of stiffness reduction in the inner apical corner, at time t, is then given by

$$s = F(\phi + 2\pi f t), \tag{5.7}$$

such that $\Phi_i(t) = s$ when the node is tagged as inner apical, and $\Phi_i(t) = 1$, otherwise, where F is a periodic function with maximum value 1 and minimum value *ias*. In order to allow variation in the proportion of each period for which the stiffness reduction is on, we choose a simple square waveform for F, and the stiffness reduction is active for some fixed proportion of each cycle (the 'cyclic on-proportion').

A meaningful summary of the original deficiency is simply the average skew; if a larger quantity of cells develop protrusions, the average skew will be high, whereas where a small number of protrusions elongate excessively, the average skew will remain low. Thus, we test the efficacy of inducing cyclic stiffness reduction by running simulations until a fixed lamina angle is achieved and recording the average skew summary as in previous simulations. We pick an arbitrary, but readily achievable, lamina angle of 10^o and proceed with a sensitivity analysis summarised in Figure 5.15.

Figures 5.15a and 5.15b show how the proportion of time for which the stiffness reduction is applied affects the skew and simulation time, respectively. Figure 5.15a shows clearly that the lower the proportion of time for which the stiffness reduction is applied, the higher the average skew of elements. When the stiffness reduction is turned off for a specific simulated cell, any especially large protrusion will relax, allowing the cell to round up while other cells have the ability to protrude further. As expected, the lower the proportion of time the stiffness reduction is applied for, the longer simulations take to achieve the specified 10^o lamina angle. Figures 5.15c and 5.15d show a similar analysis while varying f, the cyclic frequency. No trend is evident over a wide range of values, indicating that the specific frequency of oscillation does not readily impact our summary statistics.

This cyclic stiffness reduction has two interesting consequences. First, it gives us a mechanism to directly influence the quantity of skew observed in the apical protrusions, while preventing the small numbers of runaway protrusions extant in previous simulations. Second, it provides a direct tie between the timescales of apical protrusions and basal-lamina bend. While no such data are yet available, this cyclic stiffness reduction provides the theoretical possibility of matching the timescales to experimentally accessible numbers. We now return to the lack of any outward lean.

5.4.4 Additional diagonal tensile element

Having built a computational framework capable of exploring the key hypotheses based on biological observations, we still have not met the success criterion of outward lean. This leads us to hypothesise an as-yet unidentified mechanism that must play a part in recapitulating the key biological observations. Taking inspiration from the mechanism by which filopodia pull, tension is generated in the cortex of cells [13]. Pulling filopodia have been integrated into cell-based models of convergent extension [9], where the authors impose a point-to-point force caused by the supposed coordinated action of actin filaments. Such filaments have been observed in other contexts, such as in locomoting heart fibroblasts [28] in which filaments mechanically

link the front of the migrating cell to the back; this indicates the possibility that cell-length mechanical connections could play a role in cell shape change.

This leads to the hypothesis that actin filaments may exist from the outer basal region of the cell cortex, around the cell cortex, to the hook-shaped protrusion in the inner apical domain. Should such filaments exist they would provide a clear basis for outward lean, as one would imagine that a diagonal tensile element (Figure 5.16a), in addition to the four existing supports representing explicit epithelial polarity, causing simulated cells to parallelogram rather than lean. The biological basis for such a tensile element is that, when the apical protrusion grows, any existing corner-to-corner actin filaments would naturally experience greater tension. We therefore formulate the following computational hypothesis: a tensile element is added from the outer basal corner to a new 'corner' node in the inner apical region. The chosen node is that which is furthest centrally along the short axis of the element. In other words, it is the point on the cell cortex at the tip of the 'hook' shape. Formally, for each vector $\mathbf{v}$ joining each two adjacent nodes in the IB, the selected node is that which minimises $|\mathbf{s} \cdot \mathbf{v}|$, where $\mathbf{s}$ is the short axis of the element. A schematic of this process is shown in Figure 5.16b. We choose the rest length for this tensile element to be the 'regular' corner-to-corner distance so that, in the absence of an apical hook, the additional element is at mechanical equilibrium. A full schematic of the additional support is shown in Figure 5.16c, which shows the four existing tensile elements representing epithelial polarity, as well as the new element from the outer basal corner to the newly defined corner within the inner apical domain.

Figure 5.17a shows a representative simulation that demonstrates all three success criteria: a bend in the epithelium; hook-shaped cells protruding towards the centre; and clear outward lean of cells. Most importantly, Figure 5.17d demonstrates a dependence of the lean ratio on the additional diagonal spring strength. A clear trend indicates that the lean ratio approaches zero (perfect outward lean), from near unity (no outward lean), as the additional diagonal spring strength is increased. Also of importance is that this relationship has little effect on the quantity of bend observed; Figure 5.17b demonstrates little dependence on bend with additional diagonal spring strength, indicating that our hypothesised mechanism for generating outward lean does not substantially impede the quantity of invagination. On the other hand, the skew is significantly reduced, as seen in Figure 5.17c. This is perhaps not surprising, as the additional diagonal tensile element, while allowing the cells to parallelogram, does directly inhibit the growth of apical protrusions. It is for this reason that an extreme value of the inner-apical stiffness multiplier (0.02) was required to rescue

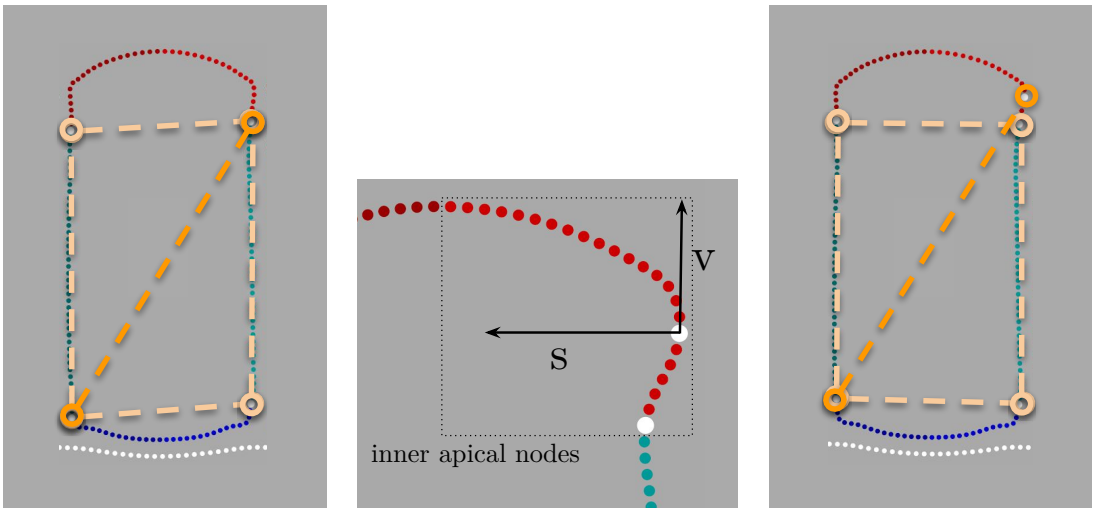


Figure 5.16: **Adding a diagonal support.** *(a) Diagram of diagonal support applied directly between the outer basal corner and the inner apical corner nodes. (b) Definition of the 'beak corner' node. (c) Diagram of diagonal support applied between the outer basal corner and the beak corner.*

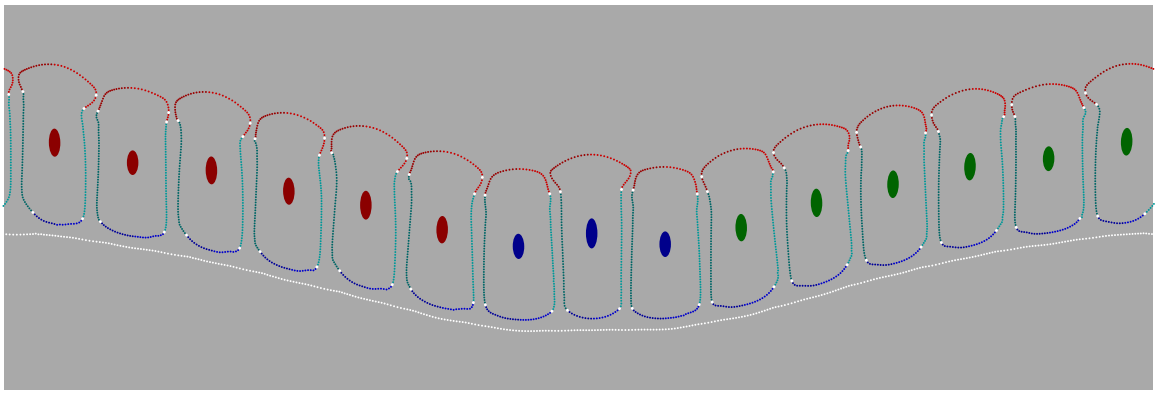
significant protrusions in this context, and that therefore significant skew is observed in simulations with reduced diagonal spring strength.

5.5 Discussion and outlook

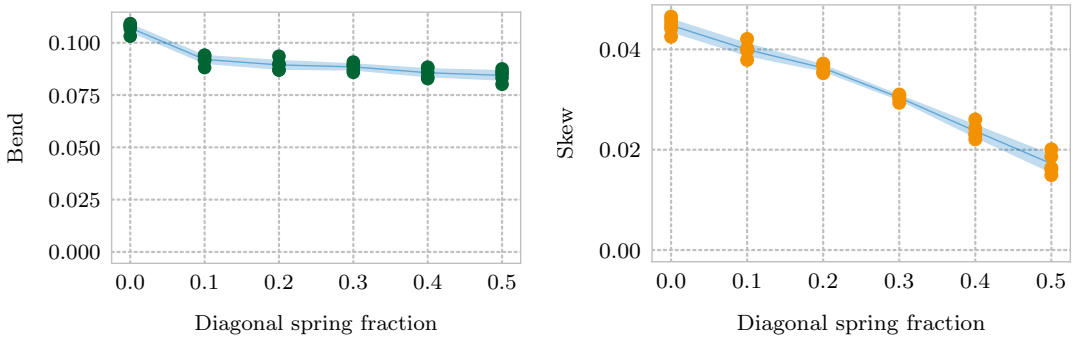
In this chapter we have developed an IB model of a bending epithelium with application to placode morphogenesis. Analysing this model, we have successfully recapitulated three key biological observations made by our experimental collaborators and, in doing so, have generated two testable hypotheses: the presence of cyclic behaviour in the formation of apical protrusions, and the presence of a diagonal tensile element generating additional rigidity in the cell cortex to prevent inward lean.

5.5.1 Summary of model progression

The first hypothesis tested was pure differential adhesion. This failed because the hypothesis did not introduce heterogeneity, and ultimately recapitulated none of the required features. Adding stiffness reduction in the inner apical domain was the first attempt to add heterogeneity, and had partial success. These simulations demonstrated a bend in the lamina, but in all cases this was attributed

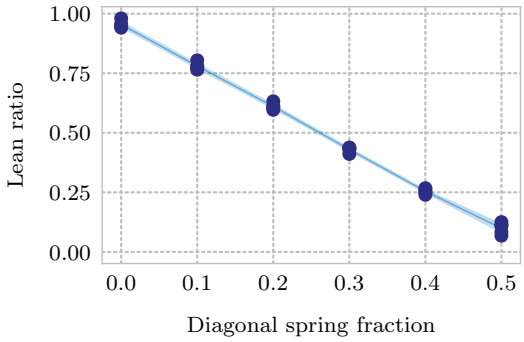


(a)



(b)

(c)



(d)

Figure 5.17: **Additional diagonal tensile element.** *(a) Snapshot of representative simulation. Protrusions exist, directed into the centre, with an observed bend, and an outward lean. (b)-(d) Blue regions are one standard deviation either side of the mean (blue lines) of five individual simulation runs per parameter value, with random noise added to perturb away from any local extremes. Parameters differ from those in Table 5.1 as follow: ias* $= 0.02$*, aam* $= 3.0$*, cop* $= 0.8$*, and variable diagonal spring fraction d. (b) Quantification of bend with d. (c) Quantification of skew with d. (d) Quantification of lean ratio with d.*

overwhelmingly to a buckling of the left- and right-most cells in the central region. In addition to the bend, hook shapes were observed in a number of cells.

Element	Description of simulations when element is removed
Diagonal spring	Loss of vertical telescoping, but can still recapitulate apical protrusions and lamina bend
Support springs	Loss of cell 'polarity'; in particular, a tendency for any bend to 'hinge' about two cells in the central region which round up completely
Cyclic nature of reduced inner apical stiffness	Protrusions are no longer stable; they grow uncontrollably on a timescale incompatible with any observed bend
Reduced inner apical stiffness	Protrusions become smaller and, when occur, are symmetric left/right, not oriented towards a 'centre'
Increased apical domain adhesion	Reduces or removes apical protrusions

Table 5.2: **Summary of model elements.** *The left-hand column lists the main model elements that have been found necessary to recapitulate vertical telescoping. The right-hand column briefly summarises model predictions with successive removal of individual elements.*

These simulations demonstrate that buckling is a limitation of the model, as buckling (with basal bulging) is not observed biologically. To combat the buckling, we added an explicit representation of apical–basal cell polarity: stiffening springs along the lateral, basal and apical domains (between each of four 'corners' calculated during node region-tagging). This reinforcement prevented the buckling and allowed simulations to display substantial bend in the basal lamina along with hook-shaped cells. Some of the hook shapes, however, become unstable and quickly extrude too far, leading to model break-down. In order to stabilise the protrusions, we hypothesise a mechanism of cyclic stiffness reduction: part of the time the membrane has reduced stiffness during which the protrusions are drawn out and locked by apical–apical adhesion, and for a period thereafter the stiffness reduction is turned off, allowing the cell to regain shape and hoist itself up over its neighbour. This allows protrusions to grow more uniformly and allows them to stabilise, increasing the average skew of simulated cells. Finally, we make a second specific hypothesis: a diagonal tensile element, linked to the apical protrusions, that allows cells to lean outwards rather than remaining perpendicular to the basal lamina. Table 5.2 summarises the key findings of this simulation study. These hypotheses were formulated based on our simulation study. A natural next step is to test them experimentally, and Professor Green aims to use ultra-resolution microscopy to look for telltale signs of actin filaments running diagonally around the 'penguin' cells.

5.5.2 Limitations

While we have made progress in addressing the unanswered mechanical questions in this biological system, we have also learned about, and wish to elaborate upon, the limitations of using the IB framework in this context. While we did not attempt to use other potentially suitable frameworks (such as the SEM or many-vertex model), we believe these limitations to be inherent to all such models that rely on the same fundamental concepts.

The largest problem is that of representing extreme cell shapes. In this context, we are referring in particular to the hook-like protrusions inherent in this biological system, but in general any shapes that are highly non-convex. Fundamentally, any modelling framework that considers cells as being unions of points in space interacting by means of some potential function or force relation will prefer 'uniform' shapes such as convex balls. Single cells in isolation will, in all discussed modelling frameworks, take on a circular shape (in two dimensions) or a spherical shape (in three dimensions), unless acted upon by some specific extrinsic driver designed to disrupt this minimal energy geometry. There is no mystery in this, but it is not at all clear how best (in general) to address this difficulty. To some extent this issue was addressed in Section 5.4.2 by the addition of the stiffening springs which, for an IB in isolation, elongated its resting configuration. This, however, is a far cry from allowing cells to robustly attain complex non-convex shapes, which has been a key challenge when modelling the placode system, and this difficulty is a compelling hypothesis for the lack of similar simulation studies in the literature.

In conclusion, this simulation study has pushed the boundaries of out-of-plane modelling of biological processes exhibiting substantial out-of-equilibrium cell shapes. We have made significant progress towards understanding the mechanical basis for invagination in early placode morphogenesis, and we have made two specific biological hypotheses about hitherto unobserved processes that we predict are likely to be acting on the system.

Chapter 6

Discussion and outlook

Developmental biology seeks to understand the processes governing the remarkable transition from the single cell to the fully functioning organism. These processes are complex and highly interconnected, making their analysis difficult. Pursuing a better understanding of such fundamental processes constitutes a meaningful step towards understanding how and why development goes wrong, and how we can potentially intervene with treatments and therapies. Computational modelling has become indispensable in our pursuit of this understanding.

As our ability to probe the biology of developing organisms continues to improve, ever more detailed descriptions of developmental systems at the cellular and subcellular resolutions are becoming available. In tandem, the increase in computing power now available enables the use of ever more detailed biophysical models of cell populations, able to probe unanswered questions in development in ever greater detail.

A gap in the field. This thesis set out to address a specific gap in the field. While several modelling frameworks that incorporate geometrically detailed descriptions of cellular morphodynamics have previously been described (Section 1.2.3), several facts are apparent.

- To date, other than that presented in this thesis, no open source computational implementations exist of any such geometrically detailed cell-based modelling frameworks that adhere to basic best practices of software engineering, a minimal threshold that must be met to have confidence in any results generated by them.

- To date, very few studies have used such geometrically detailed frameworks for anything other than toy problems, with only two notable examples: those of

Tamulonis and colleagues [162], and Sandersius and Newman [143].

- While many large software packages exist in general for cell-based modelling (Section 1.3), only Chaste emphasises software quality, testing and sustainability as a main aim.

- Of the software packages implementing cell-based models, only Chaste implements multiple paradigms in a unified framework; this is key to ensuring that the appropriate, rather than simply the available, tool is used for computational modelling.

Implementation of the IBM. The first step towards filling this gap was to implement and carefully analyse one such geometrically detailed cell-based modelling framework, the IBM. This undertaking has provided a comprehensive account of the model, a method of solution, the computational implementation and an analysis of parameter scaling, giving the firm grounding necessary to utilise it and to accurately compare it to other models. This implementation is detailed in the SIAM Journal on Scientific Computing [27] ensuring that, for the first time, this comprehensive account of the model, together with a careful and tested reference implementation, has been made available to the community.

While this implementation represents a useful contribution to the field, the following future work will add further utility. First, the implementation of the IBM in Chaste relies on the library `fftw`[1] and, due to licensing restrictions, the current implementation cannot be integrated into the 'master' branch of Chaste. Work to rectify this is ongoing and, when complete, will ensure that the IB implementation is maintained and sustained as long as Chaste itself is, ensuing the utility of the code well into the future. Second, several limitations exist in the current IB implementation, and future releases of the code would benefit from enhancements, including these specific changes:

- allowing arbitrarily shaped domains (currently all domains are square, for reasons of convenience with the fluid solver);

- allowing parameters to have dimensionality;

- allowing non-zero net forces (currently the sum of forces over the domain must be zero);

[1] `http://www.fftw.org/`

- implementing the IBM in three dimensions (a longer-term goal, well outside the scope of this thesis).

Careful regard to software best practices. As argued strongly, and backed by empirical research (Chapter 3), the quality and availability of software tools used for academic research is generally low. This leads to low confidence in the results generated by software tools, but also in wasted time reinventing the wheel and checking the work of others. The academic publishing industry, in the past, has not often peer-reviewed software tools, and as a result these have avoided the levels of scrutiny that are applied, for instance, to experimental protocols. The now near-ubiquity of software tools in research is necessitating a change in approach, and journals such as the *Journal of Open Source Software*[2] and *SoftwareX*[3] offer an avenue for academic software tools to be formally assessed for quality and usability.

I have developed and refined these qualities in Chaste, a tool that takes software quality and best practises seriously with an aim of long-term sustainability. A number of contributions I have made to Chaste are detailed in Chapter 3, and these contributions underpin a forthcoming submission to the *Journal of Open Source Software*, on which I will be the first author. There is no shortage of future work in this area as maintaining and improving Chaste, as well as promoting software best practises in academia, are continuous undertakings.

Understanding where the IBM fits in to our toolkit of computational models. Chaste allows many computational modelling paradigms to coexist in a consistent framework. It is for this reason that Osborne and colleagues [118] were able to compare a number of different frameworks on a set of benchmark problems, elucidating the strengths and weaknesses of the different frameworks. These kinds of comparison studies are vital as they inform the selection of appropriate tools for probing specific questions. The study by Osborne and colleagues opens the door for other implementations of modelling frameworks (Section 1.3) to benchmark themselves on certain model problems. Testing multiple frameworks against a set of model problems further aids in ensuring implementations are robust and correct, and that results can be relied upon.

No such work, prior to this thesis, had been undertaken for any of the more geometrically detailed cell-based modelling frameworks. This lack of comparison

[2] `https://joss.theoj.org/`
[3] `https://www.journals.elsevier.com/softwarex`

makes it difficult to be certain of the scenarios under which, say, the IBM is an appropriate computational tool. Chapter 4 takes a stride towards remedying this. By carefully comparing the IBM to the VM, and testing the frameworks side-by-side on the model problem of cell sorting, two avenues have been advanced.

First, fundamental concepts including the determination of cell neighbours and the addition of random noise, had to be implemented in the IB framework. In particular, while implementing a suitable framework for adding random noise to IBM simulations, it became apparent that the solution had implications for every other off-lattice cell-based modelling framework. This synergy is not just pleasing but constitutes a tangible improvement to cell-based models in a wider sense than just the IBM for which it was originally designed. Second, our understanding of the behaviour of the IBM is significantly enhanced, and the outcomes of the cell sorting study shed light on the scenarios under which the IBM is a suitable computational framework for cell-based modelling.

While a substantial and useful start, further work could add additional value to that presented in Chapter 4. Applying the IBM to additional model problems presented by Osborne and colleagues, while out of the scope of this thesis, would lead to further insight into its behaviour and suitability for particular circumstances. In addition, further investigation into the GRF force is warranted. While the current work introduces the necessary spatial correlations, and previous work matches the temporal dynamics to those of a purely diffusive process, a more complete method of adding random noise to simulations ought to include temporal evolution of the GRFs on specified timescales. In addition, finding a more computationally efficient manner in which to calculate or approximate such fields would greatly increase their utility, perhaps allowing their use directly on off-lattice cell populations rather than relying on interpolation on uniform grids.

Gaining insight into a novel question in developmental biology. Finally, while implementing and carefully analysing a computational framework is useful, if the framework cannot make a meaningful contribution to our understanding of biology it will have been a fairly dry exercise. The simulation studies presented in Chapter 5 demonstrate the utility of the IBM in answering a real and relevant question in developmental biology that hinges on out-of-equilibrium cell morphodynamics. No existing computational implementations of cell-based modelling frameworks would have been suitable for this study.

That it has been used successfully to aid our understanding of a novel

morphogenetic process, in direct collaboration with the experimental biologists working on the problem, demonstrates the IBM and our implementation as a useful tool for the scientific enquiry into epithelial dynamics. It is clear that the IBM has the capability to tackle a whole class of problems that less geometrically detailed frameworks cannot, and has the potential to elucidate many more biological processes in the future.

Future research directions. To conclude this thesis, I will outline key areas for future work.

1. Implement reference software tools for additional cell-based modelling frameworks that incorporate increased geometric detail, including the MVM and the SEM. This will further expand the range of tools that can be utilised for cell-based modelling, enabling even wider applicability to biological problems.

2. Conduct more extensive and thorough comparisons between these more geometrically detailed cell-based models and existing models, by applying each to the benchmark problems proposed by Osborne and colleagues. This will aid in model selection, ensuring the most appropriate tool for given biological scenarios can be chosen.

3. Further investigation into the tissue viscoelasticity properties of the IBM and related models should be undertaken. Having a better understanding of the parameter regimes in which certain viscoelasticity properties emerge would add much-needed specificity to model choice given particular biological scenarios to model.

4. The GRF force should be refined to include appropriate temporal correlation, and further work on reducing the runtime cost associated with it would substantially increase its applicability and mainstream utility in the field.

5. Methods to initialise realistic geometries from microscopy data and the ability to add quantitative mechanical information based on biological information should be developed. Such methods would, for instance, replace the Voronoi mesh generation technique in favour of even more realistic initial conditions, and allow mechanical parameters to be selected based on actual protein quantities.

6. Methods should be developed to perform inference and model calibration against quantitative datasets and experiments. Incorporating inference

techniques such as approximate Bayesian computation to infer parameter values from simulations based on experimental data would be a substantial asset when undertaking quantitative computational simulation.

Appendix A

Obtaining the source code

A.1 Chapter 2

The IBM branch of Chaste is available from the publicly available Git repository. To get the latest version, run:

```
git clone https://chaste.cs.ox.ac.uk/git/chaste.git Chaste
git checkout fcooper/immersed_boundary
```

Simulations associated with Chapter 2 take the form of a 'user project' associated with the IBM branch of the Chaste project, entitled `IbNumericsPaper`. From the Chaste source directory, run:

```
cd projects
git clone https://github.com/fcooper8472/IbNumericsPaper.git
```

The project can be compiled by following the guide available here:
`https://chaste.cs.ox.ac.uk/trac/wiki/ChasteGuides/CmakeBuildGuide`

A.2 Chapter 3

Work in this chapter related to Chaste can be found in the IBM branch of Chaste; see Appendix A.1. Other simulations can be found in the `VertexIbComp` user project; see Appendix A.3.

A.3 Chapter 4

Advancements to the IBM and VM models associated with Chapter 4 are available in the IBM branch of the Chaste project. Simulations associated with Chapter 4 take the form of a 'user project' associated with the IBM branch of the Chaste project, entitled `VertexIbComp`. From the Chaste source directory, run:

```
cd projects
git clone https://github.com/fcooper8472/VertexIbComp.git
```

The project can be compiled by following the guide available here:
`https://chaste.cs.ox.ac.uk/trac/wiki/ChasteGuides/CmakeBuildGuide`

A.4 Chapter 5

Advancements to the IBM associated with Chapter 5 are available in the IBM branch of the Chaste project. Simulations associated with Chapter 5 take the form of a 'user project' associated with the IBM branch of the Chaste project, entitled `ToothFormation`. From the Chaste source directory, run:

```
cd projects
git clone https://github.com/fcooper8472/ToothFormation.git
```

The project can be compiled by following the guide available here:
`https://chaste.cs.ox.ac.uk/trac/wiki/ChasteGuides/CmakeBuildGuide`

Appendix B

Technical details

B.1 Implementation details of skewness algorithm (Section 5.3.1)

The difficulty implementing the skewness algorithm for a general polygon centres around efficiently calculating the length of the intersection between a line and a planar shape. By rotating and repositioning the shape, we consider the mass distribution along the axis $\hat{\mathbf{x}}$ and we consider the intersection of vertical lines with the polygon. For a given point $\mathbf{\Gamma}^i$ we can determine other intersections between the vertical line through $\mathbf{\Gamma}^i$ and the polygon by looking at the change in sign of $(\mathbf{\Gamma}^j - \mathbf{\Gamma}^i) \cdot \mathbf{x}$, for $j \neq i$ as, if two consecutive points in the discretisation of the polygon lie either side of the vertical line, the cosine of the angle between the rays will also change sign.

Determining the number of intersections is carried out as above, and intersection points can be determined precisely by a linear interpolation between two neighbouring points with the aforementioned sign-changing property. Careful accounting, however, is still necessary to determine the length of the intersection, as the sign pattern becomes complicated if any of the dot products are zero (up to tolerance).

The computational expense of the algorithm described here is dominated by a comparison, for each point in the polygon, with every other point in the polygon, and is therefore of $\mathcal{O}(N^2)$ for a polygon discretised by N points.

B.2 Implementation details of IB region tagging (Section 5.3.2)

There are two particularly tricky aspects to this algorithm. One is efficiently determining which nodes are within a given interaction distance, and the other is adequately determining the free surface. The first problem is solved, as described in Section 2.5.3, by use of the spatial decomposition algorithm. The interaction

distance, however, for cell–cell interactions turns out to often be smaller than the distance required to reliably identify the 'free surface' in the apical domain. Several nodes that are clearly in the lateral domain (as judged by eye) may be too far apart from other boundaries to participate in a neighbour interaction, and this would cause the algorithm above to identify the apical lamina as being far too extensive. For this reason an additional 'box collection' (computational implementation of the spatial decomposition algorithm defined in, for various reasons, the inopportunely named class `ObsoleteBoxCollection`) is employed, with an interaction distance independent of that used to calculate neighbour interactions.

B.3 Mathematical details of IB remeshing (Section 5.3.3)

To provide a more quantitative understanding of why remeshing is necessary for allowing mechanical inhomogeneity, consider N bearings arranged as vertices of a regular N-gon, each connected to its two neighbours by identical linear springs. Further suppose these bearings are constrained to remain in a fixed circular groove (analogous to the volume-conservation property of the IBM). Given a fixed rest length r, the total potential energy E stored in this configuration of springs is given by

$$E = \sum_N \frac{1}{2}\kappa(l-r)^2 = \frac{N}{2}\kappa(l-r)^2, \tag{B.1}$$

where κ is the spring constant and l is the length of each side of the regular N-gon. It is clear to see that this is the configuration that minimises E.

Next, consider that a number, R, of the springs in the system change in stiffness by a factor α, having new spring constant $\alpha\kappa$, and the system is allowed to evolve to equilibrium. The new configuration holds the following potential energy

$$E_{new} = \frac{N-R}{2}\kappa(l_1-r)^2 + \frac{R}{2}\alpha\kappa(l_2-r)^2, \tag{B.2}$$

where l_1 and l_2 are now the lengths of springs in the $N-R$ original, and R altered, springs, respectively, such that

$$(N-R)l_1 + Rl_2 = Nl. \tag{B.3}$$

To find the preferred values of l_1 and l_2, we can find the minima of E_{new} with respect to l_1 which, after some algebra, yields the relationship

$$l_1 - r = \alpha(l_2 - r). \tag{B.4}$$

We thus see that if $\alpha < 1$, the extension in the unaltered springs is smaller than the extension in the altered springs by precisely the factor with which we altered the spring constant, and thus that the altered springs are longer (assuming all springs are under tension), corresponding to an increase in the distance between the bearings connected by the altered springs. This is undesirable, as the IB functions best in the presence of evenly spaced nodes.

The purpose of remeshing is to evenly redistribute nodes around the boundary. This keeps the boundary out of mechanical equilibrium but with the important property that any node connected to springs with reduced spring constant will be a commensurate amount easier to move, thus providing the required reduction in stiffness for the given region. Put more rigorously, to move a single node in the reduced stiffness region by a particular displacement requires the input of precisely α-times as much energy as if the stiffness were not reduced. This matches what we would intuitively desire from a region of reduced stiffness.

It is worth noting at this point the possibility of avoiding remeshing by simply selecting an appropriately altered rest length between nodes so as to keep the bearings evenly spaced. At equilibrium this would imply a force balance either side of a bearing on the boundary of the region with altered mechanical properties. The consequence of this is that the springs with altered spring constant $\alpha\kappa$ would require an altered rest length r' of

$$r' = \frac{l(\alpha - 1) + r}{\alpha}, \tag{B.5}$$

where l is the length of each spring and r is the unaltered rest length. This has several problematic implications:

- r' has an explicit dependence on l and thus is no longer independent of the resolution at which we represent the IBs;
- for small enough values of α, the altered rest length is negative; an unphysical (if mathematically sound) proposition;
- while the spring constant has been altered by a known factor, as the rest length has also changed there is a non-obvious total change in the ease with which a node can be moved. More formally, moving a node in the reduced stiffness region by a particular displacement no longer requires the input of precisely α-times as much energy, leading to a much less intuitive reduction in stiffness for that region.

The only reasonable solution is therefore to keep the rest length fixed and to implement remeshing. While the linear interpolation algorithm presented in Section 5.3.3 is straightforward to implement, care has been taken to ensure the method `EvenlySpaceAlongPath()` works in full generality with both open and closed paths, which is required when remeshing the basal lamina (an open path with periodic boundary conditions), rather than an IB representing a cell (a closed path).

A more robust method could employ the Kochanek–Bartels interpolating spline [79], which is cubic Hermite spline with parameters *tension*, *bias* and *continuity* altering the resulting shape, which would give greater precision over a simple linear interpolation. The Kochanek–Bartels spline is implemented in VTK, a dependency of Chaste, and could fairly easily be integrated; however, the linear interpolation has proved sufficient for the current work.

References

[1] Alberts B, Johnson A, Lewis J, Morgan D, Raff M, Roberts K, & Walter P. *Molecular biology of the cell*. Garland Science, 6th edition, 2015. ISBN 0815344643.

[2] Almet A A, Hughes B D, Landman K A, Näthke I S, & Osborne J M. A multicellular model of intestinal crypt buckling and fission. *Bull. Math. Biol.*, 80(2):335–359, 2018.

[3] Atwell K, Qin Z, Gavaghan D, Kugler H, Hubbard E J A, & Osborne J M. Mechano-logical model of *C. elegans* germ line suggests feedback on the cell cycle. *Development*, 142(22):3902–11, 2015.

[4] Awile O, Büyükkeçeci F, Reboux S, & Sbalzarini I F. Fast neighbor lists for adaptive-resolution particle simulations. *Comput. Phys. Commun.*, 183(5):1073–1081, 2012.

[5] Awio Web Services. W3Counter: global web stats (`https://www.w3counter.com/globalstats.php`).

[6] Bagchi P. Mesoscale simulation of blood flow in small vessels. *Biophys. J.*, 92(6):1858–1877, 2007.

[7] Basan M, Joanny J F, Prost J, & Risler T. Undulation instability of epithelial tissues. *Phys. Rev. Lett.*, 106(15):158101, 2011.

[8] Belmonte J M, Clendenon S G, Oliveira G M, Swat M H, Greene E V, Jeyaraman S, Glazier J A, & Bacallao R L. Virtual-tissue computer simulations define the roles of cell adhesion and proliferation in the onset of kidney cystic disease. *Mol. Biol. Cell*, 27(22):3673–3685, 2016.

[9] Belmonte J M, Swat M H, & Glazier J A. Filopodial-tension model of convergent-extension of tissues. *PLOS Comput. Biol.*, 12(6):e1004952, 2016.

[10] Bertet C, Sulak L, & Lecuit T. Myosin-dependent junction remodelling controls planar cell intercalation and axis elongation. *Nature*, 429:667–671, 2004.

[11] Biggs L C & Mikkola M L. Early inductive events in ectodermal appendage morphogenesis. *Semin. Cell Dev. Biol.*, 25-26:11–21, 2014.

[12] Blanchard G B, Étienne J, & Gorfinkiel N. From pulsatile apicomedial contractility to effective epithelial mechanics. *Curr. Opin. Genet. Dev.*, 51:78–87, 2018.

[13] Bornschlögl T. How filopodia pull: What we know about the mechanics and dynamics of filopodia. *Cytoskeleton*, 70(10):590–603, 2013.

[14] Bottino D. Modeling viscoelastic networks and cell deformation in the context of the immersed boundary method. *J. Comput. Phys.*, 147(1):86–113, 1998.

[15] Brady S M, Orlando D A, Lee J Y, Wang J Y, Koch J, Dinneny J R, Mace D, Ohler U, Benfey P N, Helariutta Y, Nijsse B, Boekschoten M V, Hooiveld G, Beeckman T, Wagner D, Ljung K, Fleck C, & Weijers D. A high-resolution root spatiotemporal map reveals dominant expression patterns. *Science*, 318(5851):801–806, 2007.

[16] Brett A, Croucher M, Haines R, Hettrick S, Hetherington J, Stillwell M, & Wyatt C. Research Software Engineers: State of the Nation Report 2017. Technical report, 2017.

[17] Bringley T T. *Analysis of the immersed boundary method for Stokes flow.* PhD thesis, New York University, 2008.

[18] Brodland G W. The differential interfacial tension hypothesis (DITH): a comprehensive theory for the self-rearrangement of embryonic cells and tissues. *J. Biomech. Eng.*, 124(2):188, 2002.

[19] Brodland G W. Computational modeling of cell sorting, tissue engulfment, and related phenomena: a review. *Appl. Mech. Rev.*, 57(1):47, 2004.

[20] Butler L C, Blanchard G B, Kabla A J, Lawrence N J, Welchman D P, Mahadevan L, Adams R J, & Sanson B. Cell shape changes indicate a role for extrinsic tensile forces in *Drosophila* germ-band extension. *Nat. Cell Biol.*, 11(7):859–64, 2009.

[21] Chorin A J. Numerical solution of the Navier-Stokes equations. *Math. Comput.*, 22(104):745—-762, 1968.

[22] Chorin A J. On the convergence of discrete approximations to the Navier-Stokes equations. *Math. Comput.*, 23(106):341—-353, 1969.

[23] Chu Y S, Dufour S, Thiery J P, Perez E, & Pincet F. Johnson-Kendall-Roberts theory applied to living cells. *Phys. Rev. Lett.*, 94(2):028102, 2005.

[24] Chwang A T & Wu T Y T. Hydromechanics of low-Reynolds-number flow. Part 2. Singularity method for Stokes flows. *J. Fluid Mech.*, 67(04):787, 1975.

[25] Cooke J & Zeeman E. A clock and wavefront model for control of the number of repeated structures during animal morphogenesis. *J. Theor. Biol.*, 58(2):455–476, 1976.

[26] Cooper F, Robinson M, Hettrick S, Gilchrist A, & Gavaghan D. Oxford research software engineering: survey results (`http://www.cs.ox.ac.uk/projects/RSE/rse_post/survey_results/`), 2018.

[27] Cooper F R, Baker R E, & Fletcher A G. Numerical analysis of the immersed boundary method for cell-based simulation. *SIAM J. Sci. Comput.*, 39(5):B943–B967, 2017.

[28] Cramer L P, Siebert M, & Mitchison T J. Identification of novel graded polarity actin filament bundles in locomoting heart fibroblasts: implications for the generation of motile force. *J. Cell Biol.*, 136(6):1287–305, 1997.

[29] Crampin E J, Gaffney E A, & Maini P K. Reaction and diffusion on growing domains: scenarios for robust pattern formation. *Bull. Math. Biol.*, 61(6):1093–1120, 1999.

[30] Curran S, Strandkvist C, Bathmann J, de Gennes M, Kabla A, Salbreux G, & Baum B. Myosin II controls junction fluctuations to guide epithelial tissue ordering. *Dev. Cell*, 43(4):480–492.e6, 2017.

[31] Davidson L A. Epithelial machines that shape the embryo. *Trends Cell Biol.*, 22(2):82–87, 2012.

[32] Davidson L A. No strings attached: new insights into epithelial morphogenesis. *BMC Biol.*, 10(1):105, 2012.

[33] Dias A S, de Almeida I, Belmonte J M, Glazier J A, & Stern C D. Somites without a clock. *Science*, 343(6172):791–795, 2014.

[34] Dillon R & Othmer H G. A mathematical model for outgrowth and spatial patterning of the vertebrate limb bud. *J. Theor. Biol.*, 197(3):295–330, 1999.

[35] Dillon R, Owen M, & Painter K. A single-cell-based model of multicellular growth using the immersed boundary method. *AMS Contemp Math*, 466:1–15, 2008.

[36] Draelants D, Avitabile D, & Vanroose W. Localized auxin peaks in concentration-based transport models of the shoot apical meristem. *J. R. Soc. Interface*, 12(106):20141407, 2015.

[37] Drasdo D, Hoehme S, & Hengstler J G. How predictive quantitative modelling of tissue organisation can inform liver disease pathogenesis. *J. Hepatol.*, 61(4):951–956, 2014.

[38] Dunn S J, Näthke I S, & Osborne J M. Computational models reveal a passive mechanism for cell migration in the crypt. *PLoS ONE*, 8(11):e80516, 2013.

[39] Edelsbrunner H, Kirkpatrick D, & Seidel R. On the shape of a set of points in the plane. *IEEE Trans. Inf. Theory*, 29(4):551–559, 1983.

[40] EPSRC. Software as an infrastructure. Technical report, 2012.

[41] Exner H E & Hougardy H P. *Quantitative image analysis of microstructures: a practical guide to techniques, instrumentation and assessment of materials.* DGM Informationsgesellschaft, 1988.

[42] Farhadifar R, Röper J C, Aigouy B, Eaton S, & Jülicher F. The influence of cell mechanics, cell-cell interactions, and proliferation on epithelial packing. *Curr. Biol.*, 17(24):2095–104, 2007.

[43] Fedosov D A, Caswell B, & Karniadakis G E. A multiscale red blood cell model with accurate mechanics, rheology, and dynamics. *Biophys. J.*, 98(10):2215–2225, 2010.

[44] Fernandez-Gonzalez R & Zallen J A. Oscillatory behaviors and hierarchical assembly of contractile structures in intercalating cells. *Phys. Biol.*, 8(4):045005, 2011.

[45] Fletcher A G, Cooper F R, & Baker R E. Mechanocellular models of epithelial morphogenesis. *Philos. Trans. R. Soc. B*, 372(1720), 2017.

[46] Fletcher A G, Osborne J M, Maini P K, & Gavaghan D J. Implementing vertex dynamics models of cell populations in biology within a consistent computational framework. *Prog. Biophys. Mol. Biol.*, 113(2):299–326, 2013.

[47] Fletcher A G, Osterfield M, Baker R E, & Shvartsman S Y. Vertex models of epithelial morphogenesis. *Biophys. J.*, 106(11):2291–2304, 2014.

[48] Friel J J, Grande J C, & Hetzner D. *Practical guide to image analysis.* ASM International, Materials Park, Ohio, 2000. ISBN 9780871706881.

[49] Gao X, Arpin C, Marvel J, Prokopiou S A, Gandrillon O, & Crauste F. IL-2 sensitivity and exogenous IL-2 concentration gradient tune the productive contact duration of CD8+ T cell-APC: a multiscale modeling study. *BMC Syst. Biol.*, 10(1):77, 2016.

[50] Gere J M. *Mechanics of materials.* PWS-Kent Pub. Co, 1 edition, 1990. ISBN 0534921744.

[51] Ghaffarizadeh A, Heiland R, Friedman S H, Mumenthaler S M, & Macklin P. PhysiCell: An open source physics-based cell simulator for 3-D multicellular systems. *PLoS Comput. Biol.*, 14(2):e1005991, 2018.

[52] Gibson M C, Patel A B, Nagpal R, & Perrimon N. The emergence of geometric order in proliferating metazoan epithelia. *Nature*, 442(7106):1038–1041, 2006.

[53] Givelberg E & Bunn J. A comprehensive three-dimensional model of the cochlea. *J. Comput. Phys.*, 191(2):377–391, 2003.

[54] Gorfinkiel N. From actomyosin oscillations to tissue-level deformations. *Dev. Dyn.*, 245(3):268–275, 2016.

[55] Graner F & Glazier J A. Simulation of biological cell sorting using a two-dimensional extended Potts model. *Phys. Rev. Lett.*, 69(13):2013–2016, 1992.

[56] Green J. Personal communication.

[57] Griffith B E. Immersed boundary model of aortic heart valve dynamics with physiological driving and loading conditions. *Int. J. Numer. Method Biomed. Eng.*, 28(3):317—-345, 2011.

[58] Griffith B E. On the volume conservation of the immersed boundary method. *Commun. Comput. Phys.*, 12(02):401–432, 2012.

[59] Griffith B E & Luo X. Hybrid finite difference/finite element immersed boundary method. *arXiv:1612.05916v2*, pages 1–32, 2017.

[60] Grogan J A, Connor A J, Markelc B, Muschel R J, Maini P K, Byrne H M, & Pitt-Francis J M. Microvessel Chaste: an open library for spatial modeling of vascularized tissues. *Biophys. J.*, 112(9):1767–1772, 2017.

[61] Guttman A. *R-trees: a dynamic index structure for spatial searching*. ACM, 1984.

[62] Harding M J, McGraw H F, & Nechiporuk A. The roles and regulation of multicellular rosette structures during morphogenesis. *Development*, 141(13):2549–2558, 2014.

[63] Harris A K. Is cell sorting caused by differences in the work of intercellular adhesion? A critique of the Steinberg hypothesis. *J. Theor. Biol.*, 61(2):267–285, 1976.

[64] Harvey D G, Fletcher A G, Osborne J M, & Pitt-Francis J. A parallel implementation of an off-lattice individual-based model of multicellular populations. *Comput. Phys. Commun.*, 192:130–137, 2015.

[65] Herndon T, Ash M, & Pollin R. Does high public debt consistently stifle economic growth? A critique of Reinhart and Rogoff. *Cambridge J. Econ.*, 38(2):257–279, 2014.

[66] Hirashima T, Iwasa Y, & Morishita Y. Dynamic modeling of branching morphogenesis of ureteric bud in early kidney development. *J. Theor. Biol.*, 259(1):58–66, 2009.

[67] Hockney R & Eastwood J. *Computer simulation using particles*. crc Press, 1988.

[68] Hoehme S, Brulport M, Bauer A, Bedawy E, Schormann W, Hermes M, Puppe V, Gebhardt R, Zellmer S, Schwarz M, Bockamp E, Timmel T, Hengstler J G, & Drasdo D. Prediction and validation of cell alignment along microvessels as order principle to restore tissue architecture in liver regeneration. *Proc. Natl. Acad. Sci. U. S. A.*, 107(23):10371–6, 2010.

[69] Hoehme S & Drasdo D. A cell-based simulation software for multi-cellular systems. *Bioinformatics*, 26(20):2641–2, 2010.

[70] Höhme S, Hengstler J G, Brulport M, Schäfer M, Bauer A, Gebhardt R, & Drasdo D. Mathematical modelling of liver regeneration after intoxication with CCl4. *Chem. Biol. Interact.*, 168(1):74–93, 2007.

[71] Hoover A P, Griffith B E, & Miller L A. Quantifying performance in the medusan mechanospace with an actively swimming three-dimensional jellyfish model. *J. Fluid Mech.*, 813:1112–1155, 2017.

[72] Ishimoto Y & Morishita Y. Bubbly vertex dynamics: a dynamical and geometrical model for epithelial tissues with curved cell shapes. *Phys. Rev. E*, 90(5):1–23, 2014.

[73] Jadhav S, Eggleton C D, & Konstantopoulos K. A 3-D computational model predicts that cell deformation affects selectin-mediated leukocyte rolling. *Biophys. J.*, 88(1):96–104, 2005.

[74] Jewett C E, Vanderleest T E, Miao H, Xie Y, Madhu R, Loerke D, & Blankenship J T. Planar polarized Rab35 functions as an oscillatory ratchet during cell intercalation in the *Drosophila* epithelium. *Nat. Commun.*, 8(1):476, 2017.

[75] Kang S, Kahan S, McDermott J, Flann N, & Shmulevich I. Biocellion: accelerating computer simulation of multicellular biological system models. *Bioinformatics*, 30(21):3101–3108, 2014.

[76] Kang S, Kahan S, & Momeni B. Simulating microbial community patterning using Biocellion. pages 233–253. Humana Press, New York, NY, 2014.

[77] Keller R, Davidson L A, & Shook D R. How we are shaped: the biomechanics of gastrulation. *Differentiation*, 71:171–205, 2003.

[78] Kiehart D P, Galbraith C G, Edwards K A, Rickoll W L, & Montague R A. Multiple forces contribute to cell sheet morphogenesis for dorsal closure in *Drosophila*. *J. Cell Biol.*, 149(2):471–490, 2000.

[79] Kochanek D H U, Bartels R H, Kochanek D H U, & Bartels R H. Interpolating splines with local tension, continuity, and bias control. In *Proc. 11th Annu. Conf. Comput. Graph. Interact. Tech. - SIGGRAPH '84*, volume 18, pages 33–41. ACM Press, New York, New York, USA, 1984. ISBN 0897911385.

[80] Köhn-Luque A, de Back W, Yamaguchi Y, Yoshimura K, Herrero M A, & Miura T. Dynamics of VEGF matrix-retention in vascular network patterning. *Phys. Biol.*, 10(6):066007, 2013.

[81] Köppen M, Fernández B G, Carvalho L, Jacinto A, & Heisenberg C P. Coordinated cell-shape changes control epithelial movement in zebrafish and *Drosophila*. *Development*, 133:2671–2681, 2006.

[82] Krüger T, Varnik F, & Raabe D. Efficient and accurate simulations of deformable particles immersed in a fluid using a combined immersed boundary lattice Boltzmann finite element method. *Comput. Math. with Appl.*, 61(12):3485–3505, 2011.

[83] Kursawe J, Baker R E, & Fletcher A G. Impact of implementation choices on quantitative predictions of cell-based computational models. *J. Comput. Phys.*, 345:752–767, 2017.

[84] Kursawe J, Brodskiy P A, Zartman J J, Baker R E, & Fletcher A G. Capabilities and limitations of tissue size control through passive mechanical forces. *PLOS Comput. Biol.*, 11(12):e1004679, 2015.

[85] Lan H, Wang Q, Fernandez-Gonzalez R, & Feng J J. A biomechanical model for cell polarization and intercalation during *Drosophila* germband extension. *Phys. Biol.*, 12(5):056011, 2015.

[86] Le D V, White J, Peraire J, Lim K, & Khoo B. An implicit immersed boundary method for three-dimensional fluid-membrane interactions. *J. Comput. Phys.*, 228(22):8427—-8445, 2009.

[87] Lecuit T & Lenne P F. Cell surface mechanics and the control of cell shape, tissue patterns and morphogenesis. *Nat. Rev. Mol. Cell Biol.*, 8(8):633–44, 2007.

[88] LeVeque R J & Li Z. Immersed interface methods for Stokes flow with elastic boundaries or surface tension. *SIAM J. Sci. Comput.*, 18(3):709–735, 1997.

[89] Lim S, Fernandez-Gonzalez R, & Feng J J. Modeling cell intercalation during *Drosophila* germband extension. *Phys. Biol.*, 2018.

[90] Liu Z & Keller P. Emerging imaging and genomic tools for developmental systems biology. *Dev. Cell*, 36(6):597–610, 2016.

[91] Lloyd S. Least squares quantization in PCM. *IEEE Trans. Inf. Theory*, 28(2):129–137, 1982.

[92] Lubarsky B & Krasnow M A. Tube morphogenesis: making and shaping biological tubes. *Cell*, 112(1):19–28, 2003.

[93] Magno R, Grieneisen V A, & Marée A F. The biophysical nature of cells: potential cell behaviours revealed by analytical and computational studies of cell surface mechanics. *BMC Biophys.*, 8(1):8, 2015.

[94] Mao Y, Tournier A L, Bates P A, Gale J E, Tapon N, & Thompson B J. Planar polarization of the atypical myosin Dachs orients cell divisions in *Drosophila*. *Genes Dev.*, 25(2):131–136, 2011.

[95] Mao Y, Tournier A L, Hoppe A, Kester L, Thompson B J, & Tapon N. Differential proliferation rates generate patterns of mechanical tension that orient tissue growth. *EMBO J.*, 32(21):2790–2803, 2013.

[96] Marin-Riera M, Brun-Usan M, Zimm R, Välikangas T, & Salazar-Ciudad I. Computational modeling of development by epithelia, mesenchyme and their interactions: a unified model. *Bioinformatics*, 32(2), 2015.

[97] Marin-Riera M, Moustakas-Verho J, Savriama Y, Jernvall J, & Salazar-Ciudad I. Differential tissue growth and cell adhesion alone drive early tooth morphogenesis: An ex vivo and in silico study. *PLOS Comput. Biol.*, 14(2):e1005981, 2018.

[98] Martin A C, Kaschube M, & Wieschaus E F. Pulsed contractions of an actin-myosin network drive apical constriction. *Nature*, 457(7228):495–9, 2009.

[99] Matthews D. Papers in economics 'not reproducible' (`https://www.timeshighereducation.com/news/papers-in-economics-not-reproducible`). *Times High. Educ.*, 2015.

[100] Mellor N, Adibi M, El-Showk S, De Rybel B, King J, Mähönen A P, Weijers D, & Bishopp A. Theoretical approaches to understanding root vascular patterning: a consensus between recent models. *J. Exp. Bot.*, 68(1):5–16, 2017.

[101] Merali Z. Computational science: error, why scientific programming does not compite. *Nature*, 467:775–777, 2010.

[102] Merks R M & Glazier J A. A cell-centered approach to developmental biology. *Physica A*, 352(1):113–130, 2005.

[103] Merks R M H, Guravage M, Inzé D, & Beemster G T S. VirtualLeaf: an open-source framework for cell-based modeling of plant tissue growth and development. *Plant Physiol.*, 155(2):656–66, 2011.

[104] Meyer K, Ostrenko O, Bourantas G, Morales-Navarrete H, Porat-Shliom N, Segovia-Miranda F, Nonaka H, Ghaemi A, Verbavatz J M, Brusch L, Sbalzarini I, Kalaidzidis Y, Weigert R, & Zerial M. A predictive 3D multi-scale model of biliary fluid dynamics in the liver lobule. *Cell Syst.*, 4(3):277–290.e9, 2017.

[105] Meyers S. *Effective modern C++ : 42 specific ways to improve your use of C++11 and C++14*. O'Reilly Media, 2014. ISBN 1491908432.

[106] Miller G. Scientific publishing. A scientist's nightmare: software problem leads to five retractions. *Science (80-.).*, 314(5807):1856–7, 2006.

[107] Mirams G R, Arthurs C J, Bernabeu M O, Bordas R, Cooper J, Corrias A, Davit Y, Dunn S J, Fletcher A G, Harvey D G, Marsh M E, Osborne J M, Pathmanathan P, Pitt-Francis J, Southern J, Zemzemi N, & Gavaghan D J. Chaste: an open source C++ library for computational physiology and biology. *PLoS Comput. Biol.*, 9(3):e1002970, 2013.

[108] Mitrossilis D, Röper J C, Le Roy D, Driquez B, Michel A, Ménager C, Shaw G, Le Denmat S, Ranno L, Dumas-Bouchiat F, Dempsey N M, & Farge E. Mechanotransductive cascade of Myo-II-dependent mesoderm and endoderm invaginations in embryo gastrulation. *Nat. Commun.*, 8:13883, 2017.

[109] Mittal R & Iaccarino G. Immersed boundary methods. *Annu. Rev. Fluid Mech.*, 37(1):239–261, 2005.

[110] Moler C. Matrix computation on distributed memory multiprocessors. *Hypercube Multiprocessors*, 86:181–195, 1986.

[111] Morse P M. Diatomic molecules according to the wave mechanics. II. Vibrational levels. *Phys. Rev.*, 34(1):57–64, 1929.

[112] Nagai T & Honda H. A dynamic cell model for the formation of epithelial tissues. *Philos. Mag. Part B*, 81(7):699–719, 2001.

[113] Newman T J. Modeling multicellular systems using subcellular elements. *Math. Biosci. Eng.*, 2(3):613–624, 2005.

[114] Nguyen H & Fauci L. Hydrodynamics of diatom chains and semiflexible fibres. *J. R. Soc. Interface*, 11(96), 2014.

[115] Odell G, Oster G, Burnside B, & Alberch P. A mechanical model for epithelial morphogenesis. *J. Math. Biol.*, 9(3):291–295, 1980.

[116] Odell G M, Oster G, Alberch P, & Burnside B. The mechanical basis of morphogenesis. *Dev. Biol.*, 85(2):446–462, 1981.

[117] Osborne G. Mais Lecture - A New Economic Model (`https://conservative-speeches.sayit.mysociety.org/speech/601526`), 2010.

[118] Osborne J M, Fletcher A G, Pitt-Francis J M, Maini P K, & Gavaghan D J. Comparing individual-based approaches to modelling the self-organization of multicellular tissues. *PLOS Comput. Biol.*, 13(2):e1005387, 2017.

[119] Ozik J, Collier N, Wozniak J, Macal C, Cockrell C, Friedman S, Ghaffarizadeh A, Heiland R, An G, & Macklin P. High-throughput cancer hypothesis testing with an integrated PhysiCell-EMEWS workflow. *bioRxiv*, page 196709, 2018.

[120] Panousopoulou E & Green J B A. Invagination of ectodermal placodes is driven by cell intercalation-mediated contraction of the suprabasal tissue canopy. *PLOS Biol.*, 14(3):e1002405, 2016.

[121] Pathmanathan P, Cooper J, Fletcher A, Mirams G, Murray P, Osborne J, Pitt-Francis J, Walter A, & Chapman S J. A computational study of discrete mechanical tissue models. *Phys. Biol.*, 6(3):036001, 2009.

[122] Pearl E J, Li J, & Green J B A. Cellular systems for epithelial invagination. *Philos. Trans. R. Soc. Lond. B. Biol. Sci.*, 372(1720):20150526, 2017.

[123] Peskin C S. Flow patterns around heart valves: A numerical method. *J. Comput. Phys.*, 10(2):252–271, 1972.

[124] Peskin C S. The immersed boundary method. *Acta Numer.*, 11:479–517, 2002.

[125] Peskin C S & McQueen D M. A general method for the computer simulation of biological systems interacting with fluids. In *Symp. Soc. Exp. Biol.*, volume 49, pages 265–276. 1995.

[126] Peskin C S & Printz B F. Improved volume conservation in the computation of flows with immersed elastic boundaries. *J. Comput. Phys.*, 105(1):33–46, 1993.

[127] Pitt-Francis J, Pathmanathan P, Bernabeu M O, Bordas R, Cooper J, Fletcher A G, Mirams G R, Murray P, Osborne J M, Walter A, Chapman S J, Garny A, van Leeuwen I M M, Maini P K, Rodríguez B, Waters S L, Whiteley J P, Byrne H M, & Gavaghan D J. Chaste: A test-driven approach to software development for biological modelling. *Comput. Phys. Commun.*, 180(12):2452–2471, 2009.

[128] Polyakov O, He B, Swan M, Shaevitz J W, Kaschube M, & Wieschaus E. Passive mechanical forces control cell-shape change during *Drosophila* ventral furrow formation. *Biophys. J.*, 107(4):998–1010, 2014.

[129] Purcell E M E. Life at low Reynolds number. *Am. J. Phys.*, 45(1):3, 1977.

[130] Rauzi M, Hočevar Brezavšček A, Ziherl P, & Leptin M. Physical models of mesoderm invagination in *Drosophila* embryo. *Biophys. J.*, 105(1):3–10, 2013.

[131] Rauzi M, Lenne P F, & Lecuit T. Planar polarized actomyosin contractile flows control epithelial junction remodelling. *Nature*, 468(7327):1110–1114, 2010.

[132] Refsnes Data. HTML5 video (`https://www.w3schools.com/html/html5_video.asp`).

[133] Reinhart C M & Rogoff K S. Growth in a time of debt. *Am. Econ. Rev.*, 100(2):573–578, 2010.

[134] Reinhart C M & Rogoff K S. Reinhart and Rogoff: responding to our critics. *New York Times*, 2013.

[135] Rejniak K & Dillon R. A single cell-based model of the ductal tumour microarchitecture. *Comput. Math. Methods Med.*, 8(1):51–69, 2007.

[136] Rejniak K A. An immersed boundary model of the formation and growth of solid tumors. Technical Report 19, The Ohio State University, 2004.

[137] Rejniak K A. A single-cell approach in modeling the dynamics of tumor microregions. *Math. Biosci. Eng.*, 2(3):643–655, 2005.

[138] Rejniak K A. An immersed boundary framework for modelling the growth of individual cells: an application to the early tumour development. *J. Theor. Biol.*, 247(1):186–204, 2007.

[139] Rejniak K A, Kliman H J, & Fauci L J. A computational model of the mechanics of growth of the villous trophoblast bilayer. *Bull. Math. Biol.*, 66(2):199–232, 2004.

[140] Ryan P. The path to prosperity: a blueprint for American renewal. Fiscal year 2013 budget resolution. Technical report, 2013.

[141] Safferling K, Sütterlin T, Westphal K, Ernst C, Breuhahn K, James M, Jäger D, Halama N, & Grabe N. Wound healing revised: a novel reepithelialization mechanism revealed by in vitro and in silico models. *J. Cell Biol.*, 203(4):691–709, 2013.

[142] Sánchez-Gutiérrez D, Tozluoglu M, Barry J D, Pascual A, Mao Y, & Escudero L M. Fundamental physical cellular constraints drive self-organization of tissues. *EMBO J.*, 35:77–88, 2016.

[143] Sandersius S a & Newman T J. Modeling cell rheology with the subcellular element model. *Phys. Biol.*, 5(1):015002, 2008.

[144] Sandersius S A, Weijer C J, & Newman T J. Emergent cell and tissue dynamics from subcellular modeling of active biomechanical processes. *Phys. Biol.*, 8(4):45007, 2011.

[145] Sawyer J M, Harrell J R, Shemer G, Sullivan-Brown J, Roh-Johnson M, & Goldstein B. Apical constriction: a cell shape change that can drive morphogenesis. *Dev. Biol.*, 341(1):5–19, 2010.

[146] Schliess F, Hoehme S, Henkel S G, Ghallab A, Driesch D, Böttger J, Guthke R, Pfaff M, Hengstler J G, Gebhardt R, Häussinger D, Drasdo D, & Zellmer S. Integrated metabolic spatial-temporal model for the prediction of ammonia detoxification during liver damage and regeneration. *Hepatology*, 60(6):2040–2051, 2014.

[147] Sharpe J. Computer modeling in developmental biology: growing today, essential tomorrow. *Development*, 144(23):4214–4225, 2017.

[148] Smith J L & Schoenwolf G C. Cell cycle and neuroepithelial cell shape during bending of the chick neural plate. *Anat. Rec.*, 218(2):196–206, 1987.

[149] Smith J L & Schoenwolf G C. Role of cell-cycle in regulating neuroepithelial cell shape during bending of the chick neural plate. *Cell Tissue Res.*, 252(3):491–500, 1988.

[150] Spahn P & Reuter R. A vertex model of *Drosophila* ventral furrow formation. *PLoS ONE*, 8(9):e75051, 2013.

[151] Spencer M A, Jabeen Z, & Lubensky D K. Vertex stability and topological transitions in vertex models of foams and epithelia. *Eur. Phys. J. E*, 40(1):2, 2017.

[152] St Johnston D & Sanson B. Epithelial polarity and morphogenesis. *Curr. Opin. Cell Biol.*, 23:540–546, 2011.

[153] Starruß J, de Back W, Brusch L, & Deutsch A. Morpheus: a user-friendly modeling environment for multiscale and multicellular systems biology. *Bioinformatics*, 30(9):1331–1332, 2014.

[154] Steinberg M S. Does differential adhesion govern self-assembly processes in histogenesis? Equilibrium configurations and the emergence of a hierarchy among populations of embryonic cells. *J. Exp. Zool.*, 173(4):395–433, 1970.

[155] Steinberg M S. Differential adhesion in morphogenesis: a modern view. *Curr. Opin. Genet. Dev.*, 17(4):281–286, 2007.

[156] Stroustrup B. *A Tour of C++*. Addison-Wesley Professional, 2nd edition, 2018.

[157] Stroustrup B & Sutter H. C++ Core Guidelines (`http://isocpp.github.io/CppCoreGuidelines/CppCoreGuidelines`), 2018.

[158] Sugimura K, Graner F, & Lenne P F. Measuring forces and stresses in situ in living tissues. *Development*, 143:186–196, 2016.

[159] Sutterlin T, Kolb C, Dickhaus H, Jager D, & Grabe N. Bridging the scales: semantic integration of quantitative SBML in graphical multi-cellular models and simulations with EPISIM and COPASI. *Bioinformatics*, 29(2):223–229, 2013.

[160] Sütterlin T, Tsingos E, Bensaci J, Stamatas G N, & Grabe N. A 3D self-organizing multicellular epidermis model of barrier formation and hydration with realistic cell morphology based on EPISIM. *Sci. Rep.*, 7(1):43472, 2017.

[161] Swat M H, Thomas G L, Belmonte J M, Shirinifard A, Hmeljak D, & Glazier J A. Multi-scale modeling of tissues using CompuCell3D. *Methods Cell Biol.*, 110:325–366, 2012.

[162] Tamulonis C, Postma M, Marlow H Q, Magie C R, de Jong J, & Kaandorp J. A cell-based model of *Nematostella vectensis* gastrulation including bottle cell formation, invagination and zippering. *Dev. Biol.*, 351(1):217–228, 2011.

[163] Tanaka S. Simulation frameworks for morphogenetic problems. *Computation*, 3(2):197–221, 2015.

[164] Tanaka S, Sichau D, & Iber D. LBIBCell: A cell-based simulation environment for morphogenetic problems. *Bioinformatics*, 31(14):2340–2347, 2015.

[165] Teran J M & Peskin C S. Tether force constraints in Stokes flow by the immersed boundary method on a periodic domain. *SIAM J. Sci. Comput.*, 31(5):3404–3416, 2009.

[166] Turing A M. The chemical basis of morphogenesis. *Philos. Trans. R. Soc. B*, 237(641):37–72, 1952.

[167] Verlet L. Computer experiments on classical fluids. I. Thermodynamical properties of Lennard-Jones molecules. *Phys. Rev.*, 159(1):98–103, 1967.

[168] Vogel V & Sheetz M. Local force and geometry sensing regulate cell functions. *Nat. Rev. Mol. Cell Biol.*, 7(4):265–275, 2006.

[169] W3C. Scalable vector graphics (SVG) 1.1 (second edition). Technical report, 2011.

[170] Wabnik K, Kleine-Vehn J, Balla J, Sauer M, Naramoto S, Reinöhl V, Merks R M H, Govaerts W, & Friml J. Emergence of tissue polarization from synergy of intracellular and extracellular auxin signaling. *Mol. Syst. Biol.*, 6(1):447, 2010.

[171] Williams M, Yen W, Lu X, & Sutherland A. Distinct apical and basolateral mechanisms drive planar cell polarity-dependent convergent extension of the mouse neural plate. *Dev. Cell*, 29(1):34–46, 2014.

[172] Wilson G V. Where's the real bottleneck in scientific computing? *Am. Sci.*, 94:5–6, 2006.

[173] Wolpert L. Positional information and the spatial pattern of cellular differentiation. *J. Theor. Biol.*, 25(1):1–47, 1969.

[174] Zhang Y, Thomas G L, Swat M, Shirinifard A, & Glazier J A. Computer simulations of cell sorting due to differential adhesion. *PLoS ONE*, 6(10):e24999, 2011.