

## NUMERICAL ANALYSIS OF THE IMMERSED BOUNDARY METHOD FOR CELL-BASED SIMULATION\*

FERGUS R. COOPER<sup>†</sup>, RUTH E. BAKER<sup>†</sup>, AND ALEXANDER G. FLETCHER<sup>‡</sup>

**Abstract.** Mathematical modelling provides a useful framework within which to investigate the organization of biological tissues. With advances in experimental biology leading to increasingly detailed descriptions of cellular behavior, models that consider cells as individual objects are becoming a common tool to study how processes at the single-cell level affect collective dynamics and determine tissue size, shape, and function. However, there often remains no comprehensive account of these models, their method of solution, computational implementation, or analysis of parameter scaling, hindering our ability to utilize and accurately compare different models. Here we present an efficient, open-source implementation of the immersed boundary (IB) method, tailored to simulate the dynamics of cell populations. This approach considers the dynamics of elastic membranes, representing cell boundaries, immersed in a viscous Newtonian fluid. The IB method enables complex and emergent cell shape dynamics, spatially heterogeneous cell properties, and precise control of growth mechanisms. We solve the model numerically using an established algorithm, based on the fast Fourier transform, providing full details of all technical aspects of our implementation. The implementation is undertaken within Chaste, an open-source C++ library that allows one to easily change constitutive assumptions. Our implementation scales linearly with time step, and subquadratically with mesh spacing and immersed boundary node spacing. We identify the relationship between the immersed boundary node spacing and fluid mesh spacing required to ensure fluid volume conservation within immersed boundaries, and the scaling of cell membrane stiffness and cell-cell interaction strength required when refining the immersed boundary discretization. Finally, we present a simulation study of a growing epithelial tissue to demonstrate the applicability of our implementation to relevant biological questions, highlighting several features of the IB method that make it well suited to address certain questions in epithelial morphogenesis.

**Key words.** immersed boundary method, cell-based modelling, convergence, Chaste

**AMS subject classifications.** 65M06, 76D05, 76M20, 76M22, 92C15, 92C17, 92C37

**DOI.** 10.1137/16M1092246

**1. Introduction.** The collective dynamics of populations of cells play a key role in tissue development and self-renewal, as well as in disease. Mathematical modelling of these systems is challenging due to the wide range of behaviors displayed over different time and length scales, necessitating increasingly sophisticated “multiscale” approaches [4]. Such models seek to gain insight into emergent behaviors where the coordinated action of cell-scale processes, such as the localization of membrane-bound planar cell polarity proteins, can combine to effect striking tissue-level morphogenetic changes, such as convergent extension, in a variety of developing epithelial tissues [42].

Molecular and live-imaging techniques allow tissues to be probed at ever finer scales, supporting the use of modelling frameworks within which hypotheses spanning from the subcellular to the tissue scales may be tested. A range of such models have

\*Submitted to the journal’s Computational Methods in Science and Engineering section September 2, 2016; accepted for publication (in revised form) July 24, 2017; published electronically October 3, 2017.

<http://www.siam.org/journals/sisc/39-5/M109224.html>

**Funding:** The first author’s work was supported by the Engineering and Physical Sciences Research Council (EPSRC) under grant EP/G03706X/1.

<sup>†</sup>Wolfson Centre for Mathematical Biology, Mathematical Institute, University of Oxford, Oxford, OX2 6GG, UK (fergus.cooper@maths.ox.ac.uk, ruth.baker@maths.ox.ac.uk).

<sup>‡</sup>School of Mathematics and Statistics, University of Sheffield, Sheffield, S3 7RH, UK, and The Bateson Centre, University of Sheffield, Western Bank, Sheffield, S10 2TN, UK (A.G.Fletcher@sheffield.ac.uk).

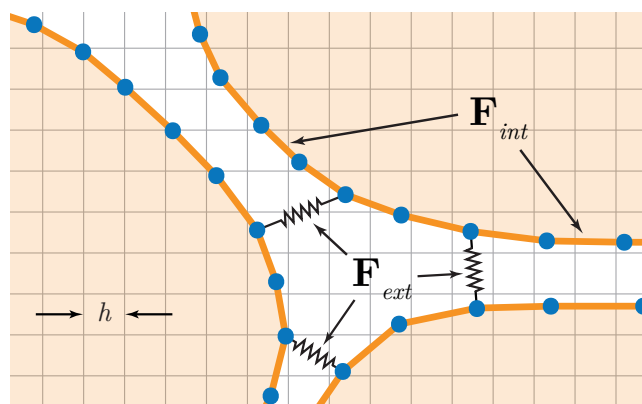


FIG. 1. Schematic of immersed boundaries. Circular nodes represent an off-lattice discretization of the immersed boundary contours. The regular grid behind the boundaries represents points on which a viscous Newtonian fluid, ubiquitous across the domain, is discretized. Adhesion links, specified as explicit force terms, exist between nodes within each immersed boundary, as well as between neighboring boundaries. The terms  $h$ ,  $\mathbf{F}_{int}$ , and  $\mathbf{F}_{ext}$  are defined in the main text.

recently been developed, from vertex models that approximate each cell geometrically by a polygon or polyhedron representing the cell's membrane [9] to subcellular element models that allow for more arbitrary cell shapes [36, 37]. Yet a firm mathematical foundation for the analysis of such models, which is required for confidence in the conclusions drawn from them, remains lacking. To help address this, we present a detailed examination of the immersed boundary (IB) method, which forms the basis for one such class of models, and a computational implementation thereof, designed to study interacting populations of eukaryotic cells.

The IB method 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 [25]. 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 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 immersed boundary 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 [6] or multicellular regions of tissue [5]. Inter- and intracellular 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 immersed boundary. A schematic of parts of three neighboring immersed boundaries is shown in Figure 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.

The IB method 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 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 vertex models [30, 38] and spheroid models [16]. Second, volume is preserved within any given closed contour of the fluid, unless specifically altered by fluid sources or sinks. Thus, the IB method allows for the study of regulated processes that affect cell size, such as cell growth, shrinkage, division, and death. Third, implementations of the IB method typically have a small number of parameters. As shown in section 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 parametrizing an IB method implementation with numerical values estimated from various experimental studies [34]. Finally, unlike numerical schemes that employ structured or unstructured grids conforming to the immersed body, in the IB method the fluid is discretized using a regular Cartesian grid that may be generated with ease. This allows a relatively simple numerical scheme, discussed in subsection 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 IB method, including a thorough treatment of the underlying mathematics in a general three-dimensional setting by Peskin [26]. Biological applications include those by Rejniak, who uses an IB method implementation to investigate the growth of solid tumors under differing geometric configurations, initial conditions, and tumor progression models [33, 34]. The same authors have investigated the mechanics of the bilayer of trophoblasts in the developing placenta [35]. Rejniak and Dillon employ a similar framework to explain the variety of different architectural forms in intraductal tumors [32]. Dillon and Othmer use an IB method to model spatial patterning of the vertebrate limb bud [5], and an IB framework for tackling general morphogenetic problems is presented by Tanaka, Sichau, and Iber [40]. Cell deformation is investigated by several authors; by Bottino in the context of passive actin cytoskeletal networks [1], by Jadhav, Eggleton, and Konstantopoulos with a three-dimensional implementation in the context of cell rolling [18], and by Le et al., who use a three-dimensional framework to investigate the deformation of red blood cells [19]. Also in three dimensions is a comprehensive model of the cochlea by Givelberg and Bunn [10], in a very different Reynolds number regime, demonstrating the versatility and range of IB methods. More recently, IB methods in three dimensions have been applied to ever more complex geometries, and studies include an investigation of the hydrodynamics of diatom chains [24], a model of actively swimming jellyfish [17], and a study of aortic heart valve dynamics [11]. These studies utilize the IBAMR implementation.<sup>1</sup> Finally, a review by Mittal and Iaccarino gives excellent background on the method and cites many other examples of its use across various, including nonbiological, application areas [22].

While, collectively, these papers provide an excellent overview of the IB method 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 IB method implementation aimed specifically at describing the collective dynamics of multicellular tissues. We provide a free, open-source implementation of the IB method complete with example simulations: we build on the established Chaste library [21, 29] to ensure that the code is robust and well-tested; we present the code

<sup>1</sup><https://github.com/IBAMR>

necessary to reproduce all figures in this paper; and we conduct a thorough numerical analysis detailing how parameters scale with respect to each other in order to build a recipe allowing consistent parametrization of models. It is worth noting that, while there are a number of IB 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 IB framework itself such as adaptive mesh refinement, have been instrumental in undertaking several of the three-dimensional studies mentioned above [13]. 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 IB framework, in order to address specific questions in the field of epithelial morphogenesis.

The remainder of this paper is structured as follows. Sections 2 to 4 give details of the IB method, its discretization, and a numerical solution using a fast Fourier transform algorithm. Section 5 outlines the C++ implementation in Chaste. Section 6 details a numerical analysis demonstrating that the computational implementation converges and elaborating on how parameters scale relative to each other. Section 7 provides a prototype simulation study and details specific biological questions that the IB method is well suited to explore. Section 8 concludes with a discussion of the choices made in our implementation and future work in this area.

**2. Immersed boundary method 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

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

$$(1b) \quad \nabla \cdot \mathbf{u} = s,$$

where  $\rho$  and  $\mu$  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  $N$  nonoverlapping closed curves in the fluid, which we will refer to as immersed boundaries and which we think of as representing cell membranes. We associate uppercase Roman indices with the immersed boundaries and lowercase Roman indices with the fluid domain  $\Omega$ . Let  $\Gamma_1, \dots, \Gamma_N$  denote the collection of immersed boundaries, and let each immersed boundary  $\Gamma_k$  be parametrized by  $\gamma_k$ . Further, let

$$(2) \quad \Gamma = \bigcup_{k=1}^N \Gamma_k$$

denote the union of these immersed boundaries, parametrized by  $\gamma$ , which is composed of  $\gamma_1, \dots, \gamma_N$  in the natural way. Let  $\mathbf{X} = \mathbf{X}(\gamma_k, t)$  denote the coordinates of the  $k$ th

immersed boundary, and let  $\mathbf{X} = \mathbf{X}(\gamma, t)$  be the combined coordinates of all immersed boundaries.

We denote the resultant force acting on the immersed boundaries 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 4. We relate the resultant force on the immersed boundaries to the body force acting on the fluid through the relation

$$(3) \quad \mathbf{f}(\mathbf{x}, t) = \int_{\Gamma} \mathbf{F}(\gamma, t) \delta(\mathbf{x} - \mathbf{X}(\gamma, t)) d\gamma = \sum_{k=1}^N \int_{\Gamma_k} \mathbf{F}(\gamma_k, t) \delta(\mathbf{x} - \mathbf{X}(\gamma_k, t)) d\gamma_k,$$

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

The immersed boundaries are assumed to move due to the fluid flow without slipping, so that a point along  $\Gamma$  moves at precisely the local fluid velocity:

$$(4) \quad \frac{\partial \mathbf{X}(\gamma, t)}{\partial t} = \mathbf{u}(\mathbf{X}(\gamma, t)) = \int_{\Omega} \mathbf{u}(\mathbf{x}, t) \delta(\mathbf{x} - \mathbf{X}(\gamma, t)) d\mathbf{x}.$$

Thus, the velocity of an arbitrary immersed boundary 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 4, but for now we consider  $s$  as an arbitrary (but known) scalar field.

**3. Nondimensionalization.** We nondimensionalize 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

$$(5) \quad \mathbf{x} = L \mathbf{x}^*, \quad \mathbf{u} = U \mathbf{u}^*, \quad t = \frac{L}{U} t^*, \quad p = \frac{U\mu}{L} p^*, \quad s = \frac{U}{L} s^*, \quad \mathbf{f} = \frac{U^2}{L} \mathbf{f}^*, \quad \nabla = \frac{1}{L} \nabla^*$$

into (1a) and (1b) and dropping the stars yields

$$(6a) \quad \frac{\partial \mathbf{u}}{\partial t} + (\mathbf{u} \cdot \nabla) \mathbf{u} + \frac{1}{Re} \left( \nabla p - \nabla^2 \mathbf{u} - \frac{1}{3} \nabla s \right) - \mathbf{f} = 0,$$

$$(6b) \quad \nabla \cdot \mathbf{u} = s,$$

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  $Re \rightarrow 0$  in (6a) and, assuming the body force,  $\mathbf{f}$ , to be of order  $1/Re$ , obtain the Stokes equations

$$(7a) \quad \nabla^2 \mathbf{u} - \nabla p + \frac{1}{3} \nabla s + \mathbf{f} = 0,$$

$$(7b) \quad \nabla \cdot \mathbf{u} = s.$$

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

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  $Re \approx 10^{-4}$  [31]. Tanaka [39] estimates 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  $\mu/\rho = 10^2[m^2s^{-1}]$ . Rejniak, Kliman, and Fauci [35] arrive at  $Re \approx 10^{-9}$  by considering the length scale to be the size of a large cytotrophoblastic cell ( $20\mu m$ ) and a characteristic velocity of  $30\mu m$  in 24 hours.

Equations (7a) and (7b) are computationally less expensive than the full Navier–Stokes to solve; for example, their linearity permits the use of efficient Green’s function methods [3]. This raises the question of the circumstances under which it is appropriate to assume Stokes flow for the IB fluid component, as described in [2, 20]. Here, we choose to solve the full Navier–Stokes equations, the reasons for which are discussed in section 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 nondimensional Navier–Stokes equations (6a) and (6b), we nondimensionalize (3) and (4) using the rescaled parameters

$$(8) \quad \mathbf{X} = L \mathbf{X}^*, \quad \mathbf{F} = \frac{U^2}{L} \mathbf{F}^*, \quad \gamma = L \gamma^*,$$

dropping the stars, as before.

**4. Discretization.** We solve the coupled problem, consisting of (3), (4), (6a), and (6b), numerically, as follows. The immersed boundaries are discretized into a finite union of points (small circles in Figure 1) that we call *nodes*. The fluid domain  $\Omega$  is discretized onto a regular Cartesian grid (square lattice in Figure 1) that we refer to as the *mesh*. In our nondimensional coordinates,  $\Omega = [0, 1] \times [0, 1]$  is discretized with  $N \times N$  grid points with mesh spacing  $h$ . We must also discretize (3) relating the force  $\mathbf{F}$  on the immersed boundaries with the body force  $\mathbf{f}$  acting on the fluid, and (4) relating the fluid and node velocities.

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

**4.1. Discrete Dirac delta function.** In the discretized system, the fluid and immersed boundaries 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 immersed boundary 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 IB method implementations are detailed by Mittal and Iaccarino [22].

Here, we make the common choice of a trigonometric function, used in several other IB method implementations [5, 34, 35], given by

$$(9) \quad \delta_h(\mathbf{x}) = \frac{1}{h^2} \phi\left(\frac{x}{h}\right) \phi\left(\frac{y}{h}\right),$$

where  $h$  is the mesh spacing, and the function  $\phi$  is given by

$$(10) \quad \phi(r) = \begin{cases} \frac{1}{4} \left(1 + \cos\left(\frac{\pi r}{2}\right)\right), & |r| \leq 2, \\ 0 & \text{otherwise.} \end{cases}$$

This choice of  $\phi$  differs from, but takes extremely similar numerical values to, that derived and used by Peskin [26]. 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 8).

We also note that, due to the bounded support of both functions,  $\delta_h(\mathbf{x})$  will only ever be nonzero at the  $4 \times 4$  mesh points closest to any given node. The choice of support size is discussed by Peskin [26] and is made purely on computational grounds: one could choose a delta function approximation with wider support, but each node on an immersed boundary would then interact with many more mesh points, slowing down the computation.

**4.2. Discretization of immersed boundaries.** We discretize each immersed boundary  $\Gamma_k$  by a set of  $N_k$  nodes whose positions are given by  $\mathbf{\Gamma}_k^1, \dots, \mathbf{\Gamma}_k^{N_k}$ . 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 immersed boundary is equal to the initial node spacing,  $\gamma_k^{max}/N_k$ . Since we impose the condition that each immersed boundary forms a closed contour, we have  $\mathbf{\Gamma}_k^{N_k+1} = \mathbf{\Gamma}_k^1$ .

**4.3. Discrete force relations.** We are now in a position to define the resultant force  $\mathbf{F}$  acting on the immersed boundaries. 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 immersed boundary.

We will consider the resultant force on each node as the combination of two types of force: “internal” forces, which depend on the positions of other nodes in the same immersed boundary; and “external” forces, which depend on the positions of nodes in different immersed boundaries. 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 neighboring cells. We represent both these mechanical interactions by linear springs, following previous IB method implementations [5, 33, 34, 35, 40], 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 neighbors by a linear spring of stiffness  $\kappa_{int}$  and rest length  $l_{int}$ . The internal force acting on node  $\mathbf{\Gamma}_k^p$  is thus given by

$$(11) \quad \mathbf{F}_{int}(\mathbf{\Gamma}_k^p, t) = \kappa_{int} \sum_{j=p \pm 1 \bmod N_k} \frac{\mathbf{\Gamma}_k^j - \mathbf{\Gamma}_k^p}{\|\mathbf{\Gamma}_k^j - \mathbf{\Gamma}_k^p\|} \left( \|\mathbf{\Gamma}_k^j - \mathbf{\Gamma}_k^p\| - l_{int} \right).$$

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

$$(12) \quad \mathbf{F}_{ext}(\mathbf{\Gamma}_k^p, t) = \kappa_{ext} \sum_{\substack{q=1 \\ q \neq k}}^N \sum_{j=1}^{N_q} H(d_{ext} - \|\mathbf{\Gamma}_q^j - \mathbf{\Gamma}_k^p\|) \frac{\mathbf{\Gamma}_q^j - \mathbf{\Gamma}_k^p}{\|\mathbf{\Gamma}_q^j - \mathbf{\Gamma}_k^p\|} \left( \|\mathbf{\Gamma}_q^j - \mathbf{\Gamma}_k^p\| - l_{ext} \right),$$

where the outer sum runs over all other immersed boundaries, 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 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,

$$(13) \quad \mathbf{F}(\mathbf{\Gamma}_k^p, t) = \mathbf{F}_{int}(\mathbf{\Gamma}_k^p, t) + \mathbf{F}_{ext}(\mathbf{\Gamma}_k^p, t).$$

**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 (6a) and (6b) numerically. We use the following numerical scheme, described first by Peskin and McQueen [27] and later, with the addition of fluid sources, by Dillon and Othmer [5], where the sums are taken over the two dimensions,  $d \in \{1, 2\}$ :

$$(14) \quad \frac{\mathbf{u}^{n+1} - \mathbf{u}^n}{\Delta t} + \sum_d u_d^n D_d^\pm \mathbf{u}^n + \frac{1}{Re} \left( \mathbf{D}^0 p^{n+1} - \sum_d D_d^+ D_d^- \mathbf{u}^{n+1} - \frac{1}{3} \mathbf{D}^0 s^n \right) - \mathbf{f}^n = 0,$$

$$(15) \quad \mathbf{D}^0 \cdot \mathbf{u}^{n+1} = s^n,$$

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

$$(16) \quad (D_d^+ \phi)(\mathbf{x}) = \frac{\phi(\mathbf{x} + h\mathbf{e}_d) - \phi(\mathbf{x})}{h},$$

$$(17) \quad (D_d^- \phi)(\mathbf{x}) = \frac{\phi(\mathbf{x}) - \phi(\mathbf{x} - h\mathbf{e}_d)}{h},$$

$$(18) \quad (D_d^0 \phi)(\mathbf{x}) = \frac{\phi(\mathbf{x} + h\mathbf{e}_d) - \phi(\mathbf{x} - h\mathbf{e}_d)}{2h},$$

$$(19) \quad \mathbf{D}^0 = (D_x^0, D_y^0),$$

$$(20) \quad u_d^n D_{dd}^\pm = \begin{cases} u_d^n D_d^- & \text{if } u_d^n > 0, \\ u_d^n D_d^+ & \text{if } u_d^n < 0, \end{cases}$$

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

**4.5. Discretization of force relation.** We discretize (3), relating the force on the fluid to the force on the immersed boundaries, as follows. For each point  $\mathbf{x}$  in the fluid mesh, we sum the force contributions from every immersed boundary node using the discrete delta function to assign the appropriate weight,

$$(21) \quad \mathbf{f}^n(\mathbf{x}) = \sum_{k=1}^N \left( \sum_{j=1}^{N_k} \mathbf{F}^n(\mathbf{\Gamma}_k^j) \delta_h(\mathbf{x} - \mathbf{\Gamma}_k^j) \Delta\gamma_k \right),$$

where the outer sum runs over the  $N$  immersed boundaries, the inner sum runs over the  $N_k$  nodes in the  $k$ th immersed boundary, and  $\Delta\gamma_k$  is the length element associated with the  $k$ th immersed boundary.

**4.6. Discretization of position-updating relation.** For simplicity, we discretize (4) using a forward Euler scheme. At the  $n$ th time step, a given immersed boundary node  $\mathbf{\Gamma}_k^j$  is displaced by  $\Delta t \mathbf{u}^n(\mathbf{\Gamma}_k^j)$ . Since, 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  $4 \times 4$  fluid mesh points closest to  $\mathbf{\Gamma}_k^j$ . The discretized relation for updating node locations is therefore given by

$$(22) \quad \left(\mathbf{\Gamma}_k^j\right)^{n+1} = \left(\mathbf{\Gamma}_k^j\right)^n + \Delta t \sum_{\mathbf{x} \in \mathcal{N}(\mathbf{\Gamma}_k^j)} \mathbf{u}^{n+1}(\mathbf{x}) \delta_h(\mathbf{x} - \mathbf{\Gamma}_k^j) h^2,$$

where  $\mathcal{N}(\mathbf{\Gamma}_k^j)$  represents the  $4 \times 4$  fluid mesh points nearest  $\mathbf{\Gamma}_k^j$  (the only points with nonzero contributions, due to the implementation of  $\delta_h$ ).

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

Suppose there are  $M$  combined sources and sinks,  $s_1, \dots, s_M$ , with  $M > N$ , located at the positions  $\mathbf{s}_1, \dots, \mathbf{s}_M$ . 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

$$(23) \quad s(\mathbf{x}) = \sum_{k=1}^M T_k \delta_h(\mathbf{x} - \mathbf{s}_k).$$

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

$$(24) \quad (\mathbf{s}_k)^{n+1} = (\mathbf{s}_k)^n + \Delta t \sum_{\mathbf{x} \in \mathcal{N}(\mathbf{s}_k)} \mathbf{u}^{n+1}(\mathbf{x}) \delta_h(\mathbf{x} - \mathbf{s}_k) h^2,$$

where  $\mathcal{N}(\mathbf{s}_k)$  represents the  $4 \times 4$  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 subsection 5.2. Note also that the number of extra “balancing sources” is not fixed, and this is discussed in section 8.

**4.8. Numerical solution.** We are now in a position to solve (6a) and (6b) numerically. Equation (21) allows the direct computation of  $\mathbf{f}^n$ , but (22) requires  $\mathbf{u}^{n+1}$ , which we must compute, given  $\mathbf{f}^n$ , from (14) and (15).

Rearranging (14) to separate the terms evaluated at different time steps yields

$$(25) \quad \left( I - \frac{\Delta t}{Re} \sum_d D_d^+ D_d^- \right) \mathbf{u}^{n+1} + \frac{\Delta t}{Re} \mathbf{D}^0 p^{n+1} = \mathbf{R}^n,$$

where

$$(26) \quad \mathbf{R}^n = \left( I - \Delta t \sum_d u_d^n D_{dd}^\pm \right) \mathbf{u}^n + \frac{\Delta t}{3Re} \mathbf{D}^0 s^n + \Delta t \mathbf{F}^n,$$

and  $I$  is the  $2 \times 2$  identity matrix.

We solve (15) and (25) directly for  $\mathbf{u}^{n+1}$  by applying a discrete Fourier transform (DFT) to eliminate  $p^{n+1}$ . For our domain  $\Omega = [0, 1] \times [0, 1]$  discretized 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  $(\cdot)_{k_1,k_2}$  by

$$(27) \quad (\hat{\cdot})_{k_1,k_2} = \sum_{y=0}^{N-1} \sum_{x=0}^{N-1} (\cdot)_{x,y} \exp \left( -\frac{2\pi i}{N} (xk_1 + yk_2) \right).$$

Under this transformation, (15) and (25) become

$$(28) \quad \left( 1 + \frac{4\Delta t}{h^2 Re} \sum_d \sin^2 \left( \frac{\pi k_d}{N} \right) \right) (\hat{u}_d)_{k_1,k_2}^{n+1} + \frac{i\Delta t}{h Re} \sin \left( \frac{2\pi k_d}{N} \right) \hat{p}_{k_1,k_2}^{n+1} = \left( \hat{R}_d \right)_{k_1,k_2}^n,$$

$$(29) \quad \frac{i}{h} \sum_d \sin \left( \frac{2\pi k_d}{N} \right) (\hat{u}_d)_{k_1,k_2}^{n+1} = (\hat{s})_{k_1,k_2}^n,$$

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

We substitute (29) into (28) to solve directly for  $p$ : if we multiply (28) by  $(i/h) \sin(2\pi k_d/N)$ , sum it over the two dimensions, and rearrange for  $\hat{p}$ , we get

$$(30) \quad \hat{p}_{k_1,k_2}^{n+1} = \frac{\left( 1 + \frac{4\Delta t}{h^2 Re} \sum_d \sin^2 \left( \frac{\pi k_d}{N} \right) \right) (\hat{s})_{k_1,k_2}^n - \frac{i}{h} \sum_d \sin \left( \frac{2\pi k_d}{N} \right) \left( \hat{R}_d \right)_{k_1,k_2}^n}{\frac{\Delta t}{h^2 Re} \sum_d \sin^2 \left( \frac{2\pi k_d}{N} \right)},$$

where every term on the right-hand side depends only on information available at the current time step. We can therefore substitute (30) back into (28) to solve for  $\hat{\mathbf{u}}_{k_1,k_2}^{n+1}$ , obtaining

$$(31) \quad (\hat{u}_d)_{k_1,k_2}^{n+1} = \frac{\left( \hat{R}_d \right)_{k_1,k_2}^n - \frac{i\Delta t}{h Re} \sin \left( \frac{2\pi k_d}{N} \right) \hat{p}_{k_1,k_2}^{n+1}}{1 + \frac{4\Delta t}{h^2 Re} \sum_d \sin^2 \left( \frac{\pi k_d}{N} \right)}.$$

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 (30) vanishes. At these points, however, the sine term multiplying  $\hat{p}_{k_1,k_2}^{n+1}$  in (28) also vanishes, and we may thus solve directly for  $(\hat{u}_d)_{k_1,k_2}^{n+1}$ . We, therefore, avoid this problem by setting  $\hat{p}_{k_1,k_2}^{n+1} = 0$  in (31). Finally, having computed  $(\hat{u}_d)_{k_1,k_2}^{n+1}$ , we apply the inverse DFT to obtain  $\mathbf{u}^{n+1}$ ,

$$(32) \quad (u_d)_{x,y}^{n+1} = \frac{1}{N^2} \sum_{k_2=0}^{N-1} \sum_{k_1=0}^{N-1} (\hat{u}_d)_{k_1,k_2}^{n+1} \exp \left( \frac{2\pi i}{N} (xk_1 + yk_2) \right).$$

**5. Computational implementation.** In this section, we describe the time-stepping algorithm for solving the IB model 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 implemented within this IB method implementation.

**5.1. Chaste.** We have implemented our IB framework as part of the Chaste C++ library [21, 29]. The IB method code is released as a feature branch of the latest development version of Chaste,<sup>2</sup> which is open source and available under the 3-clause BSD license. Chaste is developed with a test-driven approach using the unit testing framework CxxTest.<sup>3</sup> 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 our IB method implementation, as well as code to recreate all simulations from this paper, can be found in Appendix A.

As it is written in C++, Chaste is fast and able to utilize object orientation and class inheritance, enabling modularity and easy extensibility of the code base. This structure enables the IB method 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 specialized 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 4, we solve the IB model 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 (13);
3. loop over each immersed boundary node and propagate the associated force to the fluid mesh domain, as described by (21);
4. loop over each fluid source and propagate the associated strength to the fluid mesh domain, as described by (23);
5. solve the Navier–Stokes equations (6a) and (6b) using the fast Fourier transform algorithm detailed in subsection 4.8 to generate new fluid velocities;
6. use the new fluid velocities to update immersed boundary node and fluid source locations as described by (22) and (24).

An example of a simple simulation performed using this implementation within Chaste is visualized in Figure 2, where an elliptical immersed boundary relaxes over time towards a circular shape. The fluid flow is shown as a vector field of arrows.

**5.2. Implementation of cellular processes.** Sections 2 to 4 detail our IB method 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 immersed boundaries or fluid sources are altered, but which do not directly alter the underlying fluid problem.

An example of rule-based modification of immersed boundaries is cell division.

<sup>2</sup><http://www.cs.ox.ac.uk/chaste/download.html>

<sup>3</sup><http://cxxtest.com/>

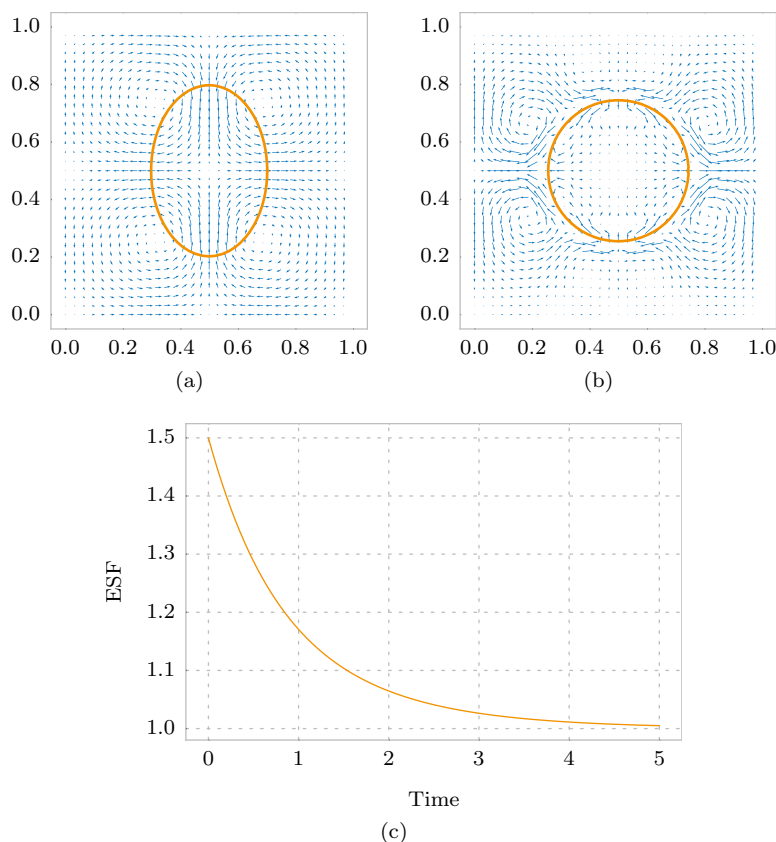


FIG. 2. An example IB method simulation. *An elliptical immersed boundary relaxing over time under the action of internal forces (subsection 4.3) and no fluid sources. In this simulation,  $h = 1/32$ ,  $\Delta t = 0.05$ , and  $N = 128$  nodes. Full details of all parameters can be found in the simulation code, available as part of the test suite “TestNumericsPaperSimulations” (see Appendix A). (a) State of the simulation after one time step, where flow is acting to reduce the elliptical immersed boundary 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 6 for details), over time.*

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 immersed boundary (representing the cell about to divide) with two immersed boundaries (representing the daughter cells). First, a division axis through the centroid of the immersed boundary is selected, by means of some rule chosen by the user. This rule may, for instance, select the shortest axis of the immersed boundary, 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 predetermined fixed distance. We make the choice that each daughter cell is represented by the same number of nodes as the parent cell, and a remeshing process instantaneously spaces these nodes evenly around the outline of each daughter cell.

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 [33] and Rejniak, Kliman, and Fauci [35]. 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 immersed boundary 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 remeshing of the nodes along the boundary so as to keep the elastic properties constant. Currently within Chaste, we have elected not to remesh, which 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 subsection 6.1. Second, as boundaries move around and change size, changes in connectivity are necessitated between neighboring cells. Within Chaste, all connections between neighboring cells are recalculated every time step based solely on all pairwise nodes within the threshold distance  $d_{ext}$ . To prevent this recalculation from being prohibitively costly, an efficient spatial decomposition algorithm is employed, as described in subsection 5.3. Finally, with the active motion of, and interaction between, immersed boundaries, the node spacings within a single boundary inevitably fluctuate. To cope with this, Chaste implements a static remeshing algorithm to redistribute the existing nodes around a given boundary in a volume-preserving manner when required.

**5.3. Computational efficiency and profiling.** The two most computationally demanding steps in our IB method implementation are solving the Navier–Stokes equations and calculating the forces acting on the immersed boundary 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. The latter is costly due to the potentially large number of pairwise interactions between nodes on neighboring immersed boundaries that must be kept track of.

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 setup 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 speedup 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 optimize the second problem of efficiently calculating pairwise interactions between nearby immersed boundary nodes, we employ a spatial decomposition algo-

TABLE 1

Code profiling. The memory footprint, the time to complete 2000 time steps, and the proportion of time spent solving the Navier–Stokes problem are presented for each of three increasingly fine simulation representations. Each simulation comprises a regular hexagonal lattice of 20 immersed boundaries, 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.

	$h = 1/512$	$h = 1/1024$	$h = 1/2048$
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

rithm [15]. The domain is broken into squares each the size of 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 is then between nodes in the same or neighboring squares. Thus, we dramatically reduce the computation necessary when  $d_{ext} \ll 1$ .

Table 1 shows various profiling statistics for a prototype simulation of 20 cells initially arranged in a hexagonal packing. The columns of Table 1 each represent a successive doubling of the resolution of both the fluid mesh and the immersed boundary nodes. As can be seen, solution of the fluid problem scales worse than calculation of the forces; however, neither component individually dominates the simulation runtime.

**6. Numerical results.** In this section, we run a number of simulations to demonstrate various properties of our IB method implementation. We first highlight an important relationship between the immersed boundary node spacing,  $\Delta\gamma_k$ , and the fluid mesh spacing,  $h$ . We go on to explore how certain parameters in the IB method 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 immersed boundary 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 [7]. The ESF for a circle is 1, and for an ellipse it is the ratio of major to minor axis length.

**6.1. Node spacing ratio and volume change.** In the continuous IB method, immersed boundaries are carried at precisely the local fluid velocity (see (4)) because they are impermeable to fluid, in the sense that any given fluid particle will remain either inside or outside a particular immersed boundary for all time. In the discretized IB method, 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 thus fluid will be able to flow across the boundary. In addition to this flow across the boundary under poorly chosen mesh spacings, lack of volume conservation is a known problem with IB numerical methods themselves [26]. While various improvements to the numerical method can be made [12, 28], these are often technically complex and

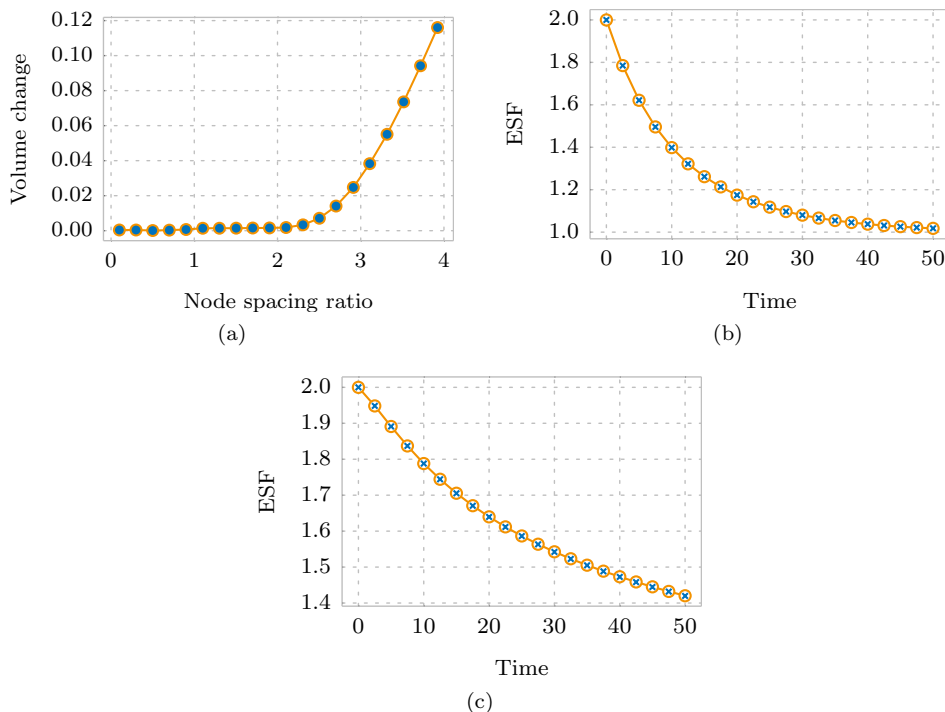


FIG. 3. (a) Node spacing ratio and volume change. A set of simulations of a single circular immersed boundary, each run for the same fixed simulation time. Across the set of simulations the node spacing ratio,  $\Delta\gamma_k/h$ , is varied and the proportional volume change of the immersed boundary 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 intra-cellular 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 immersed boundary 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 neighboring ellipses relaxing, are run in which the ESF of one ellipse is sampled at 21 time points. Circles represent a simulation in which the immersed boundary 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.

bring with them a computational cost, and it is worth noting that these errors go to zero with the mesh spacing  $h$  in any case [26].

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

To determine how small is small enough, Figure 3a shows the results of a set of simulations relating the change in volume of a circular immersed boundary 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  $\Delta\gamma_k < l_{int}$  to ensure the linear springs are under tension and will, in the absence of the volume conservation property of the IB method, contract to reduce the perimeter of the

immersed boundary. 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 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.

**6.2. Scaling of individual cell properties.** A single cell represented by an immersed boundary 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 subsection 6.1, leave the simulation output unchanged. Changing the immersed boundary 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 immersed boundary is decreased by a factor of  $\alpha$ , 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 discretized force relation equation (21), is reduced by a factor  $\alpha$ , and therefore  $\mathbf{F}$  must be increased by this factor in order to compensate. Second, since the boundary is represented by linear springs, we are now considering a system with  $\alpha$  times the number of springs, each with length reduced by a factor  $\alpha$ . 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

$$(33) \quad E_{ref} = \frac{1}{2} \kappa_{int}^{ref} (\Delta\gamma_k - l_{int})^2,$$

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

$$(34) \quad E_{new} = \frac{1}{2} \kappa_{int}^{new} \left( \frac{\Delta\gamma_k}{\alpha} - \frac{l_{int}}{\alpha} \right)^2.$$

To ensure the potential in the immersed boundary is identical in both the reference and the new configurations, we must equate  $E_{ref}$  with  $\alpha E_{new}$ , giving  $\kappa_{int}^{new} = \alpha \kappa_{int}^{ref}$ . Combining the scaling by  $\alpha$  from both considerations, we thus find that to increase the number of nodes in an immersed boundary by a factor  $\alpha$ , we require an  $\alpha^2$  increase in  $\kappa_{int}$ . Figure 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 immersed boundary, 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  $\alpha$  in each immersed boundary relative to a reference scenario will also increase the number of connections, determined via (12), by a factor  $\alpha$ . As the immersed boundaries 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 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  $\alpha$ , we must scale  $\kappa_{int}$  by  $\alpha^2$  and  $\kappa_{ext}$  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.

**6.3. Convergence analysis.** Here, we demonstrate how the numerical implementation converges with time step, fluid mesh spacing, and immersed boundary node spacing. We conduct this convergence analysis using a simple prototype simulation of an elliptical immersed boundary undergoing relaxation for a fixed simulation time. For each of the three parameters of interest,  $\Delta t$ ,  $h$ , 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 analyze convergence with time step, we run the relaxation simulation 19 times, starting with  $\Delta t = 0.5$  and each time reducing  $\Delta t$  by a factor of  $\sqrt{2}$ . Figure 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 15 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 immersed boundary nodes to eliminate the node spacing ratio issue discussed in subsection 6.2 and so as not to vary  $\Delta\gamma_k$ . Figure 4c shows convergence of the ESF with  $h$ . Defining the error in a manner similar 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 immersed boundary node spacing, we run 16 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 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 immersed boundary node spacing: for a given cell division, the shape of the resulting daughter cells should be independent of the choice of boundary parametrization. We verify this convergence by performing cell division operations on a number of elliptical immersed boundaries, each represented by a different number of nodes and using the ESF as a summary statistic of daughter cell shape. Figure 5 shows results with a log-log gradient of 1.96, demonstrating the order of convergence to be quadratic.

**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 vertex models, such as those of the *Drosophila* wing imaginal disc presented by Farhadifar et al. [8]. The aims of this section are threefold. First, we demonstrate that our implementation is capable of simulating proliferation and the growth of entire tis-

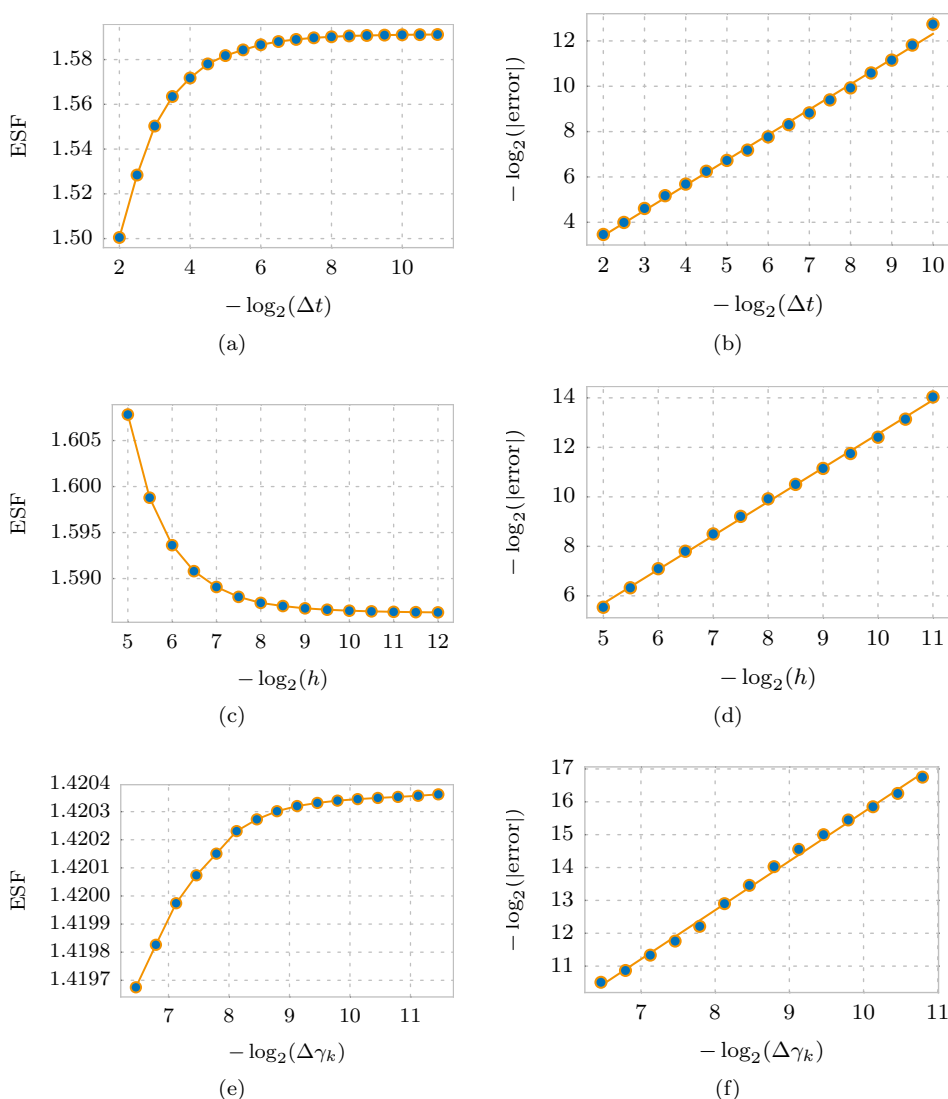


FIG. 4. Convergence of computational implementation. (a) *Convergence with time step.* 19 simulations with different values of  $\Delta t$  were run for a fixed duration of 10 time units, with the following fixed parameters: initial  $ESF = 2.0$ ,  $N = 128$  nodes,  $l_{int} = 50\%$  of node spacing,  $\kappa_{int} = 10^7$ ,  $Re = 10^{-4}$ , with  $128 \times 128$  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.* 15 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 immersed boundary node spacing.* 16 simulations with different numbers of immersed boundary 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  $64 \times 64$  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.

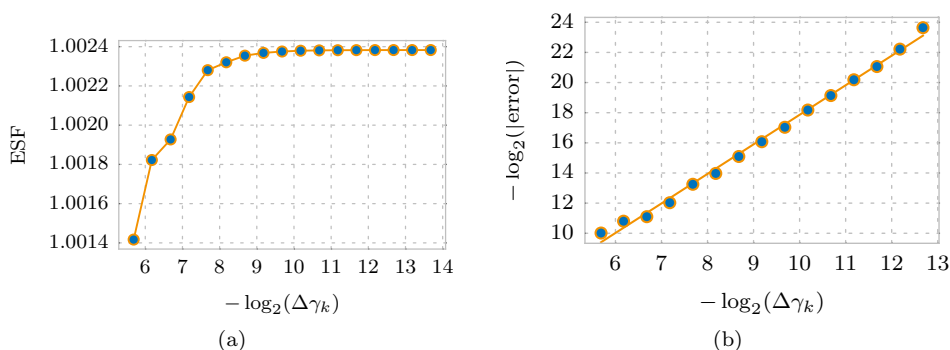


FIG. 5. Convergence of cell division implementation. (a) 17 simulation results showing the ESF of an immersed boundary resulting from the application of the cell division algorithm (subsection 5.2), from elliptical immersed boundaries with varying values of  $\Delta\gamma_k$ . (b) Linear fit between 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.

sues, 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 vertex models.

Vertex models, which treat the apical surface of simple epithelia as a tessellation of polygons, have been successfully used to probe many questions in developmental biology [9]. We aim to simulate an initially small group of cells that undergo repeated rounds of division, using our IB framework. Specifically, we begin each simulation with nine cells represented by hexagonal immersed boundaries 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 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 ( $\kappa_{int}$  and  $\kappa_{ext}$ ). We collect the polygon class distribution (PCD), the distribution of cell neighbor 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, as the property of two cells being neighbors relies on careful selection of a threshold distance. For this reason, the PCD necessarily varies depending on the threshold chosen. A detailed method for matching a PCD between vertex and IB populations lies outside the scope of this paper.

Figure 6b shows the variation in PCD for a fixed value of  $\kappa_{int}$ , while  $\kappa_{ext}$  is varied. Using such summary statistics, we can hope to relate parameters between different models to allow like-for-like comparisons.

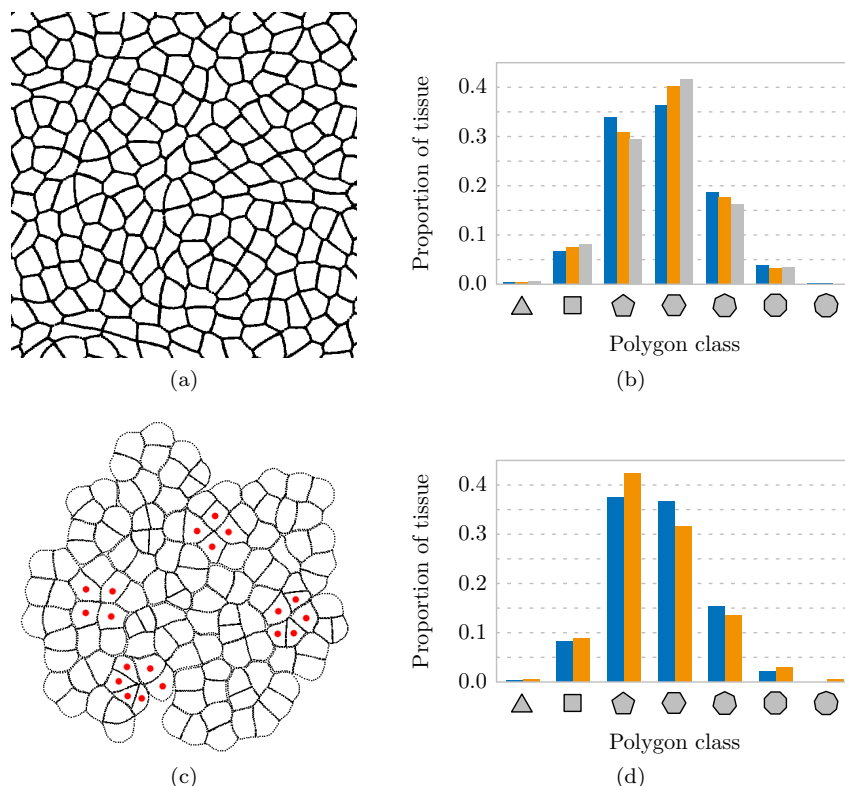


FIG. 6. Simulated epithelial tissues. (a) Snapshot of tissue morphology for one realization with  $\kappa_{int} = \kappa_{ext} = 10^7$ ,  $N = 48$  nodes per boundary, with  $256 \times 256$  fluid mesh points. (b) Polygon class distribution varies with  $\kappa_{ext}$  for fixed  $\kappa_{int} = 10^7$ . (c) Snapshot of a growing tissue simulation, displaying several four-cell and higher-order junctions. (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.

In addition to the PCD, Farhadifar et al. present the area distribution as a summary statistic resulting from different parameter regimes in their vertex models. 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 [8]. This linear distribution is recapitulated for certain parameter regimes in the vertex model 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 vertex model and cannot themselves be altered directly. By contrast, in the IB method, the sizes of cells are precisely determined via fluid sources and sinks. This makes the IB method an excellent candidate for probing the effect of variability in cell size on the tissue morphology.

Figure 6d shows a dependence on polygon distribution, for fixed  $\kappa_{int}$  and  $\kappa_{ext}$ , 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 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 vertex model. 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 [9]. 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 [14], and in the mouse anterior visceral endoderm, where they have been shown to play an important role in development [9]. While rosettes do not form naturally in vertex models, Figure 6c shows several examples in a single snapshot of rosette and similar four-cell structures forming spontaneously. The IB method, 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 IB frameworks appear well suited.

**8. Discussion.** In this paper, we have presented a thorough description of the equations governing the IB method, and full details of a common discretization 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 IB method and have demonstrated the convergence properties of our implementation. Finally, we have demonstrated, through a prototype simulation study, the potential utility of the IB method to investigate questions in epithelial morphogenesis. In this section, we return to several choices made during the formulation of our IB framework.

**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 IB method 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 [34]. A concrete example, though, of the difference between the IB method 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 nonzero body force in the IB method fluid and therefore an induced flow. Since the fluid cannot be assumed to faithfully represent underlying biology, it is not obvious that modelling a biological situation with a small Reynolds number necessarily means the Reynolds number in an associated immersed boundary problem need also be small. Rejniak et al., for instance, derive a “biological” Reynolds number of  $10^{-9}$ , but use the value  $5.9 \times 10^{-5}$  for their simulations [35], a value chosen so as to recapitulate the relevant dynamics. This discrepancy demonstrates that the IB method 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 IB method simulations need necessarily be very small. Further investigation is required to ascertain the relationship between “fluid” properties in vivo and in silico for the IB method.

Cutting experiments, for instance, where tissue is observed to recoil after ablation, could be used to fit an appropriate Reynolds number for the IB method in order to match in vivo dynamics. While IB method implementations based on Stokes flow do exist [2, 20, 41], 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.

**8.2. Discrete delta function.** We made a specific choice for the form of the discrete delta function. Peskin [26] derived the following form for  $\phi$ , in contrast to that presented in (10):

$$(35) \quad \phi(r) = \begin{cases} \frac{1}{8} \left( 3 - 2|r| + \sqrt{1 + 4|r| - 4r^2} \right), & |r| \leq 1, \\ \frac{1}{2} - \phi(2 - |r|), & 1 \leq |r| \leq 2, \\ 0, & 2 \leq |r|. \end{cases}$$

While the functional form appears quite different, the numerical values taken by the different formulations of  $\phi$  are very similar (differing by less than 0.008 at any point in the domain). Given this incredible 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 likely due to difficult branch prediction of the “if” statement necessary to compute  $\phi$  using (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  $\phi$  is immaterial.

**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 (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 behavior more closely, and one such example is the Morse potential [23], which has a functional form  $V(r) = \kappa(1 - e^{-a(r-l)})^2$ , where  $\kappa$  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 immersed boundary 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 impacts immersed boundary simulations is an open question and a topic for further study.

**8.4. Balancing sources.** Finally, in subsection 4.7 we gave no precise formulation for the number of fluid sources,  $M - N$ , in excess of those associated to immersed boundaries. The purpose of these additional sources is to balance the net fluid creation due to processes such as cell growth, in order to ensure a constant fluid volume within the domain  $\Omega$ . In our implementation we choose  $M \approx 2N$  and initially place these equidistant along the boundary  $y = 0$ . Rejniak, Kliman, and Fauci [35] use a similar approach, but do not specify the number of such additional sources, while

Dillon and Othmer [5] 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.

**9. Conclusion.** Through the availability of ever richer datasets from molecular and live-imaging studies, we are in a position to undertake data-driven computational modelling of morphogenetic processes. In tandem, the ready availability of computing power allows not only for individually costly simulations to be run but also for parameter estimation or sensitivity analysis studies requiring thousands of such simulations. The time is ripe, therefore, to take advantage of both the accessibility of high-resolution data and the availability of enormous computational power. We have presented here an open-source, efficient, and modular implementation of the IB method, one such framework able to make use of both.

A strength of such models is the ease with which cellular heterogeneity (for example, through patterning mechanisms) may be incorporated and the consequences for tissue-scale behavior may 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 analyze the tissue-level mechanical properties of such models remains an avenue for future work in this area.

**Appendix A. Obtaining the source code.** The C++ implementation of the IB method within Chaste is available as a feature branch, named “fcooper/immersed\_boundary,” as part of the publicly accessible Chaste Git repository at <https://chaste.cs.ox.ac.uk/git/chaste.git>. Further details on accessing this repository can be found in the supplementary material.

The code used for simulations in this paper is a Chaste “user project,” provided as a separate Git repository. Details on accessing this repository can be found in the supplementary material, which is linked from the main article webpage.

Within this user project, numerical convergence simulations and epithelial growth simulations can be found in /apps/src, where “.cpp” files define the simulations and “.py” files run the simulations and perform the postprocessing. Profiling simulations are defined in the test suite /test/TestProfiling.hpp. All other simulations are defined as individual tests, which are defined and documented in the test suite /test/TestNumericsPaperSimulations.hpp.

**Acknowledgments.** The Chaste developers have been of great and varied help. In particular, we wish to thank (in alphabetical order) Jonathan Cooper, Jochen Kursawe, Gary Mirams, Joe Pitt-Francis, and Martin Robinson.

#### REFERENCES

- [1] D. BOTTINO, *Modeling viscoelastic networks and cell deformation in the context of the immersed boundary method*, J. Comput. Phys., 147 (1998), pp. 86–113, <https://doi.org/10.1006/jcph.1998.6074>.
- [2] T. T. BRINGLEY, *Analysis of the Immersed Boundary Method for Stokes Flow*, Ph.D. thesis, New York University, New York, NY, 2008.
- [3] A. T. CHWANG AND T. Y.-T. WU, *Hydromechanics of low-Reynolds-number flow. Part 2. Singularity method for Stokes flows*, J. Fluid Mech., 67 (1975), pp. 787–815, <https://doi.org/10.1017/S0022112075000614>.
- [4] J. O. DADA AND P. MENDES, *Multi-scale modelling and simulation in systems biology*, Integr. Biol. (Camb), 3 (2011), pp. 86–96, <https://doi.org/10.1039/c0ib00075b>.
- [5] R. DILLON AND H. G. OTHMER, *A mathematical model for outgrowth and spatial patterning of the vertebrate limb bud*, J. Theoret. Biol., 197 (1999), pp. 295–330, <https://doi.org/10.1006/jtbi.1998.0876>.

- [6] R. DILLON, M. OWEN, AND K. PAINTER, *A single-cell-based model of multicellular growth using the immersed boundary method*, in *Moving Interface Problems and Applications in Fluid Dynamics*, Contemp. Math. 466, AMS, Providence, RI, 2008, pp. 1–15.
- [7] H. E. EXNER AND H. P. HOUGARDY, *Quantitative Image Analysis of Microstructures: A Practical Guide to Techniques, Instrumentation and Assessment of Materials*, DGM Informationsgesellschaft, Oberursel, Germany, 1988.
- [8] R. FARHADIFAR, J.-C. RÖPER, B. AIGOUY, S. EATON, AND F. JÜLICHER, *The influence of cell mechanics, cell-cell interactions, and proliferation on epithelial packing*, Curr. Biol., 17 (2007), pp. 2095–2104, <https://doi.org/10.1016/j.cub.2007.11.049>.
- [9] A. G. FLETCHER, M. OSTERFIELD, R. E. BAKER, AND S. Y. SHVARTSMAN, *Vertex models of epithelial morphogenesis*, Biophys. J., 106 (2014), pp. 2291–2304, <https://doi.org/10.1016/j.bpj.2013.11.4498>.
- [10] E. GIVELBERG AND J. BUNN, *A comprehensive three-dimensional model of the cochlea*, J. Comput. Phys., 191 (2003), pp. 377–391, [https://doi.org/10.1016/S0021-9991\(03\)00319-X](https://doi.org/10.1016/S0021-9991(03)00319-X).
- [11] B. E. GRIFFITH, *Immersed boundary model of aortic heart valve dynamics with physiological driving and loading conditions*, Int. J. Numer. Methods Biomed. Eng., 28 (2011), pp. 317–345.
- [12] B. E. GRIFFITH, *On the volume conservation of the immersed boundary method*, Commun. Comput. Phys., 12 (2012), pp. 401–432, <https://doi.org/10.4208/cicp.120111.300911s>.
- [13] B. E. GRIFFITH AND X. LUO, *Hybrid finite difference/finite element immersed boundary method*, Internat. J. Numer. Methods Engrg., to appear.
- [14] M. J. HARDING, H. F. MCGRAW, AND A. NECHIPORUK, *The roles and regulation of multicellular rosette structures during morphogenesis*, Development, 141 (2014), pp. 2549–2558.
- [15] D. G. HARVEY, A. G. FLETCHER, J. M. OSBORNE, AND J. PITT-FRANCIS, *A parallel implementation of an off-lattice individual-based model of multicellular populations*, Comput. Phys. Commun., 192 (2015), pp. 130–137, <https://doi.org/10.1016/j.cpc.2015.03.005>.
- [16] S. HOEHME, M. BRULPORT, A. BAUER, E. BEDAWY, W. SCHORMANN, M. HERMES, V. PUPPE, R. GEBHARDT, S. ZELLMER, M. SCHWARZ, E. BOCKAMP, T. TIMMEL, J. G. HENGSTLER, AND D. DRASDO, *Prediction and validation of cell alignment along microvessels as order principle to restore tissue architecture in liver regeneration*, Proc. Natl. Acad. Sci. USA, 107 (2010), pp. 10371–10376, <https://doi.org/10.1073/pnas.0909374107>.
- [17] A. P. HOOVER, B. E. GRIFFITH, AND L. A. MILLER, *Quantifying performance in the medusan mechanospace with an actively swimming three-dimensional jellyfish model*, J. Fluid Mech., 813 (2017), pp. 1112–1155, <https://doi.org/10.1017/jfm.2017.3>.
- [18] S. JADHAV, C. D. EGGLETON, AND K. KONSTANTOPOULOS, *A 3-D computational model predicts that cell deformation affects selection-mediated leukocyte rolling*, Biophys. J., 88 (2005), pp. 96–104, <https://doi.org/10.1529/biophysj.104.051029>.
- [19] D.-V. LE, J. WHITE, J. PERAIRE, K. LIM, AND B. KHOO, *An implicit immersed boundary method for three-dimensional fluid-membrane interactions*, J. Comput. Phys., 228 (2009), pp. 8427–8445, <https://doi.org/10.1016/j.jcp.2009.08.018>.
- [20] R. J. LEVEQUE AND Z. LI, *Immersed interface methods for Stokes flow with elastic boundaries or surface tension*, SIAM J. Sci. Comput., 18 (1997), pp. 709–735, <https://doi.org/10.1137/S1064827595282532>.
- [21] G. R. MIRAMS, C. J. ARTHURS, M. O. BERNABEU, R. BORDAS, J. COOPER, A. CORRIAS, Y. DAVIT, S. J. DUNN, A. G. FLETCHER, D. G. HARVEY, M. E. MARSH, J. M. OSBORNE, P. PATHMANATHAN, J. PITT-FRANCIS, J. SOUTHERN, N. ZEMZEMI, AND D. J. GAVAGHAN, *Chaste: An open source C++ library for computational physiology and biology*, PLoS Comput. Biol., 9 (2013), e1002970, <https://doi.org/10.1371/journal.pcbi.1002970>.
- [22] R. MITTAL AND G. IACCARINO, *Immersed boundary methods*, in *Annual Review of Fluid Mechanics*, Annu. Rev. Fluid Mech. 37, Annual Reviews, Palo Alto, CA, 2005, pp. 239–261, <https://doi.org/10.1146/annurev.fluid.37.061903.175743>.
- [23] P. M. MORSE, *Diatom molecules according to the wave mechanics. II. Vibrational levels*, Phys. Rev., 34 (1929), pp. 57–64, <https://doi.org/10.1103/PhysRev.34.57>.
- [24] H. NGUYEN AND L. FAUCI, *Hydrodynamics of diatom chains and semiflexible fibres*, J. R. Soc. Interface, 11 (2014), 20140314, <https://doi.org/10.1098/rsif.2014.0314>.
- [25] C. S. PESKIN, *Flow patterns around heart valves: A numerical method*, J. Comput. Phys., 10 (1972), pp. 252–271, [https://doi.org/10.1016/0021-9991\(72\)90065-4](https://doi.org/10.1016/0021-9991(72)90065-4).
- [26] C. S. PESKIN, *The immersed boundary method*, Acta Numer., 11 (2002), pp. 479–517, <https://doi.org/10.1017/S0962492902000077>.
- [27] C. S. PESKIN AND D. M. MCQUEEN, *A general method for the computer simulation of biological systems interacting with fluids*, Symp. Soc. Exp. Biol., 49 (1995), pp. 265–276.
- [28] C. S. PESKIN AND B. F. PRINTZ, *Improved volume conservation in the computation of flows*



- with immersed elastic boundaries, *J. Comput. Phys.*, 105 (1993), pp. 33–46, <https://doi.org/10.1006/jcph.1993.1051>.
- [29] J. PITT-FRANCIS, P. PATHMANATHAN, M. O. BERNABEU, R. BORDAS, J. COOPER, A. G. FLETCHER, G. R. MIRAMS, P. MURRAY, J. M. OSBORNE, A. WALTER, S. J. CHAPMAN, A. GARNY, I. M. M. VAN LEEUWEN, P. K. MAINI, B. RODRÍGUEZ, S. L. WATERS, J. P. WHITELEY, H. M. BYRNE, AND D. J. GAVAGHAN, *Chaste: A test-driven approach to software development for biological modelling*, *Comput. Phys. Commun.*, 180 (2009), pp. 2452–2471, <https://doi.org/10.1016/j.cpc.2009.07.019>.
  - [30] O. POLYAKOV, B. HE, M. SWAN, J. W. SHAEVITZ, M. KASCHUBE, AND E. WIESCHAUS, *Passive mechanical forces control cell-shape change during Drosophila ventral furrow formation*, *Biophys. J.*, 107 (2014), pp. 998–1010, <https://doi.org/10.1016/j.bpj.2014.07.013>.
  - [31] E. M. E. PURCELL, *Life at low Reynolds number*, *Am. J. Phys.*, 45 (1977), 3, <https://doi.org/10.1119/1.10903>.
  - [32] K. REJNIAK AND R. DILLON, *A single cell-based model of the ductal tumour microarchitecture*, *Comput. Math. Methods Med.*, 8 (2007), pp. 51–69, <https://doi.org/10.1080/17486700701303143>.
  - [33] K. A. REJNIAK, *A single-cell approach in modeling the dynamics of tumor microregions*, *Math. Biosci. Eng.*, 2 (2005), pp. 643–655, <https://doi.org/10.3934/mbe.2005.2.643>.
  - [34] K. A. REJNIAK, *An immersed boundary framework for modelling the growth of individual cells: An application to the early tumour development*, *J. Theoret. Biol.*, 247 (2007), pp. 186–204, <https://doi.org/10.1016/j.jtbi.2007.02.019>.
  - [35] K. A. REJNIAK, H. J. KLIMAN, AND L. J. FAUCI, *A computational model of the mechanics of growth of the villous trophoblast bilayer*, *Bull. Math. Biol.*, 66 (2004), pp. 199–232, <https://doi.org/10.1016/j.bulm.2003.06.001>.
  - [36] S. A. SANDERSIUS AND T. J. NEWMAN, *Modeling cell rheology with the subcellular element model*, *Phys. Biol.*, 5 (2008), 015002, <https://doi.org/10.1088/1478-3975/5/1/015002>.
  - [37] S. A. SANDERSIUS, C. J. WEIJER, AND T. J. NEWMAN, *Emergent cell and tissue dynamics from subcellular modeling of active biomechanical processes*, *Phys. Biol.*, 8 (2011), 45007, <https://doi.org/10.1088/1478-3975/8/4/045007>.
  - [38] P. SPAHN AND R. REUTER, *A vertex model of Drosophila ventral furrow formation*, *PLoS One*, 8 (2013), e75051, <https://doi.org/10.1371/journal.pone.0075051>.
  - [39] S. TANAKA, *Simulation frameworks for morphogenetic problems*, *Computation*, 3 (2015), pp. 197–221, <https://doi.org/10.3390/computation3020197>.
  - [40] S. TANAKA, D. SICHAU, AND D. IBER, *LBIBCell: A cell-based simulation environment for morphogenetic problems*, *Bioinformatics*, 31 (2015), pp. 2340–2347, <https://doi.org/10.1093/bioinformatics/btv147>.
  - [41] J. M. TERAN AND C. S. PESKIN, *Tether force constraints in Stokes flow by the immersed boundary method on a periodic domain*, *SIAM J. Sci. Comput.*, 31 (2009), pp. 3404–3416, <https://doi.org/10.1137/080720217>.
  - [42] J. B. WALLINGFORD, S. E. FRASER, AND R. M. HARLAND, *Convergent extension: The molecular control of polarized cell movement during embryonic development*, *Dev. Cell*, 2 (2002), pp. 695–706, [https://doi.org/10.1016/S1534-5807\(02\)00197-1](https://doi.org/10.1016/S1534-5807(02)00197-1).