# An adaptive numerical method for multi–cellular simulations of tissue development

James M. Osborne[a]

[a]*School of Mathematics and Statistics, University of Melbourne, Melbourne, 3010, Victoria, Australia*

**Abstract**

In recent years, multi–cellular models, where cells are represented as individual interacting entities, are becoming ever popular. This has led to a proliferation of novel methods and simulation tools. The first aim of this paper is to review the numerical methods utilised by multi–cellular modelling tools and to demonstrate which numerical methods are appropriate for simulations of tissue and organ development and disease. The second aim is to introduce an adaptive time–stepping algorithm and to demonstrate it's efficiency and accuracy. We focus on off–lattice, mechanics based, models where cell movement is defined by a series of first order ordinary differential equations, derived by assuming over–damped motion and balancing forces. We see that many numerical methods have been used, ranging from simple Forward Euler approaches through to higher order single–step methods like Runge–Kutta 4 and multi–step methods like Adams–Bashforth 2. Through a series of exemplar multi–cellular simulations, we see that if: care is taken to have events (births deaths and re–meshing/re–arrangements) occur on common time–steps; and boundaries are imposed on all sub–steps of numerical methods or implemented using forces, then all numerical methods can converge with the correct order. We introduce an adaptive time–stepping method and demonstrate that the best compromise between $L_\infty$ error and run–time is to use Runge–Kutta 4 with an increased time–step and moderate adaptivity. We see that a judicious choice of numerical method can speed the simulation up by a factor of 10–60 from the Forward Euler methods seen in Osborne *et. al.* [2017, `https://doi.org/10.1371/journal.pcbi.1005387`] and a further speed up by a factor of 4 can be achieved by using an adaptive time–step.

*Keywords:* Multi–Cellular Modelling, Cell–Based, Numerical Approximation, Adaptivity

## 1. Introduction

Multi–cellular modelling, where tissues are modelled as a set of discrete interacting cells, is increasingly being used to study the development and malfunction of organs [19]. Multi–cellular modelling has been used to study: Liver development and disease [15]; Kidney

*jmosborne@unimelb.edu.au, ORCID:0000-0002-5622-0104

development [9]; Skin development and wound healing [32, 33, 58]; Colorectal Crypt development and disease [16, 17, 54]; C. elegans Germline development [1]; Biofilm growth [47]; and viral immune system interaction within an infected tissue [45, 26], among others. These studies can be broken into two categories: studies following the growth and development of the tissue [1, 9, 45, 47]; or investigations into how perturbations to subcellular and cellular properties influence systems in dynamic equilibrium (where the properties of the system, i.e., numbers of cells, or average cellular velocity are approximately constant) [15, 16, 17, 32, 33, 54, 58].

Multi–cellular models can be broken up into two categories based on how cells are represented. The first of these is where cells are represented by objects that are restricted to a lattice, known as on–lattice models. The simplest example of on–lattice models are Cellular Automota (CA) models, where cells are represented as individual lattice sites (or collections of cells are an individual lattice site) [39]. Another example is the Cellular Potts Model (CPM) where cells are represented by collections of lattice sites [27]. Usually on–lattice models are simulated by defining a potential, $U$, of the system and using Monte Carlo simulation to evolve the system to minimise $U$. Stochasticity, intrinsic in the method, avoids the system being trapped at a local minimum. While being simple to implement and fast to run the constraint of being fixed to a lattice can introduce anisotropic effects in the simulations [39].

If the restraint of cells lying on a fixed lattice is removed then we get the second family of models known as off–lattice models. The simplest examples of off–lattice models are cell–center models like the Overlapping Spheres (OS) model (also known as a point based model) and Voronoi Tessellation (VT) model. In the former cells are represented as deformable spheres or as points with a decaying interaction force, and cells interact if they are within a specified interaction radius. In the latter cells are connected by a Delaunay Triangulation, with linear forces between cells, and cells are represented by the Voronoi Tessellation (of the cell centres) leading to polygonal cells [39]. Equations of motion are derived by balancing forces on cell centres. Additional off–lattice models can be created by having cells be represented by multiple components. The simplest of these is the Vertex Dynamics (VD) model where cells are represented as connected polygons whose vertices are free to move. Equations of motion are derived by considering a potential $U$ and minimising with respect to moving vertices [21]. Each of these off–lattice models can be further specialised by how it defines free and interfacial boundaries and recent work has looked at the effect of this on simulation outcomes [23]. More detailed off–lattice models consider: arbitrary subcellular elements as point forces, the Subcellular Element Method (SEM) [38, 44]; arbitrary subcellular elements as finite elements, the Finite Element Model (FEM) [4]; or non connected cell boundaries, the Dynamic Cell Model (DCM) [8, 55].

In off–lattice models the motion of nodes (cell centres or components of cells) is represented by considering forces acting on the node, $\{\mathbf{x}_i\}$. These forces can be converted into cell motion by using Newtons second law which results in a second order equation of motion for each cell. Alternatively, these applied forces can be balanced by viscous drag forces,

resisting motion of the nodes, by assuming that the motion is over–damped [39]. The assumption of over–damped motion leads to a system of $N$ first order equations of motion, for the $N$ nodes, of the form:

$$\frac{\mathrm{d}\mathbf{x}_i}{\mathrm{dt}} = \mathbf{F}_i(\mathbf{x}_1, \ldots, \mathbf{x}_N, t), \quad \text{for} \quad i = 1, \ldots, N, \tag{1}$$

Where $\mathbf{F}_i$ is the force exerted on node $i$ (here the drag coefficient has been absorbed into the force for notational simplicity). The simplest approach for the solution of this system of equations, and therefore the simulation of off–lattice multi–cellular models, is to use a Forward Euler (FE) approximation to update the cell locations over time, [39]:

$$\mathbf{x}_i(t + \delta t) = \mathbf{x}_i(t) + \delta t \mathbf{F}_i(\mathbf{x}_1(t), \ldots, \mathbf{x}_N(t), t), \quad \text{for} \quad i = 1, \ldots, N. \tag{2}$$

where $\delta t$ is the chosen time–step. Usually a default time–step of around $\delta t = 1/120 \approx 2^{-7}$ or $\delta t = 1/200 \approx 2^{-8}$ is chosen, [39]. $\delta t$ is chosen so that reducing it further doesn't change the output of the simulation, to avoid oscillating numerical solutions i.e., to ensure that simulations are numerically stable.

Higher order numerical methods have also been used, for example higher order implicit schemes like Runge–Kutta methods [13] or multi–step schemes like Adams Bashforth 2 (AB2)[24]. However, there has been no detailed study into how these methods compare. An alternative simulation method which is used by some multi–cellular simulation tools is to use the energy formulation of the equations of motion and to use stochastic simulation, for example Metropolis sampling, to evolve the system [25, 30]. This has the advantage of speed and simplicity of implementation and can account for biological noise, however simulations are not 100% reproducible and it's sometimes not clear if features of the simulation output are due to the model or due to the nature of the stochastic simulation process.

Over the years many multi–cellular computational tools have been developed. In Table 1 we present a list of popular off–lattice simulation tools, see Fletcher et. al. [19], or Vetter et. al. [55] for more complete lists. We choose not to include the popular off–lattice modelling tools EpiSim [50], Biocellion [28], and HAL [3], in our table, as they are Agent Based Modelling (ABM) frameworks which represent cells as off–lattice objects, however rather than being based on the equations of motion (or their potential form) movement is based on rules and stochastic simulation. We also don't include popular on–lattice modelling frameworks like CC3D [49] and Morpheus [48] (CPM frameworks), or general ABM frameworks like Repast [7] and NetLogo [53].

| Year* | Tool | Ref | Off–Lattice Model Type(s) | Numerical Method(s) | Notes |
|---|---|---|---|---|---|
| Explicit single step schemes | | | | | |
| 2015 | LBIBCell | [51] | DCM | FE | Coupling Lattice Boltzmann flow with Immersed Boundary model for cell boundaries. |
| 2017 | Timothy | [11] | OS | FE | Focus on HPC and parallel implementation. Allows for simulations up to $10^9$ cells, i.e., a tissue $1cm^3$ in size. |
| 2020 | Chaste | [10] | VD OS VT | FE | Large Open Source project, modular framework, multiple models and methods. |
| 2021 | EdgeBased | [5] | OS DCM FEM | FE | Framework that uses edgewise rather than pointwise interactions. Uses virtual moves to calculate movement. |
| 2021 | PalaCell2D | [8] | DCM | FE | Cells represented by boundaries. |
| 2023 | PolyHoop | [55] | DCM | FE | Representing cells as high degree polygons (in 2D). Potential defined by integrals round hoops. |
| 2023 | SimuCell3D | [43] | DCM | FE | 3D Dynamic cell model (cell boundary representation). |
| 2023 | TissueForge | [46] | OS | FE | User friendly tool for OS model (with associated ODEs for biochemistry). Also allows Newtonian dynamics. |
| Higher order single step schemes | | | | | |
| 2017 | MecaGen | [13] | VT | RK4 | Discrete Element Method (DEM) extension of VT model coupled to chemical signalling subcellular models. |
| 2019 | ya‖a | [22] | OS | Heun's | Simple OS model with limited additional cellular variables. Focus on HPC and parallel implementation. |
| 2023 | Epimech | [52] | DCM | RK2/RK4 | 2D cell boundary representations. RK2 and RK4 used in different situations. |
| Implicit multi–step step schemes | | | | | |
| 2018 | PhysiCell | [24] | OS | AB2 | Large open source project $10^5$–$10^6$ cells on quad-core desktop workstations. |
| Force based DE'S simulated by Stochastic Simulation | | | | | |
| 2010 | CellSys | [25] | OS | SS | Stochastic DEs from forces and noise. Main application in Liver development. |
| 2015 | EmbryoMaker | [30] | SEM | SS | Subcellular Element Method leads to Stochastic DEs. |

Table 1: **Off–lattice multi–cellular modelling tools.** (Continued on the following page.)

Table 1: **Off–lattice multi–cellular modelling tools.** (Continued from the previous page.) * denotes the year of the recent cited paper not the initial release. Methods legend: DCM–Dynamic Cell model; OS–Overlapping Speheres or Point based Force model; VD–Vertex Dynamics model; VT–Voronoi Tessellation; FEM–Finite Element Model; SEM–Subcellular Element Method; and ABM–Agent Based Model. Numerical methods legend: FE–Forward Euler (1st order Runge–Kutta) scheme; RK2–2nd order Runge–Kutta scheme; RK4–4th order Runge–Kutta scheme; Heun's–Heun's method (an alternative 2nd order Runge–Kutta scheme); AB2–2nd order Adams–Bashforth scheme; and SS–Stochastic Simulation.

Table 1 demonstrates the variety of multi–cellular models and simulation approaches in the literature. The most common numerical approach is FE as it is simple and well suited to the varying system sizes seen with multi–cellular models. Some frameworks use higher order methods (Heun's method Runge–Kutta 4 (RK4) and AB2) where mechanical interactions are simple and speed is crucial [13, 22, 24]. Interestingly some tools have started to use different numerical methods for different conditions. For example EpiMech [52] uses a Runge–Kutta 2 (RK2) method for simulations without a substrate and a RK4 method for simulations where cells are attached to a substrate.

In other areas of mathematics, adaptivity, where time–step or spatial resolution is varied to ensure a chosen accuracy, have been used. These adaptive schemes can be based on analytic expressions or approximations for the error (for example in fluid flow [29]) or using a heuristic to inform the choice of numerical method or resolution (in cardiac electrophysiology [56, 57]). In either case a judicious choice of numerical method and the use of adaptivity can increase the accuracy of a solution method (for a given computational cost). It can reduce the computational cost by orders of magnitude while maintaining the same accuracy (ensuring stability).

In this paper we present a numerical framework for the simulation of off–lattice multi–cellular models which can be incorporated within existing simulation frameworks which use the over–damped assumption and a Forward Euler approximation [39]. The framework allows the use of single step methods such as Runge–Kutta (RK) schemes, and implicit schemes like Backward Euler (BE). This allows an increased accuracy for the same time–step. Therefore, allowing for larger time–steps (while maintaining the same accuracy) decreasing simulation runtime. In addition the numerical framework includes time–step adaptivity which allows the maintenance of accuracy while decreasing runtimes further.

The remainder of this paper is structured as follows. Firstly, in Section 2, we present the multi–cellular models and numerical methods used in this study, along with a set of exemplar multi–cellular simulation problems. Secondly, in Section 3, we present simulations of the exemplar problems using the numerical methods presented and demonstrate the convergence properties for each exemplar, we also demonstrate the power of using an adaptive time–step. Finally, in Section 4, we discuss balancing accuracy and runtime (including with adaptivity) along with the main convergence results and present some

avenues for future work.

## 2. Methods

In this section we present: the off–lattice multi–cellular models (Section 2.1); the numerical methods (Section 2.2); and the exemplar multi–cellular problems (Section 2.3), which are used in this study.

### *2.1. Off–lattice multi–cellular models*

As discussed in the introduction off–lattice models consider the evolution of cells within a tissue by tracking the motion of "nodes", $\mathbf{x}_1, \ldots, \mathbf{x}_N \in \Re^n$ $(n = 1, 2, 3)$, which represent cell centres, subcellular components, or components of the cell boundary. Here we consider the sub–class of off–lattice models where motion is assumed to be over–damped. So the equations of motion for each node, $\mathbf{x}_i$, reduce to a force balance [39]

$$\eta \frac{\mathrm{d}\mathbf{x}_i}{\mathrm{dt}} = \mathbf{F}_i(\mathbf{x}_j, t). \tag{3}$$

where $\mathbf{F}_i$ is the force applied to the "node" which is balanced by the viscous drag (drag coefficient, $\eta$) experienced. We describe three off–lattice models: overlapping spheres, Voronoi tessellation; and vertex dynamics. Here, we restrict ourselves to the model components that are relevant for our study, and include mainly a high level description. Note that for simplicity all cells in this study have identical properties and we also exclude the growth phase in each model (i.e., all cells are mature). All results presented here also hold for heterogeneous tissues and for growing cells. More detailed model descriptions, including extensions to have cell growth and homogeneous cells can be found in Osborne et. al. [39] or Germano et. al. [23]. All model parameters used in this study are given in Table 2.

**Overlapping Spheres (OS) model.** Sometimes referred to as point force model. Cells are represented by their centres, $\mathbf{x}_i \in \Re^n$ $(n = 1, 2, 3)$, which are points free to move in space. These models were first used by [14], who defined the framework but used energy minimisation to simulate cell movement. later works [40] extended the model and moved to used the simulation methods illustrated here. 2D and 3D examples can be seen in Figure 1 (b) top and bottom row. Cell connectivity is defined by finding all cells within a specific interaction radius, $r_{\max}^{\mathrm{OS}}$. Cell connectivity changes as cells move in or out of this interaction radius. The equation of motion for cell centre $i$, $\mathbf{x}_i$, is given by, [39],

$$\eta \frac{\mathrm{d}\mathbf{x}_i}{\mathrm{dt}} = \sum_{\mathcal{N}_i^{\mathrm{OS}}} \mathbf{F}_{ij}^{\mathrm{OS}}, \tag{4}$$

where $\eta$ is the drag coefficient (which could vary for each cell but here is constant). $\mathcal{N}_i^{\mathrm{OS}}$ is the set of all neighbours of cell $i$, defined as those closer than $r_{\max}^{\mathrm{OS}}$, and $\mathbf{F}_{ij}^{\mathrm{OS}}$ is the force exerted on cell $i$ by cell $j$ which is given by

$$\mathbf{F}_{ij}^{\mathrm{OS}}(t) = \begin{cases} \mu s_{ij}\hat{\mathbf{x}}_{ij}(t)\log\left(1+\frac{||\mathbf{x}_{ij}(t)||-s_{ij}}{s_{ij}}\right), & \text{for } ||\mathbf{x}_{ij}(t)|| < s_{ij}, \\ \mu(||\mathbf{x}_{ij}(t)||-s_{ij})\hat{\mathbf{x}}_{ij}(t)\exp\left(-\rho\frac{||\mathbf{x}_{ij}(t)||-s_{ij}}{s_{ij}}\right), & \text{for } s_{ij} \leq ||\mathbf{x}_{ij}(t)|| \leq r_{\max}, \\ 0, & \text{for } ||\mathbf{x}_{ij}(t)|| > r_{\max}. \end{cases} \tag{5}$$

Here $\mu$ and $\rho$ are parameters that control the magnitude of the force and the decay of the attractive force respectively, which could depend on cell properties but here are constant. $\mathbf{x}_{ij}(t) = \mathbf{x}_i(t) - \mathbf{x}_j(t)$, and $\hat{\mathbf{x}}_{ij}(t)$ is the corresponding unit vector. $s_{ij}$ is the natural separation between cells $i$ and $j$ which is the sum of the radii, and here all cells have a radius of 0.5CD (Cell Diameters) so $s_{ij} = 1$CD. When a cell divides a random direction is chosen and 2 child cells are placed apart by 1CD. More details and a full model description/derivation can be found in [1, 41].

**Voronoi Tessellation (VT) model.** The Voronoi Tessellation model was originally proposed in [31] for studying the colonic crypt but has been extended for use in various other epithelial tissues [39]. Again cells are represented by their centres, $\mathbf{x}_i \in \Re^n$ $(n = 1, 2, 3)$, which are free to move, but here cell connectivity is defined by the Delaunay Triangulation and cell shape is defined by the Voronoi Tessellation [39]. This connectivity needs to be re–calculated regularly as cells move and cell connections change. 1D and 2D example simulations can be seen in Figures 1 (a) and (b) second row. Cell motion is again calculated by balancing the forces on the cell centres but here there is a linear spring between connected cell centres, [39]

$$\eta\frac{\mathrm{d}\mathbf{x}_i}{\mathrm{dt}} = \sum_{\mathcal{N}_i^{\mathrm{VT}}} \kappa\left(\|\mathbf{x}_{ij}(t)\| - s_{ij}\right)\hat{\mathbf{x}}_{ij}(t), \tag{6}$$

Where $\eta$ is again the drag coefficient (which could vary for each cell but here is constant). $\mathcal{N}_i^{\mathrm{VT}}$ is the set of all neighbours of cell $i$ which here is calculated from the Delaunay Triangulation. $\kappa$ is the spring constant and controls the magnitude of the force, which could depend on cell properties but here is constant. $\mathbf{x}_{ij}(t)$ and $\hat{\mathbf{x}}_{ij}(t)$ are as for the OS model. $s_{ij}$ is the target length of the edge between cells $i$ and $j$ which here is 1CD for all cells. When a cell divides a random direction is chosen and 2 child cells are placed apart by 0.1CD and connectivity is recalculated. In order to visualise we use the bounded VT model from [23].

**Vertex Dynamics (VD) model.** Here we restrict ourselves to the 2 dimensional version of the model, as presented in [20, 21], originally developed in [35, 36, 37]. Here cells are represented by polygons whose vertices, $\mathbf{x}_i \in \Re^2$, are free to move in space. Cells are connected if they share a vertex. Cell connectivity is updated by a process known

as T1 swaps which is a local re–meshing operation which allows cells to move past each other and occurs when edges are less than $d_r$ in length, see [20] for details. A similar class of VD models, using the same structure but with different forces, are defined in [18], and the convergence results presented here still hold. An example simulation can be seen in Figure 1 (b) third row. Cell motion is calculated by balancing forces on the cell vertices by minimising an energy functional, containing terms to keep cells approximately circular with a given target area.

$$\eta \frac{\mathrm{d}\mathbf{x}_i}{\mathrm{dt}} = -\nabla_i \sum_{j=1}^{N(t)} \left[ \alpha \left( A_j(t) - A_j^{\mathrm{T}}(t) \right)^2 + \beta \left( C_j(t) - C_j^{\mathrm{T}}(t) \right)^2 + \sum_{m=1}^{M_j} \gamma_{jm} d_{jm}(t) \right]. \quad (7)$$

Where $\eta$ is again the drag coefficient (which could vary for each vertex but here is constant). $A_j(t)$ ($A_j^{\mathrm{T}}(t)$) and $C_j(t)$ ($C_j^{\mathrm{T}}(t)$) are the area (target) and perimeter (target) of cell $j$ at time $t$, respectively. Here we assume that every cell has the same constant target area, $A_j^{\mathrm{T}} = A^{\mathrm{T}}$, and perimeter, $C_j^{\mathrm{T}} = 2\sqrt{A_{\mathrm{T}}/\pi}$. Note, usually the VD model has cell growth over G1 phase of the cell cycle to reach the mature target area but we used a fixed size, of $A_{\mathrm{T}} = 1$, for simplicity, the results presented here still hold for other target areas or growing cells. $\alpha$ and $\beta$ describe cell compressibility and cytoskeletal tension, respectively, $M_j$ denotes the number of vertices contained in cell $j$, $d_{jm}(t)$ is the length of edge $m$ in cell $j$ and $\gamma_{jm}$ describes cell adhesion in edge $m$, which depends on whether the edge is on the boundary, or is internal to the tissue,

$$\gamma_{jm} = \begin{cases} \gamma_{\text{cell,cell}}, & \text{if edge } m \text{ in cell } j \text{ is internal to the tissue, and} \\ \gamma_{\text{cell,boundary}}, & \text{if edge } m \text{ in cell } j \text{ is on the tissue boundary.} \end{cases} \quad (8)$$

Force originally from [35]. When a cell divides it does so by calculating the long axis and adding new vertices to make two child cells [20].

### *2.2. Numerical Methods*

For each model we are solving Equation (4), (6), or (7) which can all be written as the system

$$\frac{\mathrm{d}\mathbf{x}}{\mathrm{dt}} = \mathbf{F}(\mathbf{x}, t). \quad (9)$$

where $\mathbf{x} = (\mathbf{x}_1, \ldots \mathbf{x}_N)$ are the locations of cell centres (OS and VT) or cell vertices (VD), and $\mathbf{F}(\mathbf{x}, t)$ represents the force applied (and includes the drag coefficient, $\eta$), which is model dependent. In order to solve this we generate a numerical approximation to this system by introducing a series of $N_{\mathrm{S}}$ time–steps

$$0 = t^0 < t^1 < \ldots < t^{N_{\mathrm{s}}} = t^{\mathrm{end}}, \quad (10)$$

| Parameter | Description | Model | Value | Units |
|---|---|---|---|---|
| $\eta$ | Drag coefficient | All | 1.0 | CM hours$^{-1}$ |
| $\mu$ | Elastic interaction constant | OS | 50.0 | CM hours$^{-2}$ |
| $\rho$ | Decay in attractive force | OS | 5.0 | - |
| $r_{\max}$ | Force cut-off length | OS | 1.5 | CD |
| $s_{ij}$ | Mature cell separations | OS & VT | 1.0 | CD |
| $\kappa$ | Spring constant | VT | 50.0 | CM hours$^{-2}$ |
| $\alpha$ | Area deformation coefficient | VD | 50.0 | CM hours$^{-2}$ CD$^{-2}$ |
| $\beta$ | Membrane tension coefficient | VD | 1.0 | CM hours$^{-2}$ |
| $A^{\mathrm{T}}$ | Cell target area | VD | 1.0 | CD$^2$ |
| $\gamma_{\mathrm{cell,cell}}$ | Cell-cell adhesion coefficient | VD | 1.0 | CM CD hours$^{-2}$ |
| $\gamma_{\mathrm{cell,boundary}}$ | Cell-boundary adhesion coefficient | VD | 2.0 | CM CD hours$^{-2}$ |
| $d_r$ | Cell rearrangement threshold | VD | 0.01 | CD |

Table 2: **Mechanical parameter values used in all exemplar simulations.** Parameter values sourced from [23, 39]. Note, CD=Cell Diameters CM=Cell Mass.

where $t^i = i \times \delta t$ and $\delta t = t^{\mathrm{end}}/N_{\mathrm{S}}$ is the time–step. We then use a numerical method to calculate the sequence

$$\mathbf{x}(0) = \mathbf{x}^0, \mathbf{x}^1, \ldots, \mathbf{x}^{N_{\mathrm{S}}}. \tag{11}$$

which gives us an approximation to the solution to the problem at the chosen time–steps.

$$\mathbf{x}^i \approx \mathbf{x}(t^i). \tag{12}$$

This sequence can be calculated by many different methods. The most popular is the Forward Euler (FE) method which calculates this sequence using.

$$\mathbf{x}^{i+1} = \mathbf{x}^i + \delta t \mathbf{F}(\mathbf{x}^i, t^i). \tag{13}$$

In this study we focus on single step methods using the setup above. We use both explicit, where we end up with an explicit formula for the next term of the sequence (as for FE), and implicit methods, where we end up with an implicit formula for the next term in the sequence.

*Explicit Methods*

The most common collection of explicit single step numerical methods are known as Runge–Kutta methods [6]. In these methods we discretise the derivative as for Forward Euler, however, we also evaluate $\mathbf{F}(\mathbf{x}, t)$ at a collection of intermediary points instead of just at $\mathbf{x}^i$ and $t^i$. This gives rise to a set of updates.

**Forward Euler (FE) method.** The update for FE is given by Equation (13). This method has a theoretical order of convergence of 1 so as you halve $\delta t$ you should

halve the global truncation error (the error of the numerical approximation to the displacements compared to the exact solution).

**MidPoint (MP) method.** The update for the MP method (also known as Runge–Kutta 2) is, [6],

$$\mathbf{x}^{i+1} = \mathbf{x}^i + \delta t \mathbf{F}\left(\mathbf{x}^i + \frac{\delta t}{2}\mathbf{F}(\mathbf{x}, t^i), t^i + \frac{\delta t}{2}\right). \tag{14}$$

This method has a theoretical order of convergence of 2 so as you halve $\delta t$ you should reduce the global truncation error by a factor of 4.

**Runge–Kutta 3 (RK3) method.** The update for the RK3 method is, [6],

$$\mathbf{x}^{i+1} = \mathbf{x}^i + \frac{\delta t}{6}(\mathbf{h}^1 + 4\mathbf{h}^2 + \mathbf{h}^3), \tag{15}$$

where

$$\mathbf{h}^1 = \mathbf{F}(\mathbf{x}^i, t^i), \quad \mathbf{h}^2 = \mathbf{F}\left(\mathbf{x}^i + \frac{\delta t}{2}\mathbf{h}^1, t^i + \frac{\delta t}{2}\right), \quad \text{and} \tag{16}$$

$$\mathbf{h}^3 = \mathbf{F}\left(\mathbf{x}^i + 2\delta t \mathbf{h}^2 - \delta t \mathbf{h}^1, t^{i+1}\right). \tag{17}$$

This method has a theoretical order of convergence of 3 so as you halve $\delta t$ you should reduce the global truncation error by a factor of 8.

**Runge–Kutta 4 (RK4) method.** The update for the RK4 method is, [6],

$$\mathbf{x}^{i+1} = \mathbf{x}^i + \frac{\delta t}{6}(\mathbf{k}^1 + 2\mathbf{k}^2 + 2\mathbf{k}^3 + \mathbf{k}^4), \tag{18}$$

where

$$\mathbf{k}^1 = \mathbf{F}(\mathbf{x}^i, t^i), \quad \mathbf{k}^2 = \mathbf{F}\left(\mathbf{x}^i + \frac{\delta t}{2}\mathbf{k}^1, t^i + \frac{\delta t}{2}\right), \tag{19}$$

$$\mathbf{k}^3 = \mathbf{F}\left(\mathbf{x}^i + \frac{\delta t}{2}\mathbf{k}^2, t^i + \frac{\delta t}{2}\right), \quad \text{and} \quad \mathbf{k}^4 = \mathbf{F}\left(\mathbf{x}^i + \delta t \mathbf{k}^3, t^{i+1}\right). \tag{20}$$

This method has a theoretical order of convergence of 4 so as you halve $\delta t$ you should reduce the global truncation error by a factor of 16.

*Implicit Methods*

An alternative to these explicit Runge–Kutta schemes is to generate an implicit equation for $\mathbf{x}^{i+1}$. This can lead to increased stability. i.e., the solution may not diverge for larger $\delta t$ as can be the case for explicit methods [6]. However, rather than writing the update $\mathbf{x}^{i+1}$ explicitly we have an implicit equation to solve for $\mathbf{x}^{i+1}$, which can be costly.

**Backward Euler (BE) method.** The update for the BE method is [6]

$$\mathbf{x}^{i+1} = \mathbf{x}^i + \delta t \mathbf{F}(\mathbf{x}^{i+1}, t^{i+1}). \tag{21}$$

This method has a theoretical order of convergence of 1 so as you halve $\delta t$ you should halve the global truncation error.

**Adams Moulton (AM) method.** The update for the second order AM method (also known as the Trapezium Rule) is [6]

$$\mathbf{x}^{i+1} = \mathbf{x}^i + \frac{\delta t}{2}\left(\mathbf{F}(\mathbf{x}^{i+1}, t^{i+1}) + \mathbf{F}(\mathbf{x}^i, t^i)\right). \tag{22}$$

This method has a theoretical order of convergence of 2 so as you halve $\delta t$ you should reduce the global truncation error by a factor of 4.

In order to solve Equations (21) and (22) for $\mathbf{x}^{i+1}$ we need to resort to numerical root finding methods. Here we use the Newton Method based non–linear solvers from PETSc [2] with a relative and absolute tolerance of $\delta_{\text{TOL}}$. Note that higher order AM methods don't fit within the framework presented in this paper as they are multi–step methods i.e., they use solutions from the previous 2 (or more) time–steps to calculate $\mathbf{x}^{i+1}$.

*Adaptivity*

An adaptive time–step is a way to use a smaller time–step when the solution is varying rapidly and a larger one when the solution is varying more slowly. For some systems an approximation to the global truncation error can be calculated from the problem (*a priori* error analysis) or the problem and solution (*a posteriori* error analysis) to allow an appropriate time–step to be calculated [6]. However, due to the changing number and form of the system of equations which comprise, multi–cellular models such analysis is not feasible so instead we use a heuristic error analysis by introducing an Absolute Movement Threshold, $x_{\text{AMT}}$, which defines a maximum distance a cell can move in one time–step, and use this to inform an adaptive time–step $\widehat{\delta t}$.

The way we implement adaptivity is that if a proposed update, $\mathbf{x}^{i+1}$ causes nodes to move more than the threshold, specifically $\max\{\Delta\mathbf{x}^i\} = \max\{|\mathbf{x}^{i+1} - \mathbf{x}^i|\} > x_{\text{AMT}}$, (where the maximum is taken over the set of all nodes). then the time–step is reduced by a power of 2 (i.e., successively halved) in order to ensure $\max\{\Delta\mathbf{x}^i\} \leq x_{\text{AMT}}$. The adaptive time–step $\widehat{\delta t}$ is kept at this level until we reach the end of the global time–step, $\delta t$ where the time–step is reset to $\delta t$. Full details of the adaptive time–step algorithm are given in Algorithm 1.

The numerical parameter values used in this study are given in Table 3. Note that we use a range $\delta t = [2^{-16}, 2^{-6}] \approx [1.5 \times 10^{-5}, 1.6 \times 10^{-2}]$ for all exemplars presented in this study. This allows us to capture the convergence, or not, of all methods without using values of $\delta t$ with infeasibly long run times (greater than 10 hours).

**foreach** *global time–step,* $\delta t$ **do**

Initialise local time–step as $\widehat{\delta t} = \delta t$ and let $t^* = 0$ be the local time since the last global time–step with corresponding global time, $t$.

**while** $t^* < \delta t$ **do**

**if** $t^* + \widehat{\delta t} > \delta t$ **then**

$\widehat{\delta t} = \delta t - t^*$.

**end**

Calculate potential locations $\hat{\mathbf{x}}_{i+1}$ using the time–step $\widehat{\delta t}$.

Let $\Delta \mathbf{x}^i = \hat{\mathbf{x}}^{i+1} - \mathbf{x}^i$.

**while** $\max\{\Delta \mathbf{x}^i\} > x_{\mathrm{AMT}}$ **do**

$\widehat{\delta t} = 2^{-k}\widehat{\delta t}$ where $k = \left\lfloor \log_2 \left( \frac{\max\{\Delta \mathbf{x}^i\}}{x_{\mathrm{AMT}}} \right) \right\rfloor$.

Calculate potential new locations $\hat{\mathbf{x}}^{i+1}$ using the time–step $\widehat{\delta t}$. and update $\Delta \mathbf{x}^i$.

**end**

Let $\mathbf{x}^{i+1} = \hat{\mathbf{x}}^{i+1}$, $t = t + \widehat{\delta t}$, $t^* = t^* + \widehat{\delta t}$, and $i = i + 1$.

**end**

**end**

**Algorithm 1: Simple Adaptivity Algorithm.** Time–step is reduced to prevent movements of nodes larger than the threshold $x_{\mathrm{AMT}}$. The time–step, $\widehat{\delta t}$, is reset to the base value, $\delta t$ every global time–step.

| **Parameter** | **Description** | **Value** | **Units** |
|---|---|---|---|
| $\delta t$ | Time–step | $2^{-6}, 2^{-8}, \ldots, 2^{-18}$ | hours |
| $x_{\mathrm{AMT}}$ | Movement threshold | $1, 0.1, \ldots, 10^{-5}$ | CD |
| $\delta_{\mathrm{TOL}}$ | Non–linear solver tolerance | $10^{-5}, 10^{-10}$ | CD |

Table 3: **Numerical parameter values used in all exemplar simulations.** Note, CD=Cell Diameters.

### *2.3. Exemplar problems*

In order to demonstrate the usage, and compare the efficiency, of the numerical methods presented above, when applied to multi–cellular simulations, we use a set of exemplar problems that increase in complexity. The exemplar problems provide a surrogate for multi–cellular simulations used in the literature (both growing domains and systems in homeostasis) are presented in Figure 1 and detailed below.

**1D Compression.** The simplest multi–cellular simulation is the relaxation of a chain of cells under compression. Motivated by [12, 41], we track the position of 10 cell centers connected by springs (VT model) initially compressed to 10% of their size and allow the system to relax for 1 hour. A simulation of this exemplar can be seen in Figure 1 (a). We can compare the trajectories calculated from numerical approximation against the analytic solution $\mathbf{x}^{\text{Exact}}(t)$ (see Supplementary Section Appendix B for details of how to calculate the analytic solution) by calculating the $L_\infty$ error over time, $L_\infty(t^i) = \max\{|\mathbf{x}^i - \mathbf{x}^{\text{Exact}}(t^i)|\}$. This exemplar allows us to demonstrate the theoretical orders of convergence of the chosen numerical methods in a system with a fixed number of equations (cells).

**1D Proliferation.** The second exemplar we consider is the deformation of a chain of proliferating cells. Motivated by [34], we consider a proliferating 1D chain of cells (VT model). We start with two cells (placed 1CD apart) which divide once they reach a given age (drawn on cell birth from a $U(7, 17)$ distribution (i.e., mean cell cycle duration is 12 hours. We run the simulation for 50 hours. A simulation of this exemplar can be seen in Figure 1 (a). We use the same 1D linear spring model as in the compression example and compare simulations against exact solution using extracted division events (details in Supplementary Section Appendix B) and calculate the $L_\infty$ error over time. This exemplar allows us to demonstrate how the numerical numerical methods perform when the number of equations increases over time due to changing cell number.

**2D Monolayer.** The next exemplar is following the movement of cells within a monolayer of proliferating cells. Motivated by [23, 39], we look at a growing monolayer of cells in 2D using the OS, VT, and VD models. We start with 4 cells placed in equilibrium as shown in Figure 1 (b) and allow cells to proliferate as in the 1D Proliferation exemplar. The simulation is then run for 50 hours and the trajectories tracked, see Figure 1 (b). We compare simulations to the run using RK4 with $\delta t = 2^{-18}$ and calculate the $L_\infty$ error over time, $L_\infty(t^i) = \max\{|\mathbf{x}^i - \mathbf{x}^i_{\text{RK4},\delta t=2^{-18}}|\}$. Note, this value of $\delta t$ is sufficiently small so that using a smaller value doesn't change any of the convergence plots presented. This exemplar allows us to demonstrate the how the numerical numerical methods are affected by cell re–arrangements as the tissue grows.

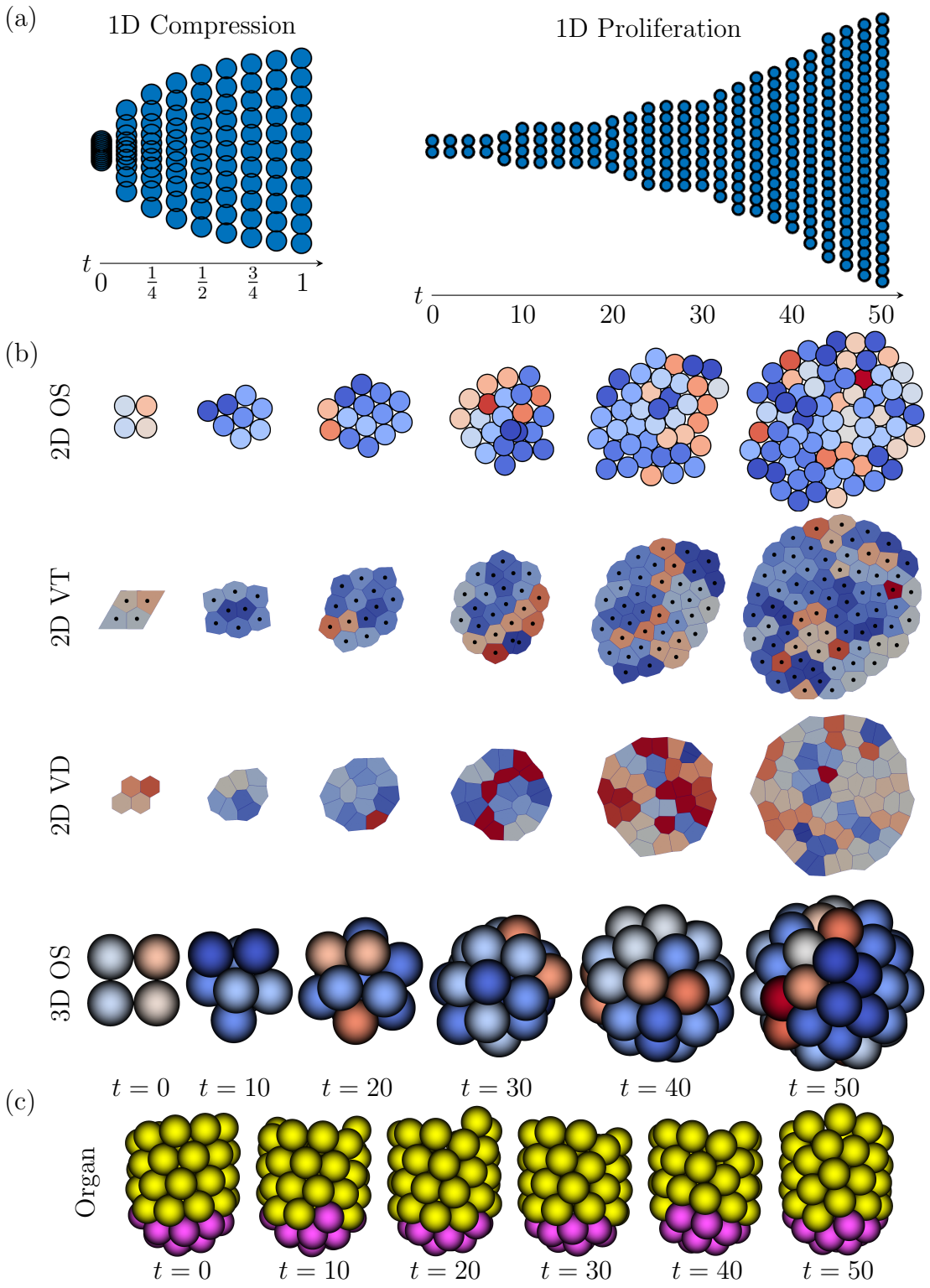



Figure 1: **Exemplar problems.** Time series snapshots from the chosen exemplar problems. (a) 1D Compression and 1D Proliferation exemplars. (b) 2D Monolayer and 3D Spheroid exemplars. (c) 3D Organ exemplar.

**3D Spheroid.** We extend the 2D monolayer exemplar to 3D by looking at the movement of cells within a spheroid of proliferating cells using an OS model. We again start with 4 cells placed in equilibrium as shown in Figure 1 (b), bottom row, and allow cells to proliferate as before. The simulation is then run for 50 hours and the trajectories tracked, allowing the $L_\infty$ error over time to be calculated in the same way as the 2D monolayer exemplar. This exemplar allows us to demonstrate the how the numerical methods behave on 3D tissues.

**3D Organ.** Our final exemplar looks at how cell turnover drives cell movement on a fixed domain. Motivated by simulations of colorectal crypts from [17] we use a 3D OS model and constrain cells to lie on a test tube shaped geometry (base radius 1.5CD and height 4CD) using a restraining force (which leads to cells lying on a fixed surface), see Figure 1 (c). We divide the domain into a proliferative region for $z \leq 2\text{CD}$ (pink cells) where cell division occurs as in the 3D Spheroid exemplar, and a differentiated region $z > 2\text{CD}$ (yellow cells) where cells don't proliferate (position dependent cell proliferation). Cells are removed when they reach the top of the domain, $z = 4\text{CD}$ (position dependent cell removal). To initialise simulations we start with 4 cells at the base of crypt and run for 100 hours (with $\delta t = 2^{-6}$ using RK4) to get to a developed crypt in dynamic equilibrium (see Figure 1 (c)). We then use this as the initial condition and simulate cell movement for a further 50 hours. We compare simulations to the run using RK4 with $\delta t = 2^{-18}$ and calculating the $L_\infty$ error over time, $L_\infty(t^i) = \max\{|\mathbf{x}^i - \mathbf{x}^i_{\text{RK4},\delta t=2^{-18}}|\}$. Note, this value of $\delta t$ is sufficiently small so that using a smaller value doesn't change any of the convergence plots presented. This exemplar allows us to demonstrate the how the numerical numerical methods behave when simulating a 3D homeostatic tissue.

### *2.4. Implementation*

All simulation were undertaken using the Chaste simulation framework. All code used is available, as a bolt on user project, under an Open Source licence, at `https://github.com/jmosborne/MultiCellularNumericalMethods`.

## 3. Results

To compare the performance of the chosen numerical schemes we simulate our exemplar systems for varying $\delta t = 2^{-k}$ (for $k = 6, 8, \ldots, 16$) and compare the results. In order to make numerical convergence possible we make the following assumptions with all simulations presented in this study:

1. all birth or death events occur only at the largest time–step, $\delta t = 2^{-6}$;
2. all re–meshing events: recalculation of Voronoi Tessellation (VT); undertaking T1 swaps (VD); or recalculation of neighbours (OS), only occur at the largest time–step, $\delta t = 2^{-6}$; and

3. all random events (birth times and division angles) are seeded so they are the same for each simulation run.

We now look at each exemplar problem in turn and discuss how the numerical methods presented perform.

### *3.1. Comparison with exact solution shows expected numerical convergence when events are synchronised.*

Figure 2 shows the $L_\infty$ error over time, the convergence of the $L_\infty$ error and the dependence of the $L_\infty$ error on computational runtime, $T_{\mathrm{R}}$, for the 1D Compression ((a) and (b)) and 1D Proliferation ((c) and (d)) exemplars. We see that the error is approximately constant over time for the 1D Compression exemplar, and constant between division events for the 1D Proliferation exemplar. In both cases the $L_\infty$ error converges as per the theoretical convergence rates (e.g. halving $\delta t$ reduces $L_\infty$ error by a factor of 16, demonstrating order 4 convergence for RK4) see Figure 2. Note that in the 1D Proliferation exemplar if cells proliferate on non common time–steps then only first order convergence is witnessed in all methods (results not included for brevity).

Figures 2 (b) and (d) show that the convergence for implicit methods is affected by the convergence of the root finding algorithms. As the $L_\infty$ error of the solution decreases the error of the root finding algorithm cancels out the increases in accuracy. This is more prevalent in the case of a higher solver tolerance ($\delta_{\mathrm{TOL}} = 10^{-5}$, Figure 2 (b)). While the implicit methods converge as expected for low solver tolerance ( $\delta_{\mathrm{TOL}} = 10^{-10}$) they take much more computational time over the explicit methods (See Figure 2 right column and Supplementary Figure S1) therefore we choose to focus on the explicit methods moving forward.

We note that for both the 1D exemplars the solution using RK4 with $\delta t \leq 10^{-16}$ is equal to the exact solution to machine precision (taken to be $< 10^{-12}$), Figures 2 (a) and (c). From now on we will use the RK4 approximation with $\delta t = 10^{-18}$, $\mathbf{x}^i_{\mathrm{RK4},\delta t=2^{-18}}$, as the "exact solution". We also note that the convergence behaviour of the explicit methods is illustrated well by first order convergence of the FE method and the fourth order convergence of the RK4 method, therefore we will focus on these 2 methods moving forward.

### *3.2. Dynamic tissue connectivity can affect convergence and accuracy in 2D and 3D*

In order see how cell rearrangement influences the performance of the numerical methods we simulate the 2D Monolayer (for VT, VD, and OS models) and 3D Spheroid (OS model) exemplars for decreasing $\delta t$ and compare the results to the "exact solution"[1], $\mathbf{x}^i_{\mathrm{RK4},\delta t=2^{-18}}$.

[1]It is not possible to calculate the exact solution in these cases as the models are intractable to analytic methods in more than one dimension.

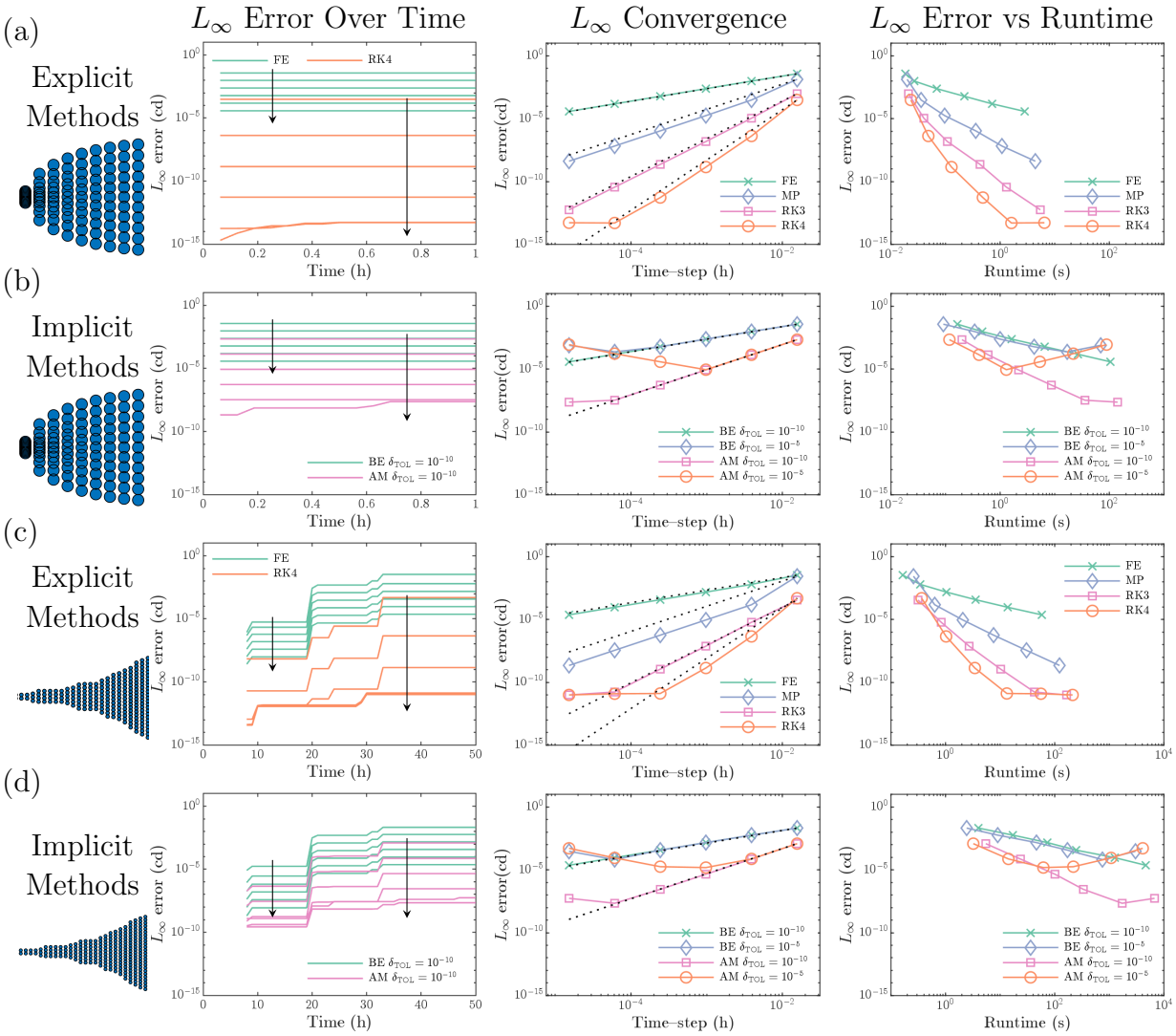



Figure 2: **One dimensional chain of cells.** (a) and (b) one dimensional chain of cells relaxing from compression over an hour (1D Compression exemplar), simulated using explicit (a) and implicit numerical methods (b). (c) and (d) one dimensional chain of cells with proliferation (mean cell cycle duration 12 hours) up to 50 hours (1D Proliferation exemplar), simulated using explicit (b) and implicit numerical methods (d). For explicit methods we present the $L_\infty$ error over time for both the FE method (green) and RK4 method (red). For implicit methods we present the $L_\infty$ error over time for both the BE method (green) and AM method (purple) (for low numerical tolerances $\delta_{\mathrm{TOL}} = 10^{-10}$). We also show how the maximum of this $L_\infty$ error varies with time–step ($\delta t$) and runtime ($T_{\mathrm{R}}$) for all numerical schemes considered here. Arrows show the direction of decreasing $\delta t$. Dotted black lines in the $L_\infty$ convergence plots show the theoretical order of convergence for each numerical method. Note in all cases simulation runtimes scale linearly with $\delta t$ see Supplementary Figure S1.

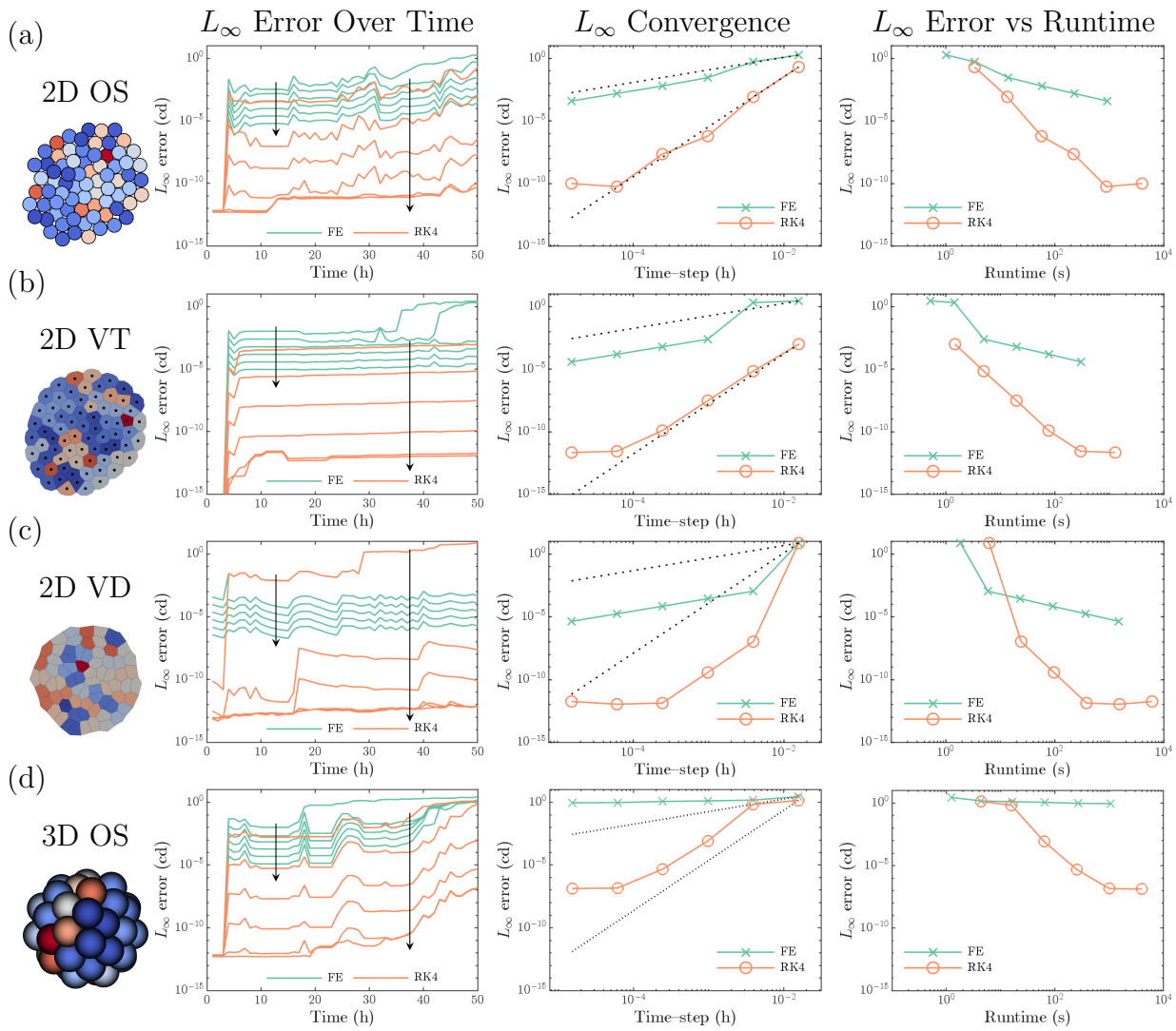



Figure 3: **Growing monolayer and spheroid.** Simulations of tissues with 4 initial cells and stochastic proliferation (mean cell cycle duration 12 hours) up to 50 hours (2D Monolayer and 3D Spheroid exemplars). For each simulation type, (a) OS model in 2D (b) VT model in 2D, (b) VD model in 2D, and (d) OS model in 3D. For each model we present the $L_\infty$ error over time for both the FE method (green) and RK4 method (red), we show how the maximum of this $L_\infty$ error varies with time–step ($\delta t$) and runtime ($T_{\mathrm{R}}$). Arrows show the direction of decreasing $\delta t$. Dotted black lines in the $L_\infty$ convergence plots show the theoretical order of convergence for each numerical method. Note simulation runtimes scale linearly with $\delta t$ see Supplementary Figure S2.

Figure 3 (a)–(c) and (d) shows the $L_\infty$ error over time, the convergence of the $L_\infty$ error and the dependence of the $L_\infty$ error on computational runtime, for the 2D Monolayer and 3D Spheroid exemplars, respectively. We see that for the 2D Monolayer exemplar we maintain convergence rates seen in 1D, Figures 3 (a)–(c), this convergence is seen after a delay (to $\delta t = 2^{-8}$) in the VD model. By looking at the $L_\infty$ error over time we see that the error increases due to the occurrence of an increased number of cells and division and "re–meshing" events. In reality, when using simulations like these are used, multiple runs are made (for different random seeds) and averages presented. Note that, if we allow re–meshing events (or divisions) to occur on any time–step we reduce down to first order convergence for all methods (results not shown for brevity).

We see that the FE method is unable to converge for the 3D Spheroid exemplar, Figure 3 (d). This is because in this exemplar there is more movement and cell rearrangement than the 2D Monolayer exemplar, which increases the error (shown in Figure 3 (d), left). This increase also occurs in the RK4 simulations however, the RK4 method is still able to converge, giving order 4 convergence down to an $L_\infty$ error of $10^{-8}$.

All RK4 simulations (for $\delta t \leq 2^{-8}$) have a lower $L_\infty$ error than FE simulations of a similar runtime, Figure 3 right column. This shows that the same error can be achieved using RK4 with fewer time–steps and a quicker runtime than using FE. Looking at the the most accurate FE simulation (where FE is still performing well, i.e., in 2D simulations) we see that the RK4 simulation with the same $L_\infty$ error are all over an order of magnitude faster than the equivalent FE simulations, Figure 3 (a)–(c) right column. For 3D simulations all RK4 simulations ($\delta t < 2^{-8}$) exhibit a lower $L_\infty$ error than is possible using the FE method for any $\delta t$ used here, Figure 3 (d). Therefore we see that using an RK4 method with appropriately chosen $\delta t$ can achieve the same $L_\infty$ error as a FE simulation over 10 times faster.

### *3.3. Convergence during dynamic equilibrium requires successively smaller time–steps.*

In Figure 4 we present simulations of our 3D Organ exemplar, in (b) we show the $L_\infty$ error over time (compared to the "exact solution", $\mathbf{x}^i_{\text{RK4},\delta t=2^{-18}}$) for the FE and RK4 methods, and in (c) we present the convergence of the $L_\infty$ error, as we reduce the time–step, after set simulation times ($t = 10, 25$, and 50 hours).

In contrast to the 3D spheroid exemplar, we observe from Figure 4 (b) that if the $L_\infty$ error is below $10^{-5}$ at $t = 25$ then it stays below $10^{-5}$ at $t = 50$. Therefore, we consider a target $L_\infty$ error of $10^{-5}$ (to denote stability of numerical method) and this is shown by the black dashed line in Figures 4 (b) and (c). We also see that FE is not able to maintain a low $L_\infty$ error for any time–step used here ($\delta t \geq 2^{-16}$), Figure 4 (b). FE could be able to maintain a low error for some $\delta t \leq 2^{-18}$, however these simulations would take at least $2 \times 10^4$ seconds $\approx 5.5$ hours to run so aren't feasible. Our choice of using forces to constrain cells to a fixed geometry gives more reliable convergence than simpler alternatives. Results

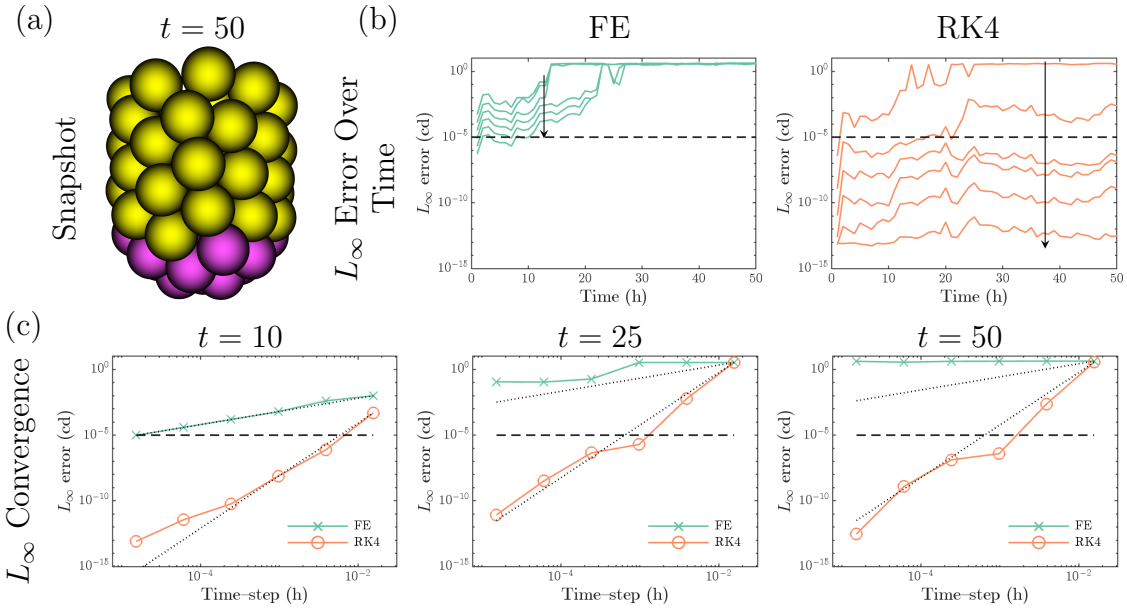



Figure 4: **Convergence for system in dynamic equilibrium**. Simulation of organ system with boundary conditions, and position dependent cell proliferation and removal. (a) example of simulation in dynamic equilibrium (3D Organ exemplar). (b) $L_\infty$ error over time for both the FE method (green) and RK4 method (red). Arrows show the direction of decreasing $\delta t$. (c) convergence of $L_\infty$ error at $t = 10, 25$, and 50 hours for both the FE method (green) and RK4 method (red). Dotted black lines show the theoretical order of convergence for each numerical method. In (b) and (c) the target $L_\infty$ error of $L_\infty = 10^{-5}$ is shown by the black dashed line.

with hard boundaries (as implemented in [17]) lose convergence faster than the force based results presented here (results not shown for brevity).

### *3.4. Adaptive calculation of time–step gives mechanism to balance speed and accuracy.*

From Figure 4 we see a time–step of $\delta t = 2^{-10}$ (for RK4) is necessary to ensure that the $L_\infty$ error remains below the target of $L_\infty = 10^{-5}$ (and that it's not possible for FE). In Figure 5 (a) we compare the convergence of the $L_\infty$ error as we include increased levels of adaptivity, $x_{\mathrm{AMT}} = 1, 0.1, 0.01, \ldots, 10^{-5}$. In Figures 5 (b) and (c) we present the $L_\infty$ error and runtime for RK4 simulations (FE versions are in Supplementary Figure S3). We can see that decreasing $x_{\mathrm{AMT}}$ leads to a subsequent decrease in $L_\infty$ error and this is maintained over time for the RK4 method even for larger global time–steps, $\delta t$. For the FE method adaptivity improves the maintenance of convergence however the only case that maintains a low $L_\infty$ error ($L_\infty \approx 10^{-4}$) for $t = 50$ hours, is for $\delta t = 2^{-16}$ and $x_{\mathrm{AMT}} = 10^{-5}$ which is computationally prohibitive (runtime is over $10^4$ seconds, shown in Supplementary Figure S3 (c)).

Illustrations of time–steps taken are presented in Figure 5 (d). We see that the time–steps are reduced when events (births or deaths shown in red and green) happen. We also see that there is a balance to be made between global time–step, $\delta t$ and adaptivity threshold, $x_{\mathrm{AMT}}$.

By looking at the contour $L_\infty = 10^{-5}$ (dashed black line in Figures 5 (b) and (c)) we can see that using RK4 we can get a the same $L_\infty$ error for a reduced runtime by using larger $\delta t$ and smaller $x_{\mathrm{AMT}}$. To make this clearer we plot a magnified version of the runtime surface in Figure 6 (a). The ability to have an adaptive simulation with lower runtime and same $L_\infty$ error as a non adaptive simulation is because the $L_\infty = 10^{-5}$ contour (dashed line) in Figure 6 (a) intersects the $T_{\mathrm{R}} = 10^{2.25}$ seconds contour close to where there is no adaptivity and intersects the $T_{\mathrm{R}} = 10^{1.5}$ seconds contour where there is a larger time–step with adaptivity, these points are illustrated by the red and blue circles respectively in Figure 6 (a).

In Figures 6 (b) and (c) we present the $L_\infty$ error over time for the numerical parameters above, and some similar choices. In (b) we show the $L_\infty$ error over time for RK4 simulations with: $\delta t = 2^{-8}$; $\delta t = 2^{-10}$; and $\delta t = 2^{-6}, x_{\mathrm{AMT}} = 10^{-2}$. In (c) we show the $L_\infty$ error over time for a FE simulation of a similar runtime ($\delta t = 2^{-10}$ green line) and a FE simulation with similar $L_\infty$ error to the optimal adaptive RK4 simulation ($\delta t = 2^{-16}$ and $x_{\mathrm{AMT}} = 10^{-5}$, pink line). $L_\infty$ error and runtimes for these examples are given in Table 4.

To get any convergence at all with the FE method we require $\delta t = 2^{-16}$ and $x_{\mathrm{AMT}} = 10^{-5}$ which takes 6106 seconds to run. We can achieve an equivalent $L_\infty$ error by running a RK4 simulation with $\delta t = 2^{-9}$ which takes 108 seconds, a speed up of by a factor of 60. Moreover, an $L_\infty$ error of $L_\infty \approx 10^{-5}$ can be achieved using RK4 either without adaptivity

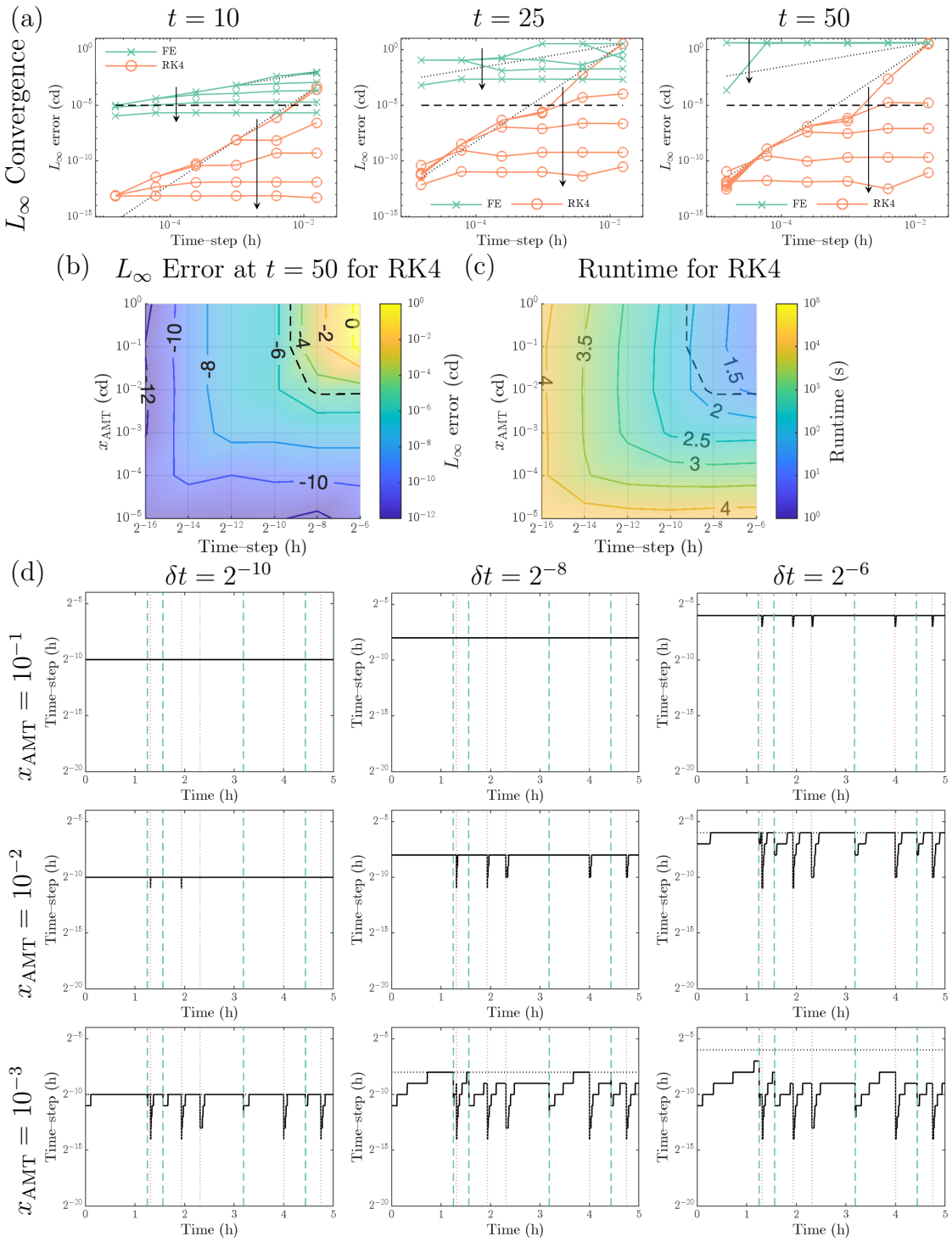



Figure 5: **Adaptivity in simulations at dynamic equilibrium.** Investigating how adaptivity influences the convergence results from Figure 4. (a) how the $L_\infty$ error converges as $\delta t$ is reduced for decreasing $x_{\mathrm{AMT}}$ at $t = 10, 25$ and 50 for both FE and RK4 numerical methods. Arrow in direction of decreasing $x_{\mathrm{AMT}}$ from 1CD to $10^{-5}$CD. Dotted black lines show the theoretical order of convergence for each numerical method. (b) $L_\infty$ error for RK4 simulations as $\delta t$ and $x_{\mathrm{AMT}}$ are varied. (c) Runtime for RK4 simulations as $\delta t$ and $x_{\mathrm{AMT}}$ are varied. In (b) and contour labels show the exponent, $x$, of the $L_\infty$ error or runtime, $10^x$. (Continued on the following page.)

Figure 5: **Adaptivity in simulations at dynamic equilibrium.** (Continued from the previous page.) The dashed contour in (b) and (c) is the target $L_\infty$ error of $L_\infty = 10^{-5}$. Similar plots for simulations using the FE numerical method show a constant $L_\infty$ error (except for the highest level of adaptivity) and are given in Supplementary Figure S3. (d) time–steps taken in simulations for various $\delta t$ and $x_{\text{AMT}}$. The time–step used by the adaptivity algorithm, $\widehat{\delta t}$, is given by the solid black line and the default time–step, $\delta t$, is given by the black dotted line (usually covered by the solid line). Red dotted lines are cell division events and green dot dashed lines show cell removal events (these are the same in all simulations).

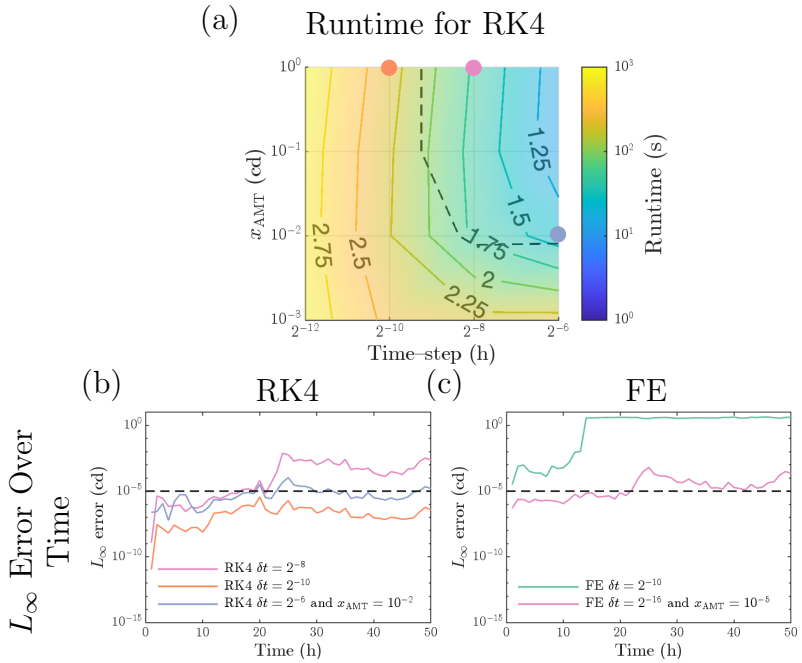



Figure 6: **Adaptivity vs refinement.** (a) Magnified version of simulation runtime from Figure 1 (b) highlighting region near target $L_\infty$ error contour ($L_\infty = 10^{-5}$). (b) $L_\infty$ error over time for numerical methods near target $L_\infty$ error contour. Line colour matches the circled points in (a). The run time for these simulations is as follows: RK4 with $\delta t = 2^{-10}$, 219 seconds; RK4 with $\delta t = 2^{-8}$, 52 seconds; and RK4 with $\delta t = 2^{-6}$ and $x_{\text{AMT}} = 10^{-2}$, 24 seconds. (c) $L_\infty$ error over time for simulations using Forward Euler with a similar run time to the simulations using RK4 in (b) $\delta t = 2^{-10}$, 60 seconds (green) and one with comparable $L_\infty$ error FE with $\delta t = 2^{-16}$ and $x_{\text{AMT}} = 10^{-5}$, 6106 seconds (pink). Line colour matches the circled points in Supplementary Figure S3. The dashed contour in (a) and the dashed line in (b) and (c) denotes the target $L_\infty$ error ($L_\infty = 10^{-5}$).

| Method | $L_\infty$ error (CD) | Runtime (seconds) |
|---|---|---|
| RK4 with $\delta t = 2^{-8}$ | $2.36 \times 10^{-3}$ | 52 |
| RK4 with $\delta t = 2^{-9}$ | $3 \times 10^{-5*}$ | $108^*$ |
| RK4 with $\delta t = 2^{-10}$ | $3.88 \times 10^{-7}$ | 219 |
| RK4 with $\delta t = 2^{-6}$ and $x_{\mathrm{AMT}} = 10^{-2}$ | $1.61 \times 10^{-5}$ | 24 |
| FE with $\delta t = 2^{-10}$ | 4.13 | 60 |
| FE with $\delta t = 2^{-16}$ and $x_{\mathrm{AMT}} = 10^{-5}$ | $2.26 \times 10^{-4}$ | 6106 |

Table 4: **Error vs runtime with adaptivity.** $L_\infty$ error and runtime for a selection of simulations from Figure 6. *Denotes values calculated from interpolation from Figures 5 (b) and (c).

using $\delta t = 2^{-9}$ (taking 108 seconds) or using $\delta t = 2^{-6}$ and adaptivity with $x_{\mathrm{AMT}} = 10^{-2}$ (taking 24 seconds). A speed up by a factor of 4.

## 4. Conclusions

In this paper we have shown that by careful implementation of higher order numerical methods and by restricting events (births, deaths and "re–meshing") to occur on common time steps, theoretical convergence of one step numerical methods, within multi–cellular simulations, can be demonstrated. Specifically the Runge–Kutta 4 method can be shown to have 4th order convergence. While implicit methods, such as Backward Euler, can be used their reliance on non–linear solvers causes any increase in stability to come at a large computational cost limiting their effectiveness.

In a growing system the Runge–Kutta 4 method can achieve the same accuracy over ten times faster than the FE method by being able to use a larger time–step, $\delta t$. For a system in dynamic equilibrium adaptivity can allow the Forward Euler method to maintain convergence. However, the Runge–Kutta 4 method (without adaptivity, with an appropriately chosen $\delta t$) can achieve the same accuracy 60 times faster. Adaptivity and a larger time–step can further reduce runtime of multicellular simulations using the Runge–Kutta 4 method by a factor of 4 leading to an over 200 fold reduction overall. While this work has demonstrated a significant speed up in multi–cellular simulations there are multiple avenues in which we could extend the work.

The current adaptive scheme only allows time–steps in multiples of 1/2, this assumption could be relaxed to have a more dynamic time–step calculation. Additionally, here we have focused on using a uniform time–step for all of the tissue. In the 3D Organ exemplar (much like for many multicellular simulations in the literature) we have a region of proliferative cells and region of differentiated cells. Where cells are proliferating we would need a small time–step, however, in the region where there is no (or even reduced) proliferation we could use a larger time–step. We could also only apply adaptivity (or different movement thresholds) in certain spatial regions to further reduce runtime without compromising on

accuracy or stability. The selected regions would identify a set of cell connections (and therefore ODEs) which would require refined time–steps. Solutions from regions would be combined at common time–step as used in the purely temporal adaptive algorithm presented here.

The techniques presented here could also be extended to work with more detailed multi–cellular models, for example the rod based modelling framework [5] or immersed boundary method [42]. Here there would be issues with including virtual moves into higher order numerical methods and with coupling numerical methods for simulating cell movement with those for fluid flow.

In this paper we have demonstrated that the "default" numerical method of Forward Euler, while simple to implement, is not the best choice for multi–cellular models. While using a Runge–Kutta 4 scheme will take approximately 4 times longer (assuming that force evaluation is the dominant component of the simulation) for the same time–step, the improved accuracy will enable a much larger time–step to be used leading to a greater than tenfold reduction in overall runtime. Moreover, by using a simple adaptivity algorithm a larger global time–step can be used, with time–step refinement where required, the same accuracy can be achieved while reducing runtime, for our example simulation by a factor of 4.

The adaptive numerical framework presented here will enable existing simulations to be undertaken faster reducing the computational burden on researchers. In addition it will allow larger more detailed tissue simulations to be developed and simulated.

## References


[1] K. Atwell, Z. Qin, D. Gavaghan, H. Kugler, E. J. A. Hubbard, and J. M. Osborne. Mechano-logical model of *C. elegans* germ line suggests feedback on the cell cycle. *Development*, 142:3902–3911, 2015. doi: 10.1242/dev.126359.

[2] S. Balay, W. D. Gropp, L. C. McInnes, and B. F. Smith. Efficient management of parallelism in object oriented numerical software libraries. In E. Arge, A. M. Bruaset, and H. P. Langtangen, editors, *Modern Software Tools in Scientific Computing*, pages 163–202. Birkhäuser Press, 1997.

[3] R. R. Bravo, E. Baratchart, J. West, R. O. Schenck, A. K. Miller, J. Gallaher, C. D. Gatenbee, D. Basanta, M. Robertson-Tessi, and A. R. A. Anderson. Hybrid automata library: A flexible platform for hybrid modeling with real-time visualization. *PLoS Computational Biology*, 16(3):1–28, 03 2020. doi: 10.1371/journal.pcbi.1007635.

[4] G. Brodland. A new cell-based FE model for the mechanics of embryonic epithelia. *Computer Methods in Biomechanics and Biomedical Engineering*, 10:121–128, 2007. doi: 10.1080/10255840601124704.

[5] P. J. Brown, J. E. F. Green, B. J. Binder, and J. M. Osborne. A rigid body framework for multicellular modeling. *Nature Computational Science*, 1(11):754–766, 2021. doi: 10.1038/s43588-021-00154-4.

[6] J. C. Butcher. *Numerical methods for ordinary differential equations.* John Wiley & Sons, 2016.

[7] N. Collier. Repast: An extensible framework for agent simulation. *The University of Chicago's Social Science Research*, 36:2003, 2003. doi: 10.1242/dev.126359.

[8] R. Conradin, C. Coreixas, J. Latt, and B. Chopard. Palacell2d: A framework for detailed tissue morphogenesis. *Journal of Computational Science*, 53:101353, 2021. doi: 10.1016/j.jocs.2021.101353.

[9] B. Cook, A. Combes, M. Little, and J. M. Osborne. Modelling cellular interactions and dynamics during kidney morphogenesis. *Bulletin of Mathematical Biology*, 84(1):8, 2021. doi: 10.1007/s11538-021-00968-3.

[10] F. R. Cooper, R. E. Baker, M. O. Bernabeu, R. Bordas, L. Bowler, A. Bueno-Orovio, H. M. Byrne, V. Carapella, L. Cardone-Noott, J. Cooper, S. Dutta, B. D. Evans, A. G. Fletcher, J. A. Grogan, W. Guo, D. G. Harvey, M. Hendrix, D. Kay, J. Kursawe, P. K. Maini, B. McMillan, G. R. Mirams, J. M. Osborne, P. Pathmanathan, J. M. Pitt-Francis, M. Robinson, B. Rodriguez, R. J. Spiteri, and D. J. Gavaghan. Chaste: Cancer, heart and soft tissue environment. *Journal of Open Source Software*, 5(47):1848, 2020. doi: 10.21105/joss.01848.

[11] R. Couturier, M. Cytowski, Z. Szymańska, P. Umiński, G. Andrejczuk, and K. Raszkowski. Implementation of an agent-based parallel tissue modelling framework for the intel mic architecture. *Scientific Programming*, 2017:8721612, 2017. doi: 10.1155/2017/8721612.

[12] Y. Davit, J. M. Osborne, H. M. Byrne, D. J. Gavaghan, and J. M. Pitt-Francis. Validity of the cauchy-born rule applied to discrete cellular-scale models of biological tissues. *Physical Review E*, 87: 042724, 4 2013. doi: 10.1103/PhysRevE.87.042724.

[13] J. Delile, M. Herrmann, N. Peyriéras, and R. Doursat. A cell-based computational model of early embryogenesis coupling mechanical behaviour and gene regulation. *Nature Communications*, 8(1): 13929, 2017. doi: 10.1038/ncomms13929.

[14] D. Drasdo, R. Kree, and J. McCaskill. Monte carlo approach to tissue-cell populations. *Phys Rev E*, 52(6):6635, 1995.

[15] D. Drasdo, S. Hoehme, and J. G. Hengstler. How predictive quantitative modelling of tissue organisation can inform liver disease pathogenesis. *Journal of Hepatology*, 61(4):951–956, 2014. doi: 10.1016/j.jhep.2014.06.013.

[16] S.-J. Dunn, I. S. Näthke, and J. M. Osborne. Computational models reveal a passive mechanism for cell migration in the crypt. *PLoS ONE*, 8(11):e80516, 2013. doi: 10.1371/journal.pone.0080516.

[17] S.-J. Dunn, J. M. Osborne, P. L. Appleton, and I. S. Näthke. Combined changes in Wnt signaling response and contact inhibition induce altered proliferation in radiation-treated intestinal crypts. *Mollecular Biology of the Cell*, 27(11):1863–1874, 2016. doi: 10.1091/mbc.E15-12-0854.

[18] 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. *Current Biology*, 17(24):2095–2104, 2007. ISSN 0960-9822. doi: https://doi.org/10.1016/j.cub.2007.11.049.

[19] A. G. Fletcher and J. M. Osborne. Seven challenges in the multiscale modeling of multicellular tissues. *WIREs Mechanisms of Disease*, 14:e1527, 2022. doi: 10.1002/wsbm.1527.

[20] A. G. Fletcher, J. M. Osborne, P. K. Maini, and D. J. Gavaghan. Implementing vertex dynamics models of cell populations in biology within a consistent computational framework. *Progress in Biophysics and Molecular Biology*, 113:299–326, 2013. doi: 10.1016/j.pbiomolbio.2013.09.003.

[21] A. G. Fletcher, M. Osterfield, R. E. Baker, and S. Y. Shvartsman. Vertex models of epithelial morphogenesis. *Biophysical Journal*, 106:2291–2304, 2014. doi: 10.1016/j.bpj.2013.11.4498.

[22] P. Germann, M. Marin-Riera, and J. Sharpe. ya||a: Gpu-powered spheroid models for mesenchyme and epithelium. *Cell Systems*, 8(3):261–266.e3, 2019. doi: 10.1016/j.cels.2019.02.007.

[23] D. P. J. Germano, A. Zanca, S. T. Johnston, J. A. Flegg, and J. M. Osborne. Free and interfacial boundaries in individual-based models of multicellular biological systems. *Bulletin of Mathematical Biology*, 85(11):111, 2023. doi: 10.1007/s11538-023-01214-8.

[24] A. Ghaffarizadeh, R. Heiland, S. H. Friedman, S. M. Mumenthaler, and P. Macklin. Physicell: An open source physics-based cell simulator for 3-d multicellular systems. *PLoS Computational Biology*, 14(2):1–31, 02 2018. doi: 10.1371/journal.pcbi.1005991.

[25] S. Hoehme and D. Drasdo. A cell-based simulation software for multi-cellular systems. *Bioinformatics*, 26(20):2641–2642, 2010. doi: 10.1093/bioinformatics/btq437.

[26] M. A. Islam, M. Getz, P. Macklin, and A. N. Ford Versypt. An agent-based modeling approach for lung fibrosis in response to covid-19. *PLOS Computational Biology*, 19(12):1–27, 12 2023. doi: 10.1371/journal.pcbi.1011741.

[27] J. A. Izaguirre, R. Chaturvedi, C. Huang, T. Cickovski, J. Coffland, G. Thomas, G. Forgacs, M. Alber, G. Hentschel, S. A. Newman, and J. A. Glazier. CompuCell, a multi-model framework for simulation of morphogenesis. *Bioinformatics*, 20:1129–1137, 2004. doi: 10.1093/bioinformatics/bth050.

[28] S. Kang, S. Kahan, J. McDermott, N. Flann, and I. Shmulevich. Biocellion : accelerating computer simulation of multicellular biological system models . *Bioinformatics*, 30(21):3101–3108, 07 2014. doi: 10.1093/bioinformatics/btu498.

[29] D. A. Kay, P. M. Gresho, D. F. Griffiths, and D. J. Silvester. Adaptive time-stepping for incompressible flow part ii: Navier–stokes equations. *SIAM Journal on Scientific Computing*, 32(1):111–128, 2010. doi: 10.1137/080728032.

[30] M. Marin-Riera, M. Brun-Usan, R. Zimm, T. Välikangas, and I. Salazar-Ciudad. Computational modeling of development by epithelia, mesenchyme and their interactions: a unified model. *Bioinformatics*, 32(2):219–225, 09 2015. doi: 10.1093/bioinformatics/btv527.

[31] F. A. Meineke, C. S. Potten, and M. Loeffler. Cell migration and organization in the intestinal crypt using a lattice-free model. *Cell Proliferation*, 34:(4):253–266, 2001. doi: 10.1046/j.0960-7722.2001.00216.x.

[32] C. Miller, E. Crampin, and J. M. Osborne. Maintaining the proliferative cell niche in multicellular models of epithelia. *Journal of Theoretical Biology*, 527:110807, 2021. doi: 10.1016/j.jtbi.2021.110807.

[33] C. Miller, E. Crampin, and J. M. Osborne. Multiscale modelling of desquamation in the interfollicular epidermis. *PLoS Computational Biology*, 18(8):1–33, 08 2022. doi: 10.1371/journal.pcbi.1010368.

[34] P. J. Murray, C. M. Edwards, M. J. Tindall, and P. Maini. From a discrete to a continuum model of cell dynamics in one dimension. *Physical Review E*, 80:031912–1, 2009. doi: 10.1103/PhysRevE.80.031912.

[35] T. Nagai and H. Honda. A dynamic cell model for the formation of epithelial tissues. *Philos. Mag. B*, 81:699–719, 2001. doi: 10.1080/13642810108205772.

[36] T. Nagai and H. Honda. Wound healing mechanism in epithelial tissues cell adhesion to basal lamina. *WSEAS Transactions on Biology and Biomedicine*, 3(6):389, 2006.

[37] T. Nagai and H. Honda. Computer simulation of wound closure in epithelial tissues: Cell–basal-lamina adhesion. *Phys. Rev. E*, 80:061903, Dec 2009. doi: 10.1103/PhysRevE.80.061903.

[38] T. J. Newman. Modeling multi-cellular systems using sub-cellular elements. *Mathematical Biosciences and Engineering*, 2:611–622, 2005. doi: 10.3934/mbe.2005.2.613.

[39] J. M. Osborne, A. G. Fletcher, J. M. Pitt-Francis, P. K. Maini, and D. J. Gavaghan. Comparing individual-based approaches to modelling the self-organization of multicellular tissues. *PLoS Computational Biology*, 13(2):1–34, 2017. doi: 10.1371/journal.pcbi.1005387.

[40] E. Palsson and H. G. Othmer. A model for individual and collective cell movement in dictyostelium discoideum. *Proc Natl Acad Sci USA*, 97(19):10448–10453, 2000.

[41] P. Pathmanathan, J. Cooper, A. Fletcher, G. Mirams, P. Murray, J. M. Osborne, J. Pitt-Francis, A. Walter, and S. J. Chapman. A computational study of discrete mechanical tissue models. *Physical Biology*, 6:036001, 2009. doi: 10.1088/1478-3975/6/3/036001.

[42] K. A. Rejniak. An immersed boundary framework for modelling the growth of individual cells: an application to the early tumour development. *Journal of Theoretical Biology*, 247:186–204, 2007. doi: 10.1016/j.jtbi.2007.02.019.

[43] S. Runser, R. Vetter, and D. Iber. 3d simulation of tissue mechanics with cell polarization. *bioRxiv*, 2023. doi: 10.1101/2023.03.28.534574.

[44] S. A. Sandersius and T. J. Newman. Modeling cell rheology with the Subcellular Element Model. *Physical Biology*, 5:015002, 2008. doi: 10.1088/1478-3975/5/1/015002.

[45] T. J. Sego, J. O. Aponte-Serrano, J. Ferrari Gianlupi, S. R. Heaps, K. Breithaupt, L. Brusch, J. Crawshaw, J. M. Osborne, E. M. Quardokus, R. K. Plemper, and J. A. Glazier. A modular framework for multiscale, multicellular, spatiotemporal modeling of acute primary viral infection and immune response in epithelial tissues and its application to drug therapy timing and effectiveness. *PLoS Computational Biology*, 16(12):1–51, 12 2020. doi: 10.1371/journal.pcbi.1008451.

[46] T. J. Sego, J. P. Sluka, H. M. Sauro, and J. A. Glazier. Tissue forge: Interactive biological and biophysics simulation environment. *PLOS Computational Biology*, 19(10):1–22, 10 2023. doi: 10.1371/journal.pcbi.1010768.

[47] W. P. J. Smith, Y. Davit, J. M. Osborne, W. Kim, K. R. Foster, and J. M. Pitt-Francis. Cell morphology drives spatial patterning in microbial communities. *Proceedings of the National Academy of Sciences*, 114(3):E280–E286, 2017. doi: 10.1073/pnas.1613007114.

[48] J. Starruß, W. de Back, L. Brusch, and A. Deutsch. Morpheus: a user-friendly modeling environment for multiscale and multicellular systems biology. *Bioinformatics*, 30(9):1331–1332, 2014. doi: 10.1093/bioinformatics/btt772.

[49] M. H. Swat, G. L. Thomas, J. M. Belmonte, A. Shirinifard, D. Hmeljak, and J. A. Glazier. Multiscale modeling of tissues using compucell3d. *Methods in Cell Biology*, 110:325, 2012. doi: 10.1016/B978-0-12-388403-9.00013-8.

[50] T. Sütterlin, C. Kolb, H. Dickhaus, D. Jäger, and N. Grabe. Bridging the scales: semantic integration of quantitative SBML in graphical multi-cellular models and simulations with EPISIM and COPASI. *Bioinformatics*, 29(2):223–229, 11 2012. doi: 10.1093/bioinformatics/bts659.

[51] S. Tanaka, D. Sichau, and D. Iber. LBIBCell: a cell-based simulation environment for morphogenetic problems. *Bioinformatics*, 31(14):2340–2347, 03 2015. doi: 10.1093/bioinformatics/btv147.

[52] A. Tervonen, S. Korpela, S. Nymark, J. Hyttinen, and T. O. Ihalainen. The effect of substrate stiffness on elastic force transmission in the epithelial monolayers over short timescales. *Cellular and Molecular Bioengineering*, 16(5):475–495, 2023. doi: 10.1007/s12195-023-00772-0.

[53] S. Tisue and U. Wilensky. Netlogo: A simple environment for modeling complexity. In *Proceedings of the Fifth International Conference on Complex Systems ICCS 2004*, volume 21, pages 16–21. Citeseer, 2004.

[54] I. van Leeuwen, G. Mirams, A. Walter, A. Fletcher, P. Murray, J. Osborne, S. Varma, S. Young, J. Cooper, B. Doyle, J. Pitt-Francis, L. Momtahan, P. Pathmanathan, J. Whiteley, S. Chapman, D. Gavaghan, O. Jensen, J. King, P. Maini, S. Waters, and H. Byrne. An integrative computational model for intestinal tissue renewal. *Cell Proliferation*, 42:617–636, 2009. doi: 10.1111/j.1365-2184.2009.00627.x.

[55] R. Vetter, S. V. M. Runser, and D. Iber. Polyhoop: Soft particle and tissue dynamics with topological transitions, 2023.

[56] J. P. Whiteley. An efficient numerical technique for the solution of the monodomain and bidomain equations. *IEEE Transactions on Biomedical Engineering*, 53(11):2139–2147, 2006. doi: 10.1109/TBME.2006.879425.

[57] J. P. Whiteley. Physiology driven adaptivity for the numerical solution of the bidomain equations. *Annals of Biomedical Engineering*, 35(9):1510–1520, 2007. doi: 10.1007/s10439-007-9337-3.

[58] A. Zanca, J. A. Flegg, and J. M. Osborne. Push or pull? cell proliferation and migration during wound healing. *Frontiers in Systems Biology*, 2, 2022. doi: 10.3389/fsysb.2022.876075.

## Declaration of competing interest

The author declares that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

## Data availability

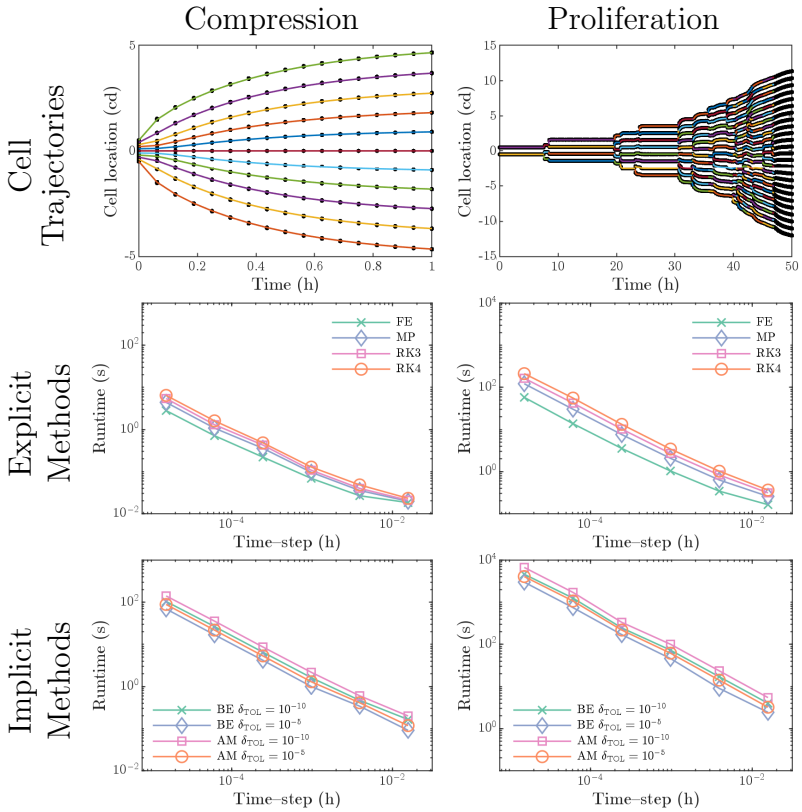
All code to reproduce this study is freely available, under an open source licence, from `https://github.com/jmosborne/MultiCellularNumericalMethods`.

## Acknowledgements

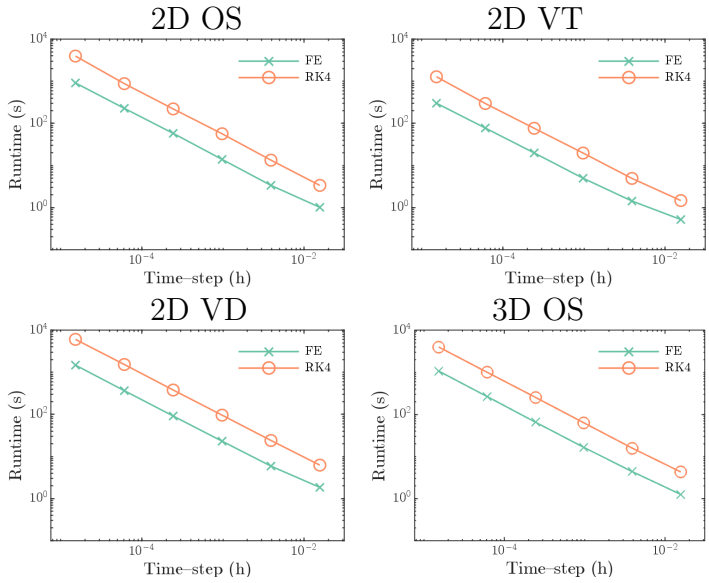
JMO would like to thank Dr Kathryn Tunyasuvunakool for her role in the initiation of this project as she laid the foundations for this work as part of her PhD project. JMO would also like to thank Kathryn's co-supervisors Professor David Gavaghan, Professor Hillel Kugler, Dr Sara-Jane Dunn, and Professor Ruth Baker for their support of this project. JMO's research is supported by the ARC (DP230100380, FT230100352).

## Supplementary Material

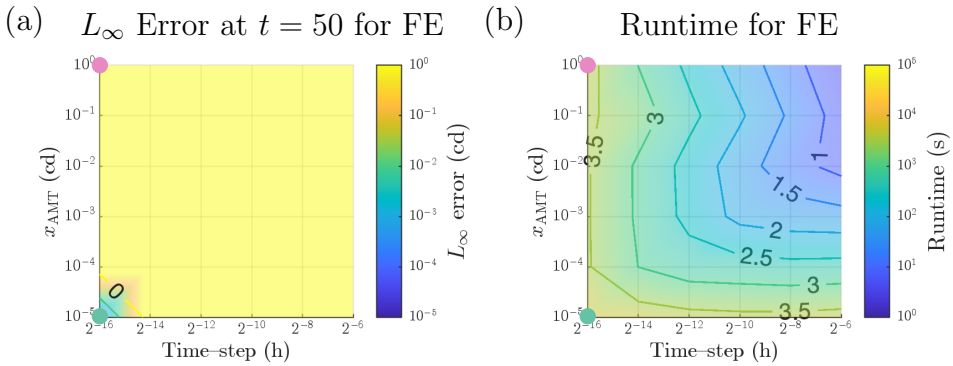
## Appendix A. Supplementary Figures




Supplementary Figure S1: **Cell trajectories and runtime for one dimensional simulations.** Top left, position of cells over time for 1D compression example. Coloured lines represent the exact solution and the numerical solutions at $2^{-4}$ hour intervals are given by the black dots. Top right, position of cells over time, coloured lines represent the exact solution and the numerical solutions at 1 hour intervals are given by the black dots. Middle and bottom, simulation runtime for simulations from Figure 2




Supplementary Figure S2: **Runtime for two and three dimensional simulations.** Runtimes as $\delta t$ is varied for simulations using FE and RK4 methods from Figure 3.




Supplementary Figure S3: **Adaptivity using FE in simulation of dynamic equilibrium.** (a) $L_\infty$ error as $\delta t$ and $x_{\mathrm{AMT}}$ are varied. (b) simulation runtime as $\delta t$ and $x_{\mathrm{AMT}}$ are varied. Contour labels show the exponent, $x$, of the $L_\infty$ error or runtime, $10^x$. Green and pink circles represent the numerical parameters of the simulations in Figure 6 (c).

## Appendix B. Analytic solution of 1D Compression and Proliferation exemplars

For a 1D chain of cells using a linear spring model it is possible to find an exact solution for the motion of the cells. In this section we detail this solution and show how to use it to get exact solutions for comparison for the 1D Compression and 1D Proliferation exemplars. For the 1D Compression exemplar the equations of motion, for cell centres $x_0, \ldots, x_n$ are

$$\eta \frac{\mathrm{d}x_0}{\mathrm{dt}} = \kappa((x_1 - x_0) - 1), \tag{S1}$$

$$\eta \frac{\mathrm{d}x_i}{\mathrm{dt}} = \kappa((x_{i+1} - x_i) - 1) - \kappa((x_i - x_{i-1}) - 1), \quad \text{for} \quad i = 1, \ldots, n-1, \tag{S2}$$

$$\eta \frac{\mathrm{d}x_n}{\mathrm{dt}} = -\kappa((x_n - x_{n-1}) - 1). \tag{S3}$$

Where $\eta$ is the drag coefficient for the cell centres and $\kappa$ is the spring coefficient representing the mechanical interaction between cell centres. We can rearrange these equations as

$$x_1 = \frac{\eta}{\kappa}\frac{\mathrm{d}x_0}{\mathrm{dt}} + x_0 + 1, \tag{S4}$$

$$x_{i+1} = \frac{\eta}{\kappa}\frac{\mathrm{d}x_i}{\mathrm{dt}} + 2x_i - x_{i-1}, \quad \text{for} \quad i = 1, \ldots, n-1, \tag{S5}$$

$$\frac{\eta}{\kappa}\frac{\mathrm{d}x_n}{\mathrm{dt}} + x_n = 1 + x_{n-1}. \tag{S6}$$

Equations (S4) and (S5) allow the calculation of $x_1, \ldots, x_n$ given some $x_0$ and then Equation (S6) gives a further condition on the solutions. Therefore we look for a solution of the form

$$x_0 = Ae^{\lambda t}. \tag{S7}$$

Substituting Equation (S7) into Equation (S4) and then Equation (S5) allows us to calculate expressions for $x_1, \ldots, x_n$

$$x_1 = \left(\frac{\eta\lambda}{\kappa} + 1\right) Ae^{\lambda t} + 1 \qquad = P_1(\lambda)Ae^{\lambda t} + 1, \tag{S8}$$

$$x_2 = \left(\left(\frac{\eta\lambda}{\kappa}\right)^2 + 3\left(\frac{\eta\lambda}{\kappa}\right) + 1\right) Ae^{\lambda t} + 2 \qquad = P_2(\lambda)Ae^{\lambda t} + 2, \tag{S9}$$

$$x_3 = \left(\left(\frac{\eta\lambda}{\kappa}\right)^3 + 5\left(\frac{\eta\lambda}{\kappa}\right)^2 + 6\left(\frac{\eta\lambda}{\kappa}\right) + 1\right) Ae^{\lambda t} + 3 \qquad = P_3(\lambda)Ae^{\lambda t} + 3, \tag{S10}$$

$$\vdots$$

$$x_n = \frac{\eta}{\kappa}\frac{\mathrm{d}x_{n-1}}{\mathrm{dt}} + 2x_{n-1} - x_{n-2} \qquad = P_n(\lambda)Ae^{\lambda t} + n. \tag{S11}$$

Where $P_i(\lambda)$ is an $i$th degree polynomial, defined and calculated as above. We then substitute Equation (S11) into Equation (S6) to give

$$\frac{\eta\lambda}{\kappa}P_n(\lambda)Ae^{\lambda t} + P_n(\lambda)Ae^{\lambda t} + n = 1 + P_{n-1}(\lambda)Ae^{\lambda t} + (n-1). \tag{S12}$$

Which can be rearranged to

$$\left(\left(\frac{\eta\lambda}{\kappa}+1\right)P_n(\lambda)-P_{n-1}(\lambda)\right)Ae^{\lambda t}=0. \tag{S13}$$

So to avoid having a trivial solution, $x_i = i$ (i.e., no displacement from equilibrium), we need

$$\left(\frac{\eta\lambda}{\kappa}+1\right)P_n(\lambda)-P_{n-1}(\lambda)=P(\lambda)=0. \tag{S14}$$

Where $P(\lambda)$ is the $n+1$st degree characteristic polynomial of the system where the roots $\lambda_i$ give non trivial solutions. Therefore the solution, $x_0$, is of the form

$$x_0(t)=\sum_{i=0}^{n}A_ie^{\lambda_i t}. \tag{S15}$$

Where $\lambda_i$ are the roots of Equation (S14) and the coefficients $A_i$ are chosen to satisfy the initial conditions $x_i(0) = x_{i_0}$. We can combine Equation (S15) with Equations (S8)–(S11), to derive the solution for $x_i$ for $i = 1, \ldots, n$,

$$x_i(t)=\sum_{j=1}^{n}P_i(\lambda_j)A_je^{\lambda_j t}+i. \tag{S16}$$

Using this we can set up the following linear system

$$\begin{bmatrix} 1 & 1 & 1 & \cdots & 1 \\ P_1(\lambda_0) & P_1(\lambda_1) & P_1(\lambda_2) & \cdots & P_1(\lambda_n) \\ P_2(\lambda_0) & P_2(\lambda_1) & P_2(\lambda_2) & \cdots & P_2(\lambda_n) \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ P_n(\lambda_0) & P_n(\lambda_1) & P_n(\lambda_2) & \cdots & P_n(\lambda_n) \end{bmatrix}\begin{bmatrix} A_0 \\ A_1 \\ A_2 \\ \vdots \\ A_n \end{bmatrix}=\begin{bmatrix} x_{0_0} \\ x_{1_0} \\ x_{2_0} \\ \vdots \\ x_{n_0} \end{bmatrix}-\begin{bmatrix} 0 \\ 1 \\ 2 \\ \vdots \\ n \end{bmatrix}, \tag{S17}$$

which can be solved for the coefficients $A_i$. This gives us the analytic solutions to the 1D Compression problem for any initial placement of cells. An example of this analytic solutions is shown in Supplementary Figure S1 top left.

The analytic solution of the 1D Proliferation example can also be calculated by specifying the division locations and using the solution method above in between division events. An example of this analytic solution with divisions is shown in Supplementary Figure S1 top right.