PhysiCell: an Open Source Physics-Based Cell Simulator for 3-D Multicellular Systems

Ahmadreza Ghaffarizadeh[3], Randy Heiland[1], Samuel H. Friedman[3,4], Shannon M Mumenthaler[2], Paul Macklin[1,3,*]

[1]Intelligent Systems Engineering, Indiana University, Bloomington, IN 47408, USA
[2]Lawrence J. Ellison Institute for Transformative Medicine, University of Southern California, Los Angeles, CA 90033, USA
[3]formerly Lawrence J. Ellison Institute for Transformative Medicine
[4]Opto-Knowledge Systems, Inc., Torrance, CA 90502, USA

[*]Paul.Macklin@MathCancer.org

Abstract

Many multicellular systems problems can only be understood by studying how cells move, grow, divide, interact, and die. Tissue-scale dynamics emerge from systems of many interacting cells as they respond to and influence their microenvironment. The ideal "virtual laboratory" for such multicellular systems simulates both the biochemical microenvironment (the "stage") and many mechanically and biochemically interacting cells (the "players" upon the stage).

PhysiCell—physics-based multicellular simulator—is an open source agent-based simulator that provides both the stage and the players for studying many interacting cells in dynamic tissue microenvironments. It builds upon a multi-substrate biotransport solver to link cell phenotype to multiple diffusing substrates and signaling factors. It includes biologically-driven sub-models for cell cycling, apoptosis, necrosis, solid and fluid volume changes, mechanics, and motility "out of the box." The C++ code has minimal dependencies, making it simple to maintain across platforms. PhysiCell has been parallelized with OpenMP, and its performance scales linearly with the number of cells. Simulations up to 10^5-10^6 cells are feasible on quad-core desktop workstations; larger simulations are attainable on single HPC compute nodes.

We demonstrate PhysiCell by simulating the impact of necrotic core biomechanics, 3-D geometry, and stochasticity on the dynamics of hanging drop tumor spheroids and ductal carcinoma in situ (DCIS) of the breast. We also demonstrate stochastic motility, chemical and contact-based interaction of multiple cell types, and the extensibility of PhysiCell with examples in synthetic multicellular systems (a "cellular cargo delivery" system, with application to anti-cancer treatments), cancer heterogeneity, and cancer immunology. PhysiCell is a powerful multicellular systems simulator that will be continually improved with new sub-models, capabilities, and performance improvements. It also represents a significant independent code base for replicating results from other simulation platforms. The PhysiCell source code, examples, documentation, and support are available under the BSD license at `http://PhysiCell.MathCancer.org` and `http://PhysiCell.sf.net`.

Author Summary

This paper introduces PhysiCell: an open source, agent-based modeling framework for 3-D multicellular simulations. It includes a standard library of sub-models for cell fluid and solid volume changes, cycle progression, apoptosis, necrosis, mechanics, and motility. PhysiCell is directly coupled to a biotransport solver to simulate many diffusing substrates and cell-secreted signals. Each cell can dynamically update its phenotype based on its microenvironmental conditions. Users can customize or replace the included sub-models.

PhysiCell runs on a variety of platforms (Linux, OSX, and Windows) with few software dependencies. Its computational cost scales linearly in the number of cells. It is feasible to simulate 500,000 cells on quad-core desktop workstations, and millions of cells on single HPC compute nodes. We demonstrate PhysiCell by simulating the impact of necrotic core biomechanics, 3-D geometry, and stochasticity on hanging drop tumor spheroids (HDS) and ductal carcinoma in situ (DCIS) of the breast. We demonstrate contact- and chemokine-based interactions among multiple cell types with examples in synthetic multicellular bioengineering, cancer heterogeneity, and cancer immunology.

We developed PhysiCell to help the scientific community tackle multicellular systems biology problems involving many interacting cells in multi-substrate microenvironments. PhysiCell is also an independent, cross-platform codebase for replicating results from other simulators.

Introduction

Many significant multicellular systems problems—such as tissue engineering, evolution in bacterial colonies, and tumor metastasis—can only be understood by studying how individual cells grow, divide, die, move, and interact [1–5]. Tissue-scale dynamics emerge as cells are influenced by biochemical and biophysical signals in the microenvironment, even as the cells continually remodel the microenvironment. Thus, the ideal "virtual laboratory" for multicellular systems biology must simultaneously simulate (1) the dynamics of many mechanically and biochemically interacting cells, and (2) tissue microenvironments with multiple diffusing chemical signals (e.g., oxygen, drugs, and signaling factors) [5]. We recently published and open sourced the first part of such a platform: BioFVM, a biotransport solver that can efficiently simulate secretion, diffusion, uptake, and decay of multiple substrates in large 3-D microenvironments, even on desktop workstations [6]. We now introduce and release as open source PhysiCell: a mechanistic off-lattice agent-based model that extends BioFVM to simulate the tissue-scale behaviors that emerge from basic biological and biophysical cell processes.

Prior work

Several major computational frameworks are available for studying 3-D multicellular systems. CompuCell3D [7] and Morpheus [8] use Cellular Potts methods to simulate cells and their morphologies. They are very user-friendly packages with graphical model editors, integrated ODE and PDE solvers, and support for molecular-scale sub-models, but they currently cannot scale to large numbers (10^5 or more) of cells. TiSim (part of the CellSys package [9]) can simulate many more cells by using a cell-centered, off-lattice approach. However, it is currently closed source, and its executables are restricted to a limited set of simulation types. Chaste [10] is a powerful, well-developed framework for multicellular modeling with integrated PDE and ODE solvers, and both cell- and vertex-based simulations of 10^5 or more cells. However, its complex codebase has many dependencies that can impede participation by new developers; it is only cross-platform

compatible by virtual machines. Biocellion [11] can simulate billions of cells on cluster computers, but it is closed source, and its restrictive user license has hindered adoption. Timothy was recently developed to simulate large-scale colonies of 10^9 cells on high-performance supercomputers [12, 13]. Most of these platforms offer a general-purpose pre-compiled "client" that can load models and settings from an XML file; this helps overcome difficulties stemming from complex dependencies. See the supplementary materials for a detailed software comparison.

These platforms typically require users to write their own code for all cell activities, by scripting basic built-in functions. (e.g., build a cell cycle model from API functions to overwrite cell volume and duplicate agents when appropriate.) As configured "out of the box," none have built-in models for cell cycling, apoptosis, and necrosis, even though these fundamental behaviors are needed in many multicellular simulations. Only CompuCell3D and Morpheus have built-in volume regulation features. Most of these packages include PDE solvers that can simulate 3-D biotransport of one or more diffusible factors in the microenvironment, but generally they are applied sequentially to one PDE at a time, meaning that solving 10 PDEs requires 10 times more computational work than solving a single PDE. This approach is not expected to efficiently scale to 3-D simulations with many diffusible factors—a key requirement in reconciling secretomics with single-cell and multicellular systems biology, particularly as we work to understand cell-cell communication involving many cell-secreted factors.

Scientific design goals and use cases

We aim to create a framework for building multicellular simulations that investigate the relationship between (diffusional) substrate limitations, multicellular biochemical communication, and essential phenotypic processes: cell cycling/division, death, volume changes, motility, adhesion, and volume exclusion/"repulsion" [1]. Building upon our experiences in agent-based modeling [14–16], we seek a system with a minimal set of mechanics to avoid cell lattice effects [2]; hence we use an off-lattice model with basic cell adhesion and "repulsion" implemented as simple potential functions [9, 10, 14]. Following our work in [14–16], we include basic models of gradual cell volume changes, rather than static cell volumes. This avoids non-physical "jumps" in cell velocity following cell division events: the sudden localized doubling of cell density causes cells to overlap, leading to large, temporary, and non-physical "repulsive" forces that can manifest as non-physical "tears" in simulated tissues.

While PhysiCell was originally developed for problems in cancer [14], its diffusional and phenotypic processes are not specific to cancer. Users can introduce new environmental substrates (e.g., extracellular matrix (ECM) as a substrate with zero diffusion coefficient), new cell types (e.g., fibroblasts with high motility, low proliferation, and secretion and degradation of ECM), or new systems of cells (e.g., a network of vascular agents that release oxygen as in [6], and that can divide and move along gradients of angiogenic growth factors). Modelers can also use PhysiCell's core functions to create new libraries that simulate physiological systems, similar to how Microvessel Chaste [17] built upon the core functions of Chaste [10] to simulate new classes of problems in vascular remodeling. In Additional PhysiCell examples, we use PhysiCell's basic building blocks to simulate synthetic multicellular systems, heterogeneous tumors, and the innate immune system as it interacts with tumor cells. As we continue to apply PhysiCell to our own work in tissue engineering, angiogenesis, microbial dynamics, and cancer, we plan to release new optional modules that introduce new capabilities and simplify modeling of *in vivo* tissues and other systems. See Availability and Future Directions.

Thus, PhysiCell is designed to be a general-purpose toolkit for exploring multicellular systems, and the multicellular behaviors that emerge from a user-implemented set of cell phenotypic rules (i.e., a set of hypotheses). Other potential uses include

simulation-generated synthetic datasets that investigate the meaning and shortcomings of experimental measurements [18, 19]. PhysiCell's design has allowed straightforward deployment on supercomputers for high-throughput simulation studies. We recently explored over 250 instances of a 3-D cancer-immune model (see Additional PhysiCell examples) to study the impact of immune cell motility and cancer-immune adhesion dynamics on a cancer immune response, thus reducing 1.5 years' worth of simulations to 2 days on a Cray supercomputer [20, 21]. We anticipate that users can similarly use PhysiCell to efficiently explore large spaces of parameter values and hypotheses (model rules) with biophysically realistic, 3-D models. In Fall 2017, we started using PhysiCell in undergraduate and graduate education, as a self-contained tool for computational tissue engineering and cancer systems engineering. Lastly, PhysiCell also serves as an independent codebase to cross-validate model predictions in Chaste, Biocellion, TiSim, Timothy, and other platforms.

Computational design goals

PhysiCell aims to balance computational speed, built-in standard functionality, flexibility, and codebase simplicity. It includes a built-in library of standardized cell cycle and cell death models co-developed with biologists and modelers here and in the MultiCellDS standardization process [22, 23], force-based cell-cell interaction mechanics, motility, and volume regulation. Users can replace any of these built-in models with their own, and they can dynamically assign custom functions to any agent at any time. Through BioFVM, PhysiCell can couple cell phenotype to many diffusible substrates. It is the only simulation package to explicitly model the cell's fluid content—a key aspect in problems such as cryobiology [24]. It can simulate systems of 10^5-10^6 cells on desktop workstations, and 10^6 or more cells on single HPC compute nodes. All this functionality and performance is achieved with only two external dependencies, and a fully cross-platform C++ codebase that we have compiled and tested on Linux, OSX, and Windows. We also distribute PhysiCell as a virtual appliance—with a full Linux desktop, 64-bit `g++` compiler, and visualization tools—that can run in widespread virtual machine software like VirtualBox.

Design and Implementation

PhysiCell is designed to study the dynamics and interactions of thousands or millions of cells in 3-D microenvironments, with microenvironment-dependent phenotypes. It uses a lattice-free, physics-based approach to reduce grid-based artifacts. It provides optimized, biologically realistic functions for key cell behaviors, including: cell cycling (multiple models for *in vitro* and *in vivo*-focused simulations) cell death (apoptosis and necrosis), volume regulation (fluid and solid biomass; nuclear and cytoplasmic sub-volumes), motility, and cell-cell mechanical interactions. Each cell agent is built with a hierarchical `Phenotype` data structure; key phenotypic processes are triggered and controlled by modifying the phenotype data. This allows users to focus on modeling microenvironment-dependent triggers of standard cell processes, rather than coding these basic processes. However, to maintain flexibility, PhysiCell is written in a modular manner so that users can extend, rewrite, or replace its functions. Users can also create custom rules, and assign them to individual agents. It is fully coupled to a fast multi-substrate diffusion code (BioFVM) that solves for vectors of diffusing substrates, so that users can tie cell phenotype to many diffusing signals.

We note that for many problems in cancer biology and tissue engineering that drove development of PhysiCell, diffusive biotransport occurs at relatively fast time scales (on the order of 0.1 min or faster) compared to cell mechanics ($\sim$ 1 min) and cell

processes ($\sim$ 10 to 100 min or slower). PhysiCell takes advantage of this by using three separate time step sizes (Δt_{diff}, Δt_{mech}, and Δt_{cells}). In particular, the cell phenotypes and arrangement (operating on slow time scales) can be treated as quasi-static when advancing the solution to the biotransport PDEs, so BioFVM can be called without modification with the cell arrangements fixed. See Time steps: for further discussion. We provide default time step sizes that should suffice for typical applications in cancer biology and tissue engineering. Problems with inherently different time scales (e.g., advection-dominated problems, or with fast-moving bacteria) will need to adjust the time steps accordingly to avoid spurious oscillations that would be expected if cell-based sources and sinks are moving relatively quickly.

PhysiCell was built by extending the `Basic_Agent` class in BioFVM [6] (a static, non-moving object that can secrete and uptake substrates) into a fully dynamic `Cell` class with changing cell volume, cycle progression, death processes, motility, and mechanics. This allows the cells to directly and efficiently interface with the multi-substrate microenvironment. PhysiCell is written in cross-platform compatible C++ and is self-contained (with minimal dependencies). It can be compiled in any C++11 compiler with OpenMP support. This simplifies installation and improves the reproducibility of the experiments. We have tested PhysiCell on Windows through MinGW-w64, and on OSX and Linux via `g++`. (OSX users should note that Xcode does not include a compliant OpenMP-compatible `g++`; they should follow our tutorials for installing a compliant `g++` via Homebrew or MacPorts. See `http://MathCancer.org/blog/physicell-tutorials/`.) PhysiCell's only external dependencies are pugixml [25] (for XML parsing) and BioFVM [6] for 3-D multi-substrate diffusion. For the user's convencience, compatible versions of pugixml and BioFVM are included in every download.

The code has been parallelized in OpenMP to make use of multi-core desktop workstations and single HPC compute nodes. In testing, its performance (the computational time to simulate a fixed amount of time on a fixed domain) scales linearly in the number of cells present in the simulation. Simulations of up to 10^6 cells are feasible on desktop workstations, and simulations beyond 10^6 cells are possible on typical HPC compute nodes.

Overall program flow

After initializing the microenvironment (through BioFVM [6]) and cells, and initializing the current simulation time $t = 0$, PhysiCell tracks (internally) t_{mech} (the next time at which cell mechanics functions are run), t_{cells} (the next time at which cell processes are run), and t_{save} (the next simulation data output time), with output frequency Δt_{save}. Initially, we set:

$$t_{\text{mech}} = \Delta t_{\text{mech}}, \qquad t_{\text{cells}} = \Delta t_{\text{cells}}, \qquad t_{\text{save}} = 0.0. \tag{1}$$

PhysiCell repeats the main program loop until reaching the maximum simulation time:

1. Save the simulation state if $t \geq t_{\text{save}}$. Set $t_{\text{save}} = t_{\text{save}} + \Delta t_{\text{save}}$.
2. Run BioFVM to update the biochemical microenvironment for cell-based secretions and uptake, and reaction-diffusion, for the current fixed cell positions. See Biochemical microenvironment.
3. For the fixed cell positions and chemical substrate fields, if $t \geq t_{\text{cells}}$, run the cell processes for each cell:
   (a) Update cell parameters using the cell's `update_phenotype` functions. See Cell characteristics and state.

(b) Advance the cell cycle (or death) model. See Cell cycling and Cell death

(c) Advance the cell's volume (and sub-volumes) using the `volume_update_function`. See Cell volume.

Then set $t_{\text{cells}} = t_{\text{cells}} + \Delta t_{\text{cells}}$.

4. If $t \geq t_{\text{mech}}$, then:

   (a) For each cell, calculate the force-based cell velocities via `update_velocity`, add the contribution of motility (which runs `update_migration_bias` as needed), and run the custom function `custom_cell_rule`. See Cell motility and Cell mechanics and motion.

   (b) For each cell, update the position using the Adams-Bashforth method. See Cell mechanics and motion.

   Set $t_{\text{mech}} = t_{\text{mech}} + \Delta t_{\text{mech}}$.

5. Update the current simulation time by $t = t + \Delta t_{\text{diff}}$. Return to Step 1.

Steps 3a-c can be combined in a parallelized OpenMP loop; we flag cells for division and removal and process these queues serially after the parallel loop to avoid data corruption. Step 4a can be parallelized across the cells by OpenMP (because cell velocities are location-dependent, and the cell positions are fixed throughout 4a), and then Step 4b can be parallelized across the cells with these computed velocities.

Biochemical microenvironment

We use BioFVM to simulate the chemical microenvironment with a vector of reaction-diffusion PDEs with both bulk source/sinks and cell-centered sources and sinks [6]. To briefly summarize that prior work, we model the biochemical microenvironment (with computational domain Ω and boundary $\partial\Omega$, discretized as a Cartesian mesh for computational efficiency) as a vector of reaction-diffusion PDEs for a vector of chemical substrates $\boldsymbol{\rho}$ of the form

$$\begin{aligned} \frac{\partial \boldsymbol{\rho}}{\partial t} &= \overbrace{\mathbf{D}\nabla^2\boldsymbol{\rho}}^{\text{diffusion}} - \overbrace{\boldsymbol{\lambda}\boldsymbol{\rho}}^{\text{decay}} + \overbrace{\mathbf{S}(\boldsymbol{\rho}^* - \boldsymbol{\rho})}^{\text{bulk source}} - \overbrace{\mathbf{U}\boldsymbol{\rho}}^{\text{bulk uptake}} \\ &\quad + \overbrace{\sum_{\text{cells } k} \delta\left(\mathbf{x} - \mathbf{x}_k\right) W_k \left[\mathbf{S}_k(\boldsymbol{\rho}_k^* - \boldsymbol{\rho}) - \mathbf{U}_k\boldsymbol{\rho}\right]}^{\text{sources and uptake by cells}} \text{ in } \Omega \end{aligned} \tag{2}$$

with zero flux conditions on $\partial\Omega$. Here, $\delta(\mathbf{x})$ is the Dirac delta function, $\mathbf{x}_k$ is the kth cell's position, W_k is its volume, $\mathbf{S}_k$ is its vector of source rates, $\mathbf{U}_k$ is its vector of uptake rates, and $\boldsymbol{\rho}^*$ is the vector of saturation densities (the densities at which the cells stop secreting). Likewise, $\mathbf{D}$ and $\boldsymbol{\lambda}$ are the vectors of diffusion coefficients and decay rates, $\mathbf{S}$ is the bulk supply rate, and $\mathbf{U}$ is the bulk uptake function. All vector-vector products $\mathbf{ab}$ are element-wise (Hadamard product).

Numerically, we solve for the solution at time $t + \Delta t_{\text{diff}}$ by a first-order operator splitting: we first solve the bulk source/sink terms across the domain (and overwrite the stored solution), then solve the cell-centered source/sink terms (and overwrite the solution), and then the reaction-diffusion terms (again, overwriting the stored solutions). As we detailed and verified in [6], because we used (numerically stable) first-order backward differences in all our time discretizations, the overall method is first-order accurate in time and numerically stable.

After this operator splitting, the bulk source/sink terms are a decoupled set of systems of ODEs (one vector of ODEs in each computational voxel), which we solve by the backwards Euler method for first-order accuracy and numerical stability. This is trivally parallelized by OpenMP by dividing the voxels across the processor cores. Similarly, we solve the cell-based source/sink equations (one system of ODEs per cell) by backwards-Euler, overwriting the solution in the voxel containing the cell's center. As we showed in [6], this is first-order accurate and numerically stable, and it performs best when the computational voxels are comparable to the cells' sizes or larger. In all our work, we satisfy this requirement by using 20 μm voxels. As with the bulk source/sinks, we trivially parallelize by dividing the cell source/sinks (one system of ODEs per cell) across the processor cores with OpenMP.

To solve the vector of diffusion-decay PDEs, we use the locally one-dimensional method: a specialized operator splitting that turns the 3-D PDEs into a sequence of one-dimensional PDEs. In each x-strip, y-strip, or z-strip, we discretize the PDEs with second-order centered differences (spatial derivatives) and first-order backwards differences (time derivatives) to obtain a tridiagonal linear system in each strip, which we solve exactly with a tailored Thomas solver. The Thomas solver itself cannot be trivially parallelized, but because we have many simultaneous x-, y-, and z-strips, we can distribute many instances of the Thomas solver across processor cores by OpenMP. This technique, along with other optimizations (tailored overloaded vector operators, a vectorized Thomas solver, pre-computing and caching the forward sweeps of the Thomas solvers, and other similar optimizations) was tested to scale linearly in the number of substrates (number of PDEs), the number of computational voxels, and the number of discrete cell sources/sinks. The method was numerically stable even for large Δt, first-order accurate in time, and second-order accurate in space. Simulating 10 PDEs on 1,000,000 voxels takes approximately 2.6 times more computational time than simulating a single PDE. We found that for typical magnitudes of $\mathbf{D}$, $\mathbf{S}$, $\mathbf{U}$, and $\boldsymbol{\lambda}$, using $\Delta t_{\text{diff}} = 0.01$ min and $\Delta x = 20$ μm gave solutions with 5% relative accuracy or better. For more algorithmic detail and extensive convergence testing on a variety of problems, see [6] and its supplementary material.

Agent-based cell model

PhysiCell implements key cell-scale processes—cell cycling and death, volume changes, mechanics, and motility—and lets users link the parameters of these processes to microenvironmental conditions. PhysiCell then scales these basic cell-scale hypotheses to simulate thousands or millions of cells, from which tissue-scale behavior emerges. Here, we summarize the key functions. For each sub-model, see the supplementary materials for the full equations, further biological motivation, and reference parameter values.

Cell characteristics and state Cell agents have a variety of phenotypic properties, including position ($\mathbf{x}_i$), volume (and sub-volumes), cell cycle or death status, and mechanics (adhesive, deformation, and motility) parameters. Each cell has a hierarchically-structured, independent `Phenotype` object that organizes information on cell cycling, death, volume, geometry, mechanics, motility, and secretion. Each cell also has an independent `Cell_Functions` object that collects functions to update cell volume, motility, velocity (mechanics), and other properties. See the user manual (included in every PhysiCell download Version 1.2.0 or later) for a list of all the cell agents' attributes and functions to access/update them. Below, we describe standardized, built-in models to update these properties. The models can be replaced by user-defined functions; the supplied models serve as biophysically reasonable default functions that capture the key aspects of these processes.

Cell volume Each cell tracks V (total volume), V_{F} (total fluid volume), V_{S} (total solid volume), V_{NS} (nuclear solid volume), V_{CS} (cytoplasmic solid volume), V_{N} (total nuclear volume), and V_{C} (total cytoplasmic volume). Key parameters include nuclear solid, cytoplasmic solid, and fluid rate change parameters (r_{N}, r_{C}, and r_{F}), the cell's "target" fluid fraction f_{F}, target solid volume V^*_{NS}, and target cytoplasmic to nuclear volume ratio f_{CN}. For each cell, these volumes are modeled with a system of ODEs:

$$\frac{dV_{\mathrm{F}}}{dt} = r_{\mathrm{F}}\left(V^*_{\mathrm{F}}(t) - V_{\mathrm{F}}\right) \tag{3}$$

$$\frac{dV_{\mathrm{NS}}}{dt} = r_{\mathrm{N}}\left(V^*_{\mathrm{NS}}(t) - V_{\mathrm{NS}}\right) \tag{4}$$

$$\frac{dV_{\mathrm{CS}}}{dt} = r_{\mathrm{C}}\left(V^*_{\mathrm{CS}}(t) - V_{\mathrm{CS}}\right) \tag{5}$$

where we use the constitutive laws

$$V^*_{\mathrm{CS}}(t) = f_{\mathrm{CN}} V^*_{\mathrm{NS}}(t) \tag{6}$$

$$V^*_{\mathrm{F}}(t) = f_{\mathrm{F}} V(t). \tag{7}$$

These parameters are updated as the cell progresses through its current cycle or death process. (See Cell cycling and Cell death.) For example, we halve V^*_{NS} after a cell division and double it upon re-entry to the cell cycle. We set $V^*_{\mathrm{NS}} = f_{\mathrm{CN}} = 0$ at the onset of apoptosis, and we set r_{F}, r_{C}, and r_{N} based on key apoptosis time scales. See the supplementary materials for the full equations, sample solution plots, further biological motivation, convergence testing, and reference parameter values. These ODEs are numerically solved using first-order forward Euler discretizations.

Cell motility Each cell can set its own persistence time (T_{per}), migration speed (s_{mot}), migration bias direction ($\mathbf{d}_{\mathrm{bias}}$, a unit vector), and a migration bias b that ranges from 0 (Brownian motion) to 1 (deterministic motion along $\mathbf{d}_{\mathrm{bias}}$). When updating the cell's velocity, its migration velocity $\mathbf{v}_{\mathrm{mot}}$ is added to the currently velocity (as calculated by mechanics; see Cell mechanics and motion). The cell changes it migration velocity stochastically between t and $t + \Delta t_{\mathrm{mech}}$ with probability

$$\mathrm{Prob}(\text{change } \mathbf{v}_{\mathrm{mot}}) = \frac{\Delta t_{\mathrm{mech}}}{T_{\mathrm{per}}} \tag{8}$$

The user-defined function `update_migration_bias` (see the user manual and supplementary material) sets $\mathbf{v}_{\mathrm{bias}}$, b, and s_{mot}. The migration velocity $\mathbf{v}_{\mathrm{mot}}$ is then updated according to

$$\mathbf{v}_{\mathrm{mot}} = s_{\mathrm{mot}} \frac{(1-b)\boldsymbol{\xi} + b\mathbf{d}_{\mathrm{bias}}}{||(1-b)\boldsymbol{\xi} + b\mathbf{d}_{\mathrm{bias}}||} \tag{9}$$

where $\boldsymbol{\xi}$ is a random unit vector. See the user manual and supplementary materials for more details, as well as examples for biased migration along chemical gradients.

Cell mechanics and motion We model cell mechanics and motion as in our prior work [14]: we update each cell's position $\mathbf{x}_i$ by calculating its current velocity $\mathbf{v}_i$ based upon the balance of forces acting upon it. The main forces include cell motility, drag-like forces, and cell-cell and cell-matrix interaction forces: adhesion and "repulsion" (resistance to deformation and/or volume exclusion [26]). As in prior cell-centered models [14, 27, 28], we apply an inertialess assumption ($m_i \dot{\mathbf{v}}_i \approx \mathbf{0}$) to explicitly solve for each cell's velocity. As before [14], we model adhesion and repulsion with interaction

potentials that depend upon each cell's size, maximum adhesion distance, adhesion and repulsion parameters, and distance to other cells. The cell's velocity $\mathbf{v}_i$ is given by

$$\mathbf{v}_i = \sum_{j \in \mathcal{N}(i)} \left(-\overbrace{\sqrt{c^i_{\text{cca}} c^j_{\text{cca}}} \nabla \phi_{1,R_{i,A}+R_{j,A}} \left(\mathbf{x}_i - \mathbf{x}_j\right)}^{\text{cell-cell adhesion}} - \overbrace{\sqrt{c^i_{\text{ccr}} c^j_{\text{ccr}}} \nabla \psi_{1,R_i+R_j} \left(\mathbf{x}_i - \mathbf{x}_j\right)}^{\text{cell-cell repulsion}} \right) .$$
$$- \overbrace{c^i_{\text{cba}} \nabla \phi_{1,R_{i,A}} \left(-d(\mathbf{x}_i)\mathbf{n}(\mathbf{x}_j)\right)}^{\text{cell-BM adhesion}} - \overbrace{c^i_{\text{cbr}} \nabla \psi_{1,R_i} \left(-d(\mathbf{x}_i)\mathbf{n}(\mathbf{x}_j)\right)}^{\text{cell-BM repulsion}} + \mathbf{v}_{i,\text{mot}} \tag{10}$$

where as in [14] $\phi_{n,R}(\mathbf{x})$ is an adhesion interaction potential function which is zero for $||\mathbf{x}|| > R$, and approaches zero with smoothness given by n. Similarly, $\psi_{n,R}(\mathbf{x})$ is a repulsion interaction potential function that is zero for $||\mathbf{x}|| > R$. Thus, cell-cell mechanical interactions occur over finite distances. Here, c^i_{cca} and c^i_{ccr} are the ith cell's cell-cell adhesion and repulsion parameters, R_i is its radius, and $R_{i,A}$ is its maximum adhesion distance (typically a fixed multiple of R_i). Note that if cell i and j have identical cell-cell adhesion and repulsion parameters c_{cca} and c_{ccr}, then the cell-cell interaction coefficients simplify to the form we published in [14]; otherwise, this is a phenomenologic form that allows either cell's coefficient to affect the cell-cell interaction strength, with equal and opposite effect on both cells. We plan future improvements to move beyond the current phenomenologic forms to more mechanistic models.

Also $d(\mathbf{x})$ is the distance to the nearest basement membrane (if any), $\mathbf{n}(\mathbf{x})$ is a unit vector normal to the basement membrane, and so $-d(\mathbf{x}_i)\mathbf{n}\left(\mathbf{x}_i\right)$ points from the cell's position $\mathbf{x}_i$ to the nearest point on the basement membrane, located at $\mathbf{x}_i - d\left(\mathbf{x}_i\right)\mathbf{n}\left(\mathbf{x}_i\right)$. (See [29–31] for more information on level set (distance function) representations of surfaces, there applied to tumor growth models.) $\mathcal{N}(i)$ is the (finite) list of cells that could potentially interact with cell i. (See Key code optimizations.) Further references and the full forms of the potential functions are given in the supplementary materials.

The cell's position is updated using the second-order Adam's Bashforth discretization:

$$\mathbf{x}_i\left(t + \Delta t_{\text{mech}}\right) = \mathbf{x}_i(t) + \frac{1}{2}\Delta t_{\text{mech}}\Big(3\mathbf{v}_i(t) - \mathbf{v}_i\left(t - \Delta t_{\text{mech}}\right)\Big) \tag{11}$$

Cell cycling PhysiCell includes a cell cycle modeling framework, where each cell cycle model is a collection of phases $\{X_i\}$, transition rates $\{r_{ij}\}$ between the phases, and a cell division phase transition. As of Version 1.2.0, users can also set phase entry and exit functions (associated with the phases X_i) that are executed at entry into or exit from the phase; these can be used to model processes such as mutation of cell parameters. The framework also allows users to set arrest functions (associated with the transition rates r_{ij}) that block the transition. This is useful for modeling effects like volume restrictions. See the user manual for full details. As in [14], we use the phase transition rates to calculate the phase change probabilities in any time interval $[t, t + \Delta t]$: the probability of transitioning from phase X_i to phase X_j in this time interval is given by

$$\text{Prob}\left(\text{transition from } X_i \text{ to } X_j | \text{ not arrested }\right) \approx r_{ij}\Delta t. \tag{12}$$

Users can set individual transitions r_{ij} to have deterministic duration, with duration $1/r_{ij}$. See the user manual for full details.

Each cell agent tracks its current cell cycle phase $\mathcal{S}_k$ and its total time spent in that phase (t_k). Users can change the transition rates at any time, in part based upon microenvironmental conditions (e.g., based upon oxygenation or cell contact).

As a concrete example, consider the "Ki67 Advanced" model from our prior work calibrating oxygen-dependent growth to Ki67 data in ductal carcinoma in situ (DCIS)

[14, 15, 32]. The phases are K_1 (Ki67+ cycling cells, prior to cell division), K_2 (Ki67+ cycling cells, after cell division), and Q (Ki67- quiescent cells). K_1 and K_2 have stochastic durations (with means T_1 and T_2). We model the transition rate from Q to K_1 as

$$r_{Q1} = \frac{1}{\overline{T}_{\mathrm{Q}}} \max\left\{ \left(\frac{\mathrm{pO}_2 - \mathrm{pO}_{2,\mathrm{hypoxia}}}{\overline{\mathrm{pO}_2} - \mathrm{pO}_{2,\mathrm{hypoxia}}} \right), 0 \right\}, \tag{13}$$

where cells spend a mean time of $\overline{T}_{\mathrm{Q}}$ in the Q phase when $\mathrm{pO}_2 = \overline{\mathrm{pO}_2}$. Cells double V^*_{NS} when transitioning from Q to K_1 (to double their nuclear content), and they halve V^*_{NS} (and all the sub-volumes) when dividing into two daughter cells at the $K_1 \longrightarrow K_2$ transition. The full set of supported cell cycle models—along with reference parameter values—is given in the supplementary materials.

Cell death PhysiCell currently includes models for two types of cell death: apoptosis (programmed cell death) and necrosis (unprogrammed cell death) [33]. At any time, each agent (with index i) has two death rates ($r_{\mathrm{A},i}$ for apoptosis, and $r_{\mathrm{N},i}$ for necrosis), which can be continually updated. For any death rate r_i and any time interval $[t, t+\Delta t]$, the cell has a probability of entering the corresponding death state D_i:

$$\mathrm{Prob}\Big(\mathcal{S}_i(t+\Delta t) = D_i\Big) = 1 - \exp\big(-r_i \Delta t\big) \approx r_i \Delta t. \tag{14}$$

Apoptosis: Upon entering the apoptotic state, we set $f_{\mathrm{CN}} = 0$ (to simulate shrinking and blebbing of the cytoplasm), $V^*_{\mathrm{NS}} = 0$ (to simulate degradation of the nucleus), and $f_{\mathrm{F}} = 0$ (to simulate the active elimination of water from the cell). The rates r_{N}, r_{F}, and r_{C} are set to match time scales of cell volume loss in apoptotic cells. The cell is removed once its volume drops below a user-set threshold, or after mean duration of T_{A}.

Necrosis: When a cell becomes necrotic, we set $f_{\mathrm{CN}} = V^*_{\mathrm{NS}} = 0$ to model cytoplasmic and nuclear degradation. Early necrotic cells undergo oncosis (cell death-related swelling); we model this by setting $f_{\mathrm{F}} = 1$. (Note that some regard oncosis as the actual death process, and necrosis as post-mortem cell degradation [34, 35].) Once the cell volume passes a critical threshold, it lyses, and we set $f_{\mathrm{F}} = 0$. The rate parameters r_{F}, r_{N}, and r_{C} are set to match expected time scales throughout necrosis [33]. PhysiCell includes codes to trigger necrosis deterministically or stochastically:

Deterministic Necrosis: This implements a common model of necrosis (see the review [2]), where cells instantly become necrotic whenever oxygenation pO_2 drops below a threshold value $\mathrm{pO}_{2,\mathrm{threshold}}$, as in our earlier work [14]. This is equivalent to the letting $r_{\mathrm{N}} \to \infty$.

Stochastic Necrosis: This model updates our prior work [14], based upon *in vitro* observations that cells can survive low oxygen conditions for hours or days. Here,

$$r_{\mathrm{N}}\left(\mathrm{pO}_2\right) = \begin{cases} 0 & \text{if } \mathrm{pO}_{2,\mathrm{threshold}} < \mathrm{pO}_2 \\ r_{\mathrm{N,max}} \left(\frac{\mathrm{pO}_{2,\mathrm{threshold}} - \mathrm{pO}_2}{\mathrm{pO}_{2,\mathrm{threshold}} - \mathrm{pO}_{2,\mathrm{crit}}} \right) & \text{if } \mathrm{pO}_{2,\mathrm{crit}} < \mathrm{pO}_2 \le \mathrm{pO}_{2,\mathrm{threshold}} \\ r_{\mathrm{N,max}} & \text{if } \mathrm{pO}_2 \le \mathrm{pO}_{2,\mathrm{crit}}. \end{cases} \tag{15}$$

That is, necrotic death begins when $\mathrm{pO}_2 < \mathrm{pO}_{2,\mathrm{threshold}}$, and the death rate ramps linearly until saturating at a maximum rate $r_{\mathrm{N,max}}$ for $\mathrm{pO}_2 < \mathrm{pO}_{2,\mathrm{crit}}$. Equivalently, cells survive on average $1/r_{\mathrm{N,max}}$ time in very low oxygen conditions [14].

Numerical implementation

Time steps: PhysiCell has three time steps to model (fast) diffusive biotransport processes (Δt_{diff}; default 0.01 min), cell movement (Δt_{mech}; default 0.1 min), and (relatively slow) cell processes (Δt_{cells}; default 6 min). We use these time steps to set how frequently biotransport processes, cell movement processes, and cell phenotype processes are updated. See Fig. 1.

The default Δt_{diff} was chosen for diffusion, decay, and uptake/secretion parameter values typical for the cancer and tissue engineering problems that drove PhysiCell's development. In prior testing, relative errors did not exceed 5% for this value [6]. In mechanical relaxation tests for overlapping cells and compressed tumor spheroids, we found that $\Delta t_{\text{mech}} = 0.1$ min gave solutions that converged at first-to-second order accuracy, had relative errors 5% or less, and avoided spurious oscillations and other artifacts for cell velocities under ~ 1 μm/min (typical for cancer biology problems); see the Supplementary Materials. The cell cycle, death, and volume change models were numerically stable and first-order accurate with relative errors of 5% or less for $\Delta t_{\text{cells}} = 6$ min. See the Supplementary Materials. Users should reduce Δt_{cells} for problems with faster phenotypic processes. Users anticipating faster cell movement (e.g., motile bacteria) should reduce Δt_{mech}. We recommend setting $\Delta t_{\text{diff}} \leq \frac{1}{10} \Delta t_{\text{mech}}$.

Mathematically, this time scale separation allows us to hold cell positions fixed (quasi-static) when updating the PDE solutions, and then hold the chemical fields fixed when updating cell positions and phenotypes. We have used similar techniques in nonlinear continuum models of tumor growth (slow time scale) in heterogeneous biochemical microenvironments (fast time scale) [29–31].

Estimated computational cost scaling We now assess the computational effort needed for each iteration in the main program loop. (Reference Overall program flow.) Step 1 (save simulation data), Step 3 (run cell processes), and Step 4b (update positions) clearly entail a constant amount of work for each cell. Thus, summing these steps over all cells $n(t)$ requires $\mathcal{O}(n)$ work. By prior analysis, BioFVM (Step 2) also scales linearly in $n(t)$ [6].

Step 4a (update velocities) is the most computationally expensive step. In straightforward implementations, each cell tests for mechanical interaction with $n-1$ other cells, giving an $\mathcal{O}(n^2)$ total computational cost at each time step. However, the interaction data structure (see Key code optimizations) restricts interaction testing to a smaller set $\mathcal{N}(i)$.

In the appendix, we show that each $\mathcal{N}(i)$ has at most N_{max} cells. Thus, Step 4b has a fixed maximum cost for each cell, and the cost of the loop scales linearly in n.

Key code optimizations To prevent computational costs from scaling quadratically in the number of cells, we designed a cell-cell interaction data structure (IDS) that efficiently estimates a set $\mathcal{N}$ of possible neighbor cells for each cell agent. See the supplementary material for further detail.

PhysiCell uses OpenMP to parallelize most loops over the list of cells. This includes sampling the microenvironment, updating cell phenotype parameters, advancing the cell cycle or death model, advancing the volume model, running any custom function, and calculating the cell velocity. We do not parallelize loops that change the IDS: cell division, cell removal, and updating the cell position.

As discussed above, we defined three separate computational step sizes ($\Delta t_{\text{diff}} < \Delta t_{\text{mech}} < \Delta t_{\text{cells}}$) to take advantage of the multiple time scales of the multicellular system. As indicated in the overall program flow above, we update each process according to its own time step, rather than at each simulation step. Fig. 1 further illustrates how the

multiple times steps reduce the computational cost. See the supplementary materials for further detail and the default step sizes for cancer biology.

Convergence and validation testing

We performed convergence testing on all the major components of PhysiCell. BioFVM was previously tested as first-order accurate in Δt, second-order accurate in Δx, and sufficiently accurate at $\Delta x = 20\ \mu$m and $\Delta t_{\text{diff}} = 0.01$ to 0.05 min for tumor growth problems [6]. We performed two tests for cell-cell mechanics and motion: First, we placed two cells in partial overlap, simulated their relaxation to equilibrium, and measured the cell spacing at several times. Second, we created a compressed cluster of 50,000 cells, simulated its mechanical relaxation to equilibrium, and measured its diameter at several times. Both tests converged to first-order accuracy in Δt at all measured times, showing that PhysiCell converged in both short-time mechanical dynamics and in long-time behavior. $\Delta t_{\text{mech}} \sim 0.1$ min gives sufficient accuracy for typical cancer problems.

We simulated the volume model for a single proliferating, apoptotic, and necrotic cell, and measured the sub-volumes at multiple times. It converged with first-order accuracy in Δt at all tested times, and $\Delta t_{\text{cell}} = 6$ min gave sufficient accuracy. We tested the stochastic transition codes by simulating the Ki67-advanced cell cycle model and the apoptosis death model (with stochastic duration), and measuring the sub-population counts and population fractions over time for several values of Δt_{cell}. For each Δt, we performed 100 simulations and compared the mean solution behavior against known coarse-grained ODE model behavior. $\Delta t_{\text{cell}} = 6$ min and 60 min both gave an excellent match between the PhysiCell behavior and theory for all the compared curves. See the supplementary materials for full testing results.

Performance testing (summary)

By our testing, recent quad-core desktop workstations (with hyperthreading, for 8 total execution threads) can simulate 10-30 days in systems of up to 10^5 to 10^6 cells in 3 days or less (wall time). Single HPC compute nodes (typically two 6-8 core processors, with hyperthreading and 24-32 execution threads) can simulate larger systems up to $\sim$2 million cells in about 2 days. Future releases of PhysiCell will address current performance bottlenecks; see Availability and Future Directions. The Results will give a demonstration of $\mathcal{O}(n)$ computational cost scaling.

Results

We demonstrated PhysiCell's potential to simulate large multicellular systems—and its ability to test the emergent tissue-scale effects of cell-scale hypotheses—on several examples arising from cancer biology and synthetic multicellular systems bioengineering. For the first two examples, we compared the impact of the deterministic and stochastic necrosis models. (See Cell death above.) We used the Ki67-advanced cell cycle model with deterministic K_1, K_2, and A phase durations for the first two examples. (See Cell cycling.) We used a simpler "live cells" cycle model [22] in the remaining examples, where live cells proliferate with a variable birthrate, apoptose, or necrose. We provide detailed parameter values in the supplementary materials for the HDS and DCIS examples, and the full source code and postprocessing routines for both examples in every PhysiCell download. Full source code and simple build instructions are provided for the remaining examples. Reference simulation outputs for the first two examples are available at `http://PhysiCell.MathCancer.org`.

Test platforms

The Hanging drop tumor spheroids and Ductal carcinoma in situ (DCIS) examples were tested on (1) a desktop workstation (quad-core Intel i7-4790, 3.60 GHz, 8 execution threads, 16 GB memory) with mingw-w64 (`g++` ver. 4.9.1) on 64-bit Windows 7, and (2) a single HPC compute node (dual 6-core Intel Xeon X5690, 3.47 GHz, 24 execution threads, 48 GB memory) with `g++` (ver. 4.8.4) on Ubuntu 14.04. The tests were performed using PhysiCell 1.0.0, although release 1.2.0 has updated the tests for compatibility. The CPU architecture was newer on the desktop (2014 Haswell) than on the HPC node (2011 Westmere). The newer "Biorobots", Anti-cancer biorobots, Cancer heterogeneity and immune response, and Adding an immune response examples were tested on a quad-core Intel i7-4770K, 4.06 GHz, 8 execution threads, 32 GB memory, using PhysiCell Versions 1.2.1 with `g++` 7.1.0 on 64-bit Windows 10 (via MinGW-w64).

Hanging drop tumor spheroids

Hanging drop spheroids (HDS)—a 3-D cell culture model where a small cluster or aggregate of tumor cells is suspended in a drop of growth medium by surface tension—are increasingly used to approximate 3-D *in vivo* growth conditions [36]. Unlike traditional 2-D monolayer experiments, HDSs allow scientists to investigate the impact of substrate gradients on tumor growth, particularly oxygen gradients. Their relatively simple geometry makes them ideal for testing computational models.

We simulated HDS growth by placing an initial cluster of ~ 2300 cells in an 8 mm^3 fluid domain, with Dirichlet conditions $\text{pO}_2 = 38$ mmHg (5% oxygen: physioxic conditions [37]) on the computational boundary. The simulation results are shown in Fig. 2 for deterministic necrosis (left column) and stochastic necrosis (right column), at 4, 8, and 16 days. In Fig. 3, we show the tumor diameter (left panel) and number of agents (right panel) versus time. Both simulations reached $\sim 10^6$ cells by 18 days. See the simulation videos Video S1 and Video S2.

Deterministic versus stochastic necrosis Both models yielded similar dynamics. Hypoxic gradients emerged quickly, limiting (pO_2-dependent) cell division to the outermost portions of the tumors. This, in turn, lead the tumor diameters to grow linearly (at similar rates); see Fig. 3. This matches our theoretical expectations for a spheroid of radius $R(t)$ whose growth is restricted to an outer layer of fixed thickness T:

$$\frac{d}{dt}V(t) = c \cdot \overbrace{4\pi R^2(t) \cdot T}^{\text{growing region}} \implies \frac{d}{dt}\left(\frac{4}{3}\pi R^3(t)\right) = c \cdot 4\pi R^2(t) \cdot T \tag{16}$$

$$\implies \frac{d}{dt}R(t) = cT = \text{ constant.} \tag{17}$$

In both models, the innermost portion of the necrotic core developed a network of fluid voids or cracks. This phenomenon emerges from competing biophysical effects of the multicellular system and its cell-scale mechanical details: necrotic cells lose volume, even as they continue to adhere, leading to the formation of cracks. To our knowledge, this is the only model that has predicted this necrotic tumor microarchitecture, which would be very difficult to simulate by continuum methods except with very high-resolution meshes comparable to the ~ 1 to 10 μm feature size. These cracked necrotic core structures have been observed with *in vitro* hanging drop spheroids (e.g., [5, 38, 39]) See Fig. 2.

There were notable differences between the models. The deterministic model had a sharp perinecrotic boundary between the viable and necrotic tissues, whereas the stochastic model demonstrated a perinecrotic transition zone with substantial mixing of viable and necrotic cells. Because cells do not immediately necrose in the stochastic

model, it retained a center of quiescent viable cells longer than the deterministic model. The growth curves for the deterministic and stochastic models appear to diverge after approximately 8 days, when the deterministic necrotic core is better defined with more cracks than the stochastic core. This may be due to differences in hypoxic gradients (the tumor with more void spaces will have shallower oxygen gradients, and hence more cell cycle entry), but further simulations would be required to rule out stochastic effects. Interestingly, the stochastic model's growth curve appears to run parallel to the deterministic curve for later times, once its necrotic core becomes better defined.

Performance scaling Throughout the simulations, the computational cost (the wall time required to simulate one hour) scaled approximately linearly with the number of agents present in the simulation, on both the desktop workstation and the HPC node; see Fig. 4. (See also Estimated computational cost scaling.) Increasing the number of execution threads improved performance, even when running on slower processor cores. See the right panel in Fig. 4, where moving from the newer 8-threaded machine to the older 24-threaded machine improved performance by a factor of 2 to 2.5.

The simulations reached $\sim 10^6$ cells on our HPC tests after 67 hours (deterministic, 17 simulated days) to 76 hours (stochastic, 18.2 simulated days) of wall time, including saving full simulation output data once per simulated hour. See Fig. 3. The desktop workstation simulated past 573,000 cells (about 14.6 days of simulated time) in approximately 80 hours of wall time. The desktop tests did not run out of memory, and the simulations can be completed to the full 18 days and 10^6 cells if needed.

Ductal carcinoma in situ (DCIS)

DCIS is a pre-malignant breast condition where epithelial cells ("tumor cells") divide abnormally to fill the breast duct lumen. Oxygen can only reach the tumor cells by diffusion from outside the duct, leading to the emergence of hypoxia and an inner necrotic core. See [14, 15, 32] for further biological and clinical discussion. As in [14], we approximate a partly-filled breast duct as a 3-D "test tube" with a level set function representation. Cell adhere to cells and the duct wall; cells and the duct wall push against cells to resist deformation. Oxygen diffuses from the duct wall and is consumed by tumor cells. The rate of cycle entry increases linearly with pO_2 (see Cell cycling).

In Fig. 5, we show DCIS simulations in a 1 mm segment of breast duct (317.5 μm diameter), using deterministic necrosis (left side) and stochastic necrosis (right side), plotted at 10 and 30 days. See also Video S3 and Video S4. As in prior work [14], the simulations predict cell-scale details observed in DCIS pathology, such as the appearance of pairs of Ki67+ daughter cells, the spatially isolated apoptotic cells (which arises from the model assumption that apoptosis is a stochastic, low-frequency event that is independent of oxygenation), and the higher occurrence of Ki67+ cells near the duct wall (where we modeled the probability of cell cycle entry as proportional to oxygenation).

Comparison of necrosis models; comparison with the spheroid example As in the HDS example, the deterministic model had a sharp, smooth perinecrotic boundary, whereas the stochastic model demonstrated a perinecrotic boundary region with mixed viable and necrotic cells. In the stochastic model, proliferation halted in the duct center, but necrosis appeared later. The perinecrotic mixing effect was most pronounced at the leading edge of the tumor, where tissue was transitioning from non-hypoxic/non-necrotic to necrotic. Areas with longer-term hypoxia had smoother necrotic boundaries. This effect did not emerge in the HDS example due to its symmetry.

Interestingly, the mechanical "cracks" seen in the tumor spheroids do not appear here, because the breast duct compresses the necrotic core to collapse any fluid-filled

voids. This shows the importance of the 3-D geometry and the biophysical impact of the basement membrane, as well as the need to account for such effects when approximating *in vivo* conditions with bioengineered model systems.

Both models gave approximately the same growth rate of ~ 1 cm/year (Fig. 6, left). We cannot select one model over the other based solely upon continuum-scale, coarse-grained outputs. However, we could further assess the models by comparing their distinct differences in multicellular-scale patterning to DCIS pathology. This further highlights the need and potential for multicellular modeling in evaluating cell-scale hypotheses.

Comparison with prior 2-D modeling results In 3D, neither necrosis model reproduced the mechanical "tears" between the proliferative rim and the necrotic core predicted by earlier 2-D simulations [14]; this is because more viable tissue is fluxing into smaller necrotic areas in the 3-D geometry compared to the 2-D geometry.

Additional PhysiCell examples

In the following examples, we demonstrate PhysiCell in applications involving interactions via contact and chemical factors between multiple cell types, in 2-D and 3-D simulations. Most of the examples include cell motility, "custom" phenotype rules, additional "custom" mechanics, and custom data.

Please visit `http://www.mathcancer.org/blog/physicell-sample-projects/` for instructions on building, running, and visualizing the following sample projects. The sample projects are structured to mirror the intended design of user projects: a project-tailored `Makefile` and `main.cpp` in the main directory, and most custom code in the project-tailored functions in the `custom_modules` directory.

The instructions can also be found in the User Guide (found in the documentation folder of every PhysiCell download) and the Quickstart guide (found in the root directory of every PhysiCell download).

"Biorobots" First, we tested PhysiCell for its potential in aiding in the design of synthetic multicellular systems. We investigated rulesets to create a cellular cargo delivery system, including the following main components:

1. "Director" cells secrete a diffusible chemoattractant c_1 and are otherwise static. They use the typical cell-cell repulsion mechanics.

2. "Cargo" cells have a surface receptor $0 \leq R \leq 1$. When $R = 1$, they secrete a diffusible chemoattractant c_2. When $R = 0$, they stop secreting the factor. They use the typical cell-cell repulsion mechanics.

3. "Worker" cells can either be adhered or not adhered to cargo cells.

   (a) When they are unadhered, they perform biased random migration towards gradients of ∇c_2. Whenever they touch a cell, they test for presence of receptor R. If R is expressed (i.e., the worker has found cargo), the worker forms an elastic adhesion with the cell and sets $R = 0$ on the cargo cell.

   (b) When they are adhered, they perform biased random migration towards ∇c_1. When c_1 exceeds a threshold, they break their adhesive bond.

As a simple model of this targeted cell adhesion, we used a `custom_cell_rule` to implement:

$$\mathbf{v}_{i,\text{Hook}} = k\left(\mathbf{x}_j - \mathbf{x}_i\right) \tag{18}$$

$$\mathbf{v}_{j,\text{Hook}} = k\left(\mathbf{x}_i - \mathbf{x}_j\right) \tag{19}$$

for adhered cells j and i. We then add these to the cell's velocities. PhysiCell evaluates the custom rule when evaluating cell mechanics (velocity update).

Simulation outputs are shown in Fig. 7; see also Video S5. As we can see, the cargo and worker cells successfully interact to modulate their behaviors. Notably, the worker cells are seen making multiple transits from the supply of cellular cargo to the directors, showing the robustness of both the rules and their implementation in PhysiCell.

This and the following examples also demonstrate the intended arrangement of projects: users do not modify the contents of `./core/`, but instead place their codes in `./custom_modules/`, include these in a main project `main.cpp`, and modify the `Makefile` to compile and link the components. This design is intended to allow users to update the PhysiCell and BioFVM core components without overwriting their own customizations.

Anti-cancer biorobots We adapted the "biorobots" to test their potential as an anti-cancer treatment. Many proposed cancer therapies attempt to target cancer cells by finding unique surface or other molecules to target, so that drugs can be conjugated to custom antibodies or encapsulated in custom nanoparticles. These still generally rely upon passive delivery of the therapeutics, even though cancers are often poorly perfused. See e.g. [40, 41]. We tested the "biorobots" as a potential solution, with the following modifications:

1. There are no "director" cells. Instead, cancer cells consume oxygen (as in prior examples), which creates an oxygen gradient that can be leveraged for worker cell "homing."

2. Adhered "cargo" cells detach themselves from "worker" cells when $\text{pO}_2 < \text{pO}_{2,\text{drop}}$ and secrete a therapeutic compound [drug] that diffuses with the typical form given in Biochemical microenvironment.

3. Adhered worker cells perform biased random migration along $-\nabla \text{pO}_2$.

4. In any time interval $[t, t + \Delta t_{\text{cells}}]$, cancer cells accumulate drug-induced damage according to:

$$\frac{d}{dt}\left[\text{damage}\right] = r_{\text{damage}}\left[\text{drug}\right] - r_{\text{repair}}\left[\text{damage}\right] \tag{20}$$

   where r_{damage} and r_{repair} are the damage and repair rates. Each tumor cell agent tracks its own level of damage. In the same time interval the probability of a cell apoptosing due to the drug is

$$\text{Prob}\left(\text{apoptosis}\right) = r_{\text{death}}\left[\text{drug}\right]\Delta t_{\text{cells}}, \tag{21}$$

   where r_{death} is the death rate when $[\text{damage}] = 1$.

5. Unadhered worker cells disabled motility if c_1 falls below a threshold value.

We simulated 1 week of tumor growth, and "injected" a mixture of 10% worker cells and 90% cargo cells near the tumor, and set the parameter $\text{pO}_{2,\text{drop}}$ to 10 mmHg. In the simulation (Fig. 8 and Video S6), overall the cargo cells were delivered into the

tumor (and they can be seen pushing tumor cells out of the way), tumor cells were indeed killed. However, once enough cancer cells were killed, hypoxia was reduced so that worker cells clustered near the oxygen minimum, but no longer released their "cargo" (because $pO_2 > pO_{2,\text{drop}}$ throughout the domain). Setting $pO_{2,\text{drop}} = 15$ mmHg reduced but did not eliminate this behavior. See Fig. 8 and Video S6. Thus, an "anti-cancer biorobot" system as explored here could potentially be beneficial (in particular, homing towards and penetrating tumors without need for cancer-specific targets), but the "cargo release" rules need to be carefully engineered. Such a system could potentially activate and deactivate to keep a tumor cell population in control, and to reduce hypoxia (which is known to drive cancer cell adaptation to more aggressive phenotypes [42, 43]).

Cancer heterogeneity and immune response Next, we applied PhysiCell to another area of interest in the cancer community: heterogeneity. We seeded an initial tumor, and assigned each cell a random expression of a mutant "oncoprotein" $0 \leq p \leq 2$. We started the simulation with a normal distribution of p, with mean 1 and standard deviation of approximately 0.3. We used the same oxygen-dependent stochastic cell cycle entry and necrosis as in prior examples. We modified the model to set the rate of cycle entry to scale proportionally to p, so that increased p increases the rate of cell division. The initial, intermediate, and final morphologies of the tumor are plotted in Fig. 9, along with the changing histograms of p. In the plot, cell color ranges from blue ($p = 0$) to yellow ($p = 2$). We can see clear selection for the yellower cells with greater expression of p. Moreover, see that while the initial tumor began with an uniform "salt and pepper" distribution of blue and yellow cells and a symmetric morphology, symmetry was broken by the end as regions with higher initial p and greater access to oxygen form dominant focal growths of "yellow" clones. This demonstrates further stochasticity in the simulation that would be difficult to predict with continuum approaches. See Video S7 to better examine these dynamics.

Adding an immune response We extended the heterogeneity example to 3D, and integrated a simple model of an immune attack on the heterogeneous tumor:

1. All cancer cells consume oxygen as before. They also secrete an immunostimulatory factor c (e.g., a chemokine such as basic fibroblast growth factor (bFGF) [44]), which diffuses according to the standard PDE we introduced earlier.

2. As a simple model of immunogenicity, the mutant oncoprotein is assumed to increase immunogenicity proportionally to p, similarly to mutant tumor-associated epitopes being presented on MHCs (major histocompatibility complexes) [45, 46].

3. Unadhered immune cells perform biased random migration (in our simulations $b = 0.5$) towards ∇c and test for contact with cells. If they detect contact, they form an adhesion using the same model as in the "Biorobots" example and switch off motility. In any time interval $[t, t + \Delta t]$, we give the cell a probability of forming an adhesion regulated by

$$\text{Prob(forming a new adhesion to the tested cell)} = r_{\text{adhesion}} \Delta t, \tag{22}$$

   where r_{adhesion} is the immune cell's rate of forming new cell adhesions.

4. While adhered, immune cells attempt to induce apoptosis in the adhered cell with probability

$$\text{Prob(inducing apoptosis)} = r_{\text{kill}} p \Delta t, \tag{23}$$

   where p is used as our surrogate marker for immunogenicity (see above), and r_{kill} is the rate at which adhered immune cells kill tumor cells with $p = 1$. If they induce apoptosis, they detach and resume their search for new cells.

5. Adhered immune cells have a probability of detachment given by

$$\text{Prob(detaching from an adhered tumor cell)} = r_{\text{detach}} \Delta t. \tag{24}$$

In Fig. 10 and Video S8, we first simulate two weeks of tumor growth without an immune reaction. As in the prior example, we see the initially symmetric tumor develop asymmetric focal growths of "yellow" cells, again showing the selection for cells with the most oncoprotein. At two weeks, we introduce 7500 immune cells (red) that invade the tumor. Apoptotic tumor cells are labeled in cyan. The immune cells continue to migrate towards the center of the tumor up gradients of c, and within a few days, the tumor cell population is drastically reduced, with more "blue" cells (less immunogenic) remaining than "yellow cells" (most immunogenic). However, because the immune cell migration was strongly biased along ∇c, they pass by some tumor cells at the outer periphery. These surviving cells repopulate the tumor. This highlights the importance of stochasticity in immune cell migration; if homing is too strong, immune cells cannot mix with tumor cells, leading them to cluster in dense regions which can only interact with tumor cells on their edges. Less biased migration would increase mixing of cancer and immune cells and increase the efficacy of the immune attack. In [20, 21], we used high-throughput computing to further expand this investigation to explore the impact of stochastic migration and tumor-immune cell adhesion dynamics on the treatment efficacy.

Availability and Future Directions

Getting started as a new PhysiCell user/developer

Necessary tools Users need a working `g++` development environment with support for OpenMP and Makefiles (or a reasonably compatible 64-bit C++11 compliant compiler). (Future releases will likely include compile options using `cmake`; see Availability and Future Directions.) We provide tutorials to set up a `g++` environment in Windows and OSX at:

OSX via Homebrew:
`http://www.mathcancer.org/blog/setting-up-gcc-openmp-on-osx-homebrew-edition/`

OSX via MacPorts:
`http://www.mathcancer.org/blog/setting-up-gcc-openmp-on-osx-macports-edition/`

Windows via mingw-w64:
`http://www.mathcancer.org/blog/setting-up-a-64-bit-gcc-environment-on-windows/`

Mac OSX users should use the tutorials above; the version of `g++` included with `XCode` does not support OpenMP. Moreover, OSX users need to set a `PHYSICELL_CPP` environment variable as noted in the Homebrew and MacPorts tutorials.

Alternatively, users can code within a virtual machine using the PhysiCell virtual appliance. See the tutorial at

`http://www.mathcancer.org/blog/getting-started-with-a-physicell-virtual-applian`

Downloading PhysiCell Users can download a PhysiCell release at:

SourceForge:

https://sourceforge.net/projects/physicell/
The green "download" button will download the most recent source file (as a zip file). You can get the most recent virtual appliance by browsing to:

https://sourceforge.net/projects/physicell/files/PhysiCell/

then browsing a recent release directory (e.g., PhysiCell 1.2.2), and downloading the ova file.

GitHub:
https://github.com/MathCancer/PhysiCell/releases/latest
Download either PhysiCell_V.x.y.z.zip (source) or PhysiCell.x.y.z.ova (virtual appliance).

PhysiCell is licensed under the (3-clause) BSD license, which is compatible with commercial products and can be included in GPL-licensed projects.

Learning to use PhysiCell PhysiCell's main project website can be found at

http://PhysiCell.MathCancer.org.

Each PhysiCell download comes with a Quickstart guide (Quickstart.pdf) in the main directory. We recommend running through this guide to populate, compile, run, and visualize your first projects. (The four examples in Additional PhysiCell examples are included as sample projects, with compile instructions in the Quickstart.)

A full user manual is included in the documentation directory of every PhysiCell download. See documentation/User_Guide.pdf. This user manual includes full details on how to access and modify each cell's Phenotype, how to write custom fuctions and data elements, and how to interact with the BioFVM microenvironment [6].

Users creating their own simulation projects should start with the 2-D and 3-D template projects. User-created code should be placed in the custom_modules directory; users should not modify functions in core (core PhysiCell functionality) or modules (PhysiCell-provided, non-core functions). See the Quickstart and User Guide for further details.

We post tutorials at our PhysiCell blog, available at

http://mathcancer.org/blog/physicell-tutorials.

Check there for tips and tricks to building simulators, and visualizing results. We also frequently post updates on PhysiCell on Twitter, under the hashtag #PhysiCell.

Getting started as a new PhysiCell contributor

We welcome new functionality and bug fix contributions from the community. No special tools or libraries are required for PhysiCell development aside from a C++11/OpenMP compliant compiler, make, and a copy of the PhysiCell source. Developers should fork the current development version of the PhysiCell repository (available on GitHub at https://github.com/MathCancer/PhysiCell), develop their contributions (either in the modules directory as new functions, named as PhysiCell_[name].cpp, or as fixes to existing files), and then submit a pull request. Users should also email the PhysiCell mailing list when submitting code contributions.

Project coding conventions can be found at

`http://www.mathcancer.org/blog/mathcancer-c-style-and-practices-guide/`.

Getting help

Users encountering problems in compiling PhysiCell projects should first consult the FAQ page at:

`http://www.mathcancer.org/blog/common-problems-and-solutions/`

Users (and developers) can join the PhysiCell mailing list to ask the community questions or discuss ideas, at `physicell-users@googlegroups.com`. Users can submit bug reports at the GitHub issue tracker:

`https://github.com/MathCancer/PhysiCell/issues`

Limitations and potential solutions

Scientific limitations BioFVM does not implement advective transport as of Version 1.1.7, and so PhysiCell cannot readily be applied to advection-dominated problems without user-supplied advection solvers. We also have not yet written a model of extracellular matrix; users could add this effect by introducing custom data that (1) "anchor" each cell to a position in space beyond cell-cell mechanical operations, and (2) slowly evolve these "anchor points" to model ECM rearrangement. We are currently exploring this approach to model liver parenchyma, and we will post sample codes as the work progresses. Alternatively, users could introduce an independent finite element mesh for the ECM, evolve it under (e.g., viscoelastic) laws, and attach cells to the nearest lattice site. As a simpler approach, users could include a non-diffusing ECM substrate (in BioFVM), which could be used to reduce the cells' velocities (as extra drag). Vary the cell-cell mechanics parameters in `phenotype.mechanics` with the ECM density for this approach.

PhysiCell's cell-centered approach may not be ideal for application to some morphogenesis problems; see the recent work by Osborne and co-workers that compared several discrete modeling frameworks in morphogenesis test problems [47]. PhysiCell does not model cell morphology, and so it cannot directly model cell elongation processes needed in some morphogenesis problems. However, users could potentially model elongated cells as two or more agents, and introduce manual adhesive links similar to our "biorobots" examples. PhysiCell has not implemented cell fusion (another mechanism of morphogenesis), but users could manually implement this by (1) testing for cell-cell contact, (2) moving one cell to the center of volume of the two cells, (3) updating that cell's volume to include both original cells' volumes, and (4) deleting the second cell. PhysiCell has not yet written functions to deal with polarized cell-cell adhesion, or to update its orientation. Thus, PhysiCell may need further development and user contributions to model the full spectrum of morphogenetic mechanisms.

We note that PhysiCell can manually implement molecular-scale biology as ODEs (e.g., as in the drug damage example) via custom data, but this is currently difficult to scale to large systems of ODEs with many parameters. Moreover, while we have provided examples of user-defined cell cycle models and other custom functions in the User Guide, sample codes, and tutorials, we do not provide templates for such functions beyond the documentation. We are currently testing methods to support SBML specification of ODE models in the cell agents (e.g., via libRoadrunner [48]), but until SBML is supported,

we do not provide templates to integrate ODE models into PhysiCell functions to alter cell phenotype.

PhysiCell does not currently provide direct methods to model contact-based cell-cell interactions; these can be manually implemented as we did in the immune and biorobots examples. We also do not yet provide alternative mechanics models (e.g., viscoelastic) out-of-the-box, but users can design their own cell velocity functions as needed to replace the default mechanics. We have not yet implemented direct calculations of cell pressures and strains, so this could hinder some applications in mechanobiology. Users could implement this by creating a custom velocity update function that calculates pressures and strains from the potential functions. A preliminary version—calculated from the "resistance" potential functions similarly to [49]—is included as `state.basic_pressure`. See the User Guide for more details.

Software limitations PhysiCell is intended to function as a modular engine and to interact with standardized data (e.g., MultiCellDS [22]), but it has not yet fully implemented MultiCellDS import and export. Graphical simulation design tools, data visualization tools, and analysis tools are needed to widen its accessibility beyond seasoned C++ programmers and reduce its learning curve. In the future, PhysiCell should switch from Makefiles to CMake to facilitate simpler cross-platform compiling.

Future improvements

Numerical improvements The biggest code bottleneck is cell-cell interaction testing: cell volume can vary by a factor of 100, and hence the cell diameter (and interaction distance) can vary by a factor of 50. The number of cells in the list of interacting neighbors $\mathcal{N}(i)$ scales inversely with the minimum cell volume; see the supplementary material. Future versions of PhysiCell will introduce a nested mesh interaction testing structure to more accurately estimate $\mathcal{N}(i)$ in regions with small cells. Although PhysiCell's design objectives are not focused on larger simulations with 10^8-10^9 cells, extension to supercomputers by wrapping the code in MPI with appropriate data mirroring could be useful. Likewise, the algorithms used in BioFVM could be implemented on a graphics card via OpenCL or OpenACC. We will explore these approaches in the future.

Scientific improvements We will develop SBML (systems biology markup language) importers for molecular-scale biology, most likely by the C code generation features in COPASI [50] or libRoadrunner [48]. This will make it straightforward to integrate systems of ODEs to individual cells. We will add new cycle models for flow cytometry-driven problems (G1, S, and G2/M phases). We plan to add more advanced cell mechanics models (e.g., as in [9, 51, 52]), and to extend PhysiCell's included standard functions to include extracellular matrix mechanics. We also plan to introduce built-in functions for polarized cell adhesion and updating the cells' orientations. As our PhysiCell-based projects progress, we will "upstream" our new functions to PhysiCell as new optional modules. We are currently testing and refining the elastic-based cell adhesions in Additional PhysiCell examples for inclusion as standard models, and we are also developing vascularization libraries as part of our work to simulate metastatic breast cancers injected in mammary fat pads.

User-focused improvements In the coming months, we will continue publishing blog posts and code samples at `http://MathCancer.org/blog/physicell-tutorials/`. We will create pre-compiled clients that can initiate simulations based upon a digital snapshot (intitial arrangement of cells) and digital cell lines (self-contained, model-independent sets of cell phenotype data), using the emerging MultiCellDS standard [22, 23]. We will

develop support to read parameters from XML configuration files, and we plan to offer PhysiCell as a compiled, linkable library. Moreover, we plan to develop user-friendly, web-based applications of PhysiCell to simulate the multicellular response to diffusing engineered nanoparticles [53].

Supporting Information

Video S1

3-D simulation of 18 days of hanging drop tumor spheroid growth from 2300 cells to 1.2 million cells, using the deterministic necrosis model. Available at:
`https://www.youtube.com/watch?v=WMhYW9D4SqM`

Video S2

3-D simulation of 18 days of hanging drop tumor spheroid growth from 2300 cells to 1 million cells, using the stochastic necrosis model. Available at:
`https://www.youtube.com/watch?v=xrOqqJ_Exd4`

Video S3

3-D simulation video of 30 days of DCIS growth in a 1 mm length of breast duct, using the deterministic necrosis model. Available at:
`https://www.youtube.com/watch?v=ntVKOr9poro`

Video S4

3-D simulation video of 30 days of DCIS growth in a 1 mm length of breast duct, using the stochastic necrosis model. Available at:
`https://www.youtube.com/watch?v=-lRot-dfwJk`

Video S5

2-D simulation of the "biorobots" example, showing a synthetic multicellular cargo delivery system. Available at: `https://www.youtube.com/watch?v=NdjvXI_x8uE`

Video S6

2-D simulations of the "biorobots" adapted for use as a cancer treatment, where cargo cells detach and secrete a therapeutic once reaching hypoxic tissues. Available at: `https://www.youtube.com/watch?v=wuDZ4OjW__M`

Video S7

2-D simulation of a tumor whose heterogeneous oncoprotein expression drives proliferation and selection. Available at: `https://www.youtube.com/watch?v=bPDw6l4zkF0`

Video S8

3-D simulation of immune cells attacking a tumor with heterogeneous proliferation and immunogenicity. Available at: `https://www.youtube.com/watch?v=nJ2urSm4ilU`

User manual

User Manual for PhysiCell 1.2.2.

Quickstart guide

A fast guide to building, running, and visualizing your first PhysiCell simulations. We recommend that new users start with this guide.

Supplementary materials

Extensive supplementary materials include: full mathematical model details, supporting literature, and reference parameter values for breast epithelial cells; expanded numerical implementation details; convergence and validation testing results; full parameter values for the main tests; and an expanded feature comparison of PhysiCell and other 3-D multicellular simulation platforms.

Acknowledgments

We thank the Breast Cancer Research Foundation and the Jayne Koskinas and Ted Giovanis Foundation for Health and Policy for generous support. Partial support was provided by the USC James H. Zumberge Research and Innovation Fund (2012 Large Interdisciplinary Award), the National Institutes of Health (5U54CA143907,1R01CA180149) , and the National Science Foundation (1720625). We thank David B. Agus and the USC Center for Applied Molecular Medicine for generous institutional support. We thank Margy Villa for her assistance in early testing, and Edwin Juarez and Rishi Rawat for useful discussions. We greatly appreciate the deep discussions with the MultiCellDS review panel in developing standardized cell cycle and death models. We thank Jakob Kather and Jan Poleszczuk for discussions on cancer immunology. We thank Nathan Choi for his 3-D hanging drop spheroid work in Figure 2.

References

1. Macklin P. Biological background. In: V. Cristini and J.S. Lowengrub, Multiscale Modeling of Cancer: An Integrated Experimental and Mathematical Modeling Approach. Cambridge, UK: Cambridge University Press; 2010. p. 8–23. (invited author: P. Macklin). Available from: `http://dx.doi.org/10.1017/CBO9780511781452.003`.

2. Lowengrub J, Frieboes HB, Jin F, Chuang YL, Li X, Macklin P, et al. Nonlinear modeling of cancer: Bridging the gap between cells and tumors. Nonlinearity. 2010;23(1):R1–R91. (invited author: J. Lowengrub). Available from: `http://dx.doi.org/10.1088/0951-7715/23/1/R01`.

3. Deisboeck TS, Wang Z, Macklin P, Cristini V. Multiscale Cancer Modeling. Annu Rev Biomed Eng. 2011;13:127–155. (invited author: T.S. Deisboeck). Available from: `http://dx.doi.org/10.1146/ANNUREV-BIOENG-071910-124729`.

4. Hatzikirou H, Chauviere A, Bauer AL, Leier A, Lewis MT, Macklin P, et al. Integrative physical oncology. WIREs Syst Biol Med. 2012;4(1):1–14. (invited author: V. Cristini). Available from: `http://dx.doi.org/10.1002/wsbm.158`.

5. Macklin P, Frieboes HB, Sparks JL, Ghaffarizadeh A, Friedman SH, Juarez EF, et al. Progress Towards Computational 3-D Multicellular Systems Biology. In: Rejniak KA, editor. Systems Biology of Tumor Microenvironment, Advances in Experimental Medicine and Biology. vol. 936. Bern, Switzerland: Springer; 2016 (in press). (invited author: P. Macklin). Available from: http://dx.doi.org/10.1007/978-3-319-42023-3_12.

6. Ghaffarizadeh A, Friedman SH, Macklin P. BioFVM: an efficient, parallelized diffusive transport solver for 3-D biological simulations. Bioinformatics. 2016;32(8):1256–1258. Available from: http://dx.doi.org/10.1093/bioinformatics/btv730.

7. Swat MH, Thomas GL, Belmonte JM, Shirinifard A, Hmeljak D, Glazier JA. Chapter 13 - Multi-Scale Modeling of Tissues Using CompuCell3D. In: Asthagiri AR, Arkin AP, editors. Computational Methods in Cell Biology. vol. 110 of Methods in Cell Biology. Academic Press; 2012. p. 325 – 366. Available from: http://dx.doi.org/10.1016/B978-0-12-388403-9.00013-8.

8. Starruß J, de Back W, Brusch L, Deutsch A. Morpheus: a user-friendly modeling environment for multiscale and multicellular systems biology. Bioinformatics. 2014;30(9):1331–1332. Available from: http://dx.doi.org/10.1093/bioinformatics/btt772.

9. Hoehme S, Drasdo D. A cell-based simulation software for multi-cellular systems. Bioinformatics. 2010;26(20):2641–2642. Available from: http://dx.doi.org/10.1093/bioinformatics/btq437.

10. Mirams GR, Arthurs CJ, Bernabeu MO, Bordas R, Cooper J, Corrias A, et al. Chaste: An Open Source C++ Library for Computational Physiology and Biology. PLoS Comput Biol. 2013 03;9(3):1–8. Available from: http://dx.doi.org/10.1371/journal.pcbi.1002970.

11. Kang S, Kahan S, McDermott J, Flann N, Shmulevich I. Biocellion: accelerating computer simulation of multicellular biological system models. Bioinformatics. 2014;30(21):3101–3108. Available from: http://dx.doi.org/10.1093/bioinformatics/btu498.

12. Cytowski M, Szymanska Z. Large-Scale Parallel Simulations of 3D Cell Colony Dynamics. Computing in Science Engineering. 2014 Sept;16(5):86–95. Available from: http://dx.doi.org/10.1109/MCSE.2014.2.

13. Cytowski M, Szymanska Z. Large-Scale Parallel Simulations of 3D Cell Colony Dynamics: The Cellular Environment. Computing in Science Engineering. 2015 Sept;17(5):44–48. Available from: http://dx.doi.org/10.1109/MCSE.2015.66.

14. Macklin P, Edgerton ME, Thompson AM, Cristini V. Patient-calibrated agent-based modelling of ductal carcinoma in situ (DCIS): From microscopic measurements to macroscopic predictions of clinical progression. J Theor Biol. 2012;301:122–40. Available from: http://dx.doi.org/10.1016/j.jtbi.2012.02.002.

15. Hyun AZ, Macklin P. Improved patient-specific calibration for agent-based cancer modeling. J Theor Biol. 2013;317:422–4. Available from: http://dx.doi.org/10.1016/j.jtbi.2012.10.017.

16. Mumenthaler SM, D'Antonio G, Preziosi L, Macklin P. The need for integrative computational oncology: An illustrated example through MMP-mediated tissue degradation. Front Oncol. 2013;3:194. Available from: http://dx.doi.org/10.3389/fonc.2013.00194.

17. Grogan JA, Connor AJ, Markelc B, Muschel RJ, Maini PK, Byrne HM, et al. Microvessel Chaste: An Open Library for Spatial Modeling of Vascularized Tissues. Biophys J. 2017;112(9):1767–72. Available from: http://dx.doi.org/10.1016/j.bpj.2017.03.036.

18. Schumacher LJ, Maini PK, Baker RE. Semblance of Heterogeneity in Collective Cell Migration. Cell Sys. 2017;5(2):119–27. Available from: https://dx.doi.org/10.1016/j.cels.2017.06.006.

19. Macklin P. When Seeing Isn't Believing: How Math Can Guide Our Interpretation of Measurements and Experiments. Cell Sys. 2017;5(2):92–4. Available from: http://dx.doi.org/10.1016/j.cels.2017.08.005.

20. Ozik J, Collier N, Wozniak J, Macal C, Cockrell C, Friedman S, et al. High-throughput cancer hypothesis testing with an integrated PhysiCell-EMEWS workflow. bioRxiv, accepted SuperComputing SC17 Computational Approaches for Cancer Workshop. 2017;p. 196709. Available from: https://dx.doi.org/10.1101/196709.

21. Ozik J, Collier N, Wozniak J, Macal C, Cockrell C, Friedman S, et al. High-throughput cancer hypothesis testing with an integrated PhysiCell-EMEWS workflow [Presented at SuperComputing SC17, Computational Approaches for Cancer Workshop]. FigShare. 2017;p. 5687197.v1. Available from: https://dx.doi.org/10.6084/m9.figshare.5687197.v1.

22. Friedman SH, Anderson ARA, Bortz DM, Fletcher AG, Frieboes HB, Ghaffarizadeh A, et al. MultiCellDS: a community-developed standard for curating microenvironment-dependent multicellular data. bioRxiv. 2016;Available from: https://dx.doi.org/10.1101/090456.

23. Macklin P, et al.. MultiCellDS Project Website; 2014-2016. Available from: https://MultiCellDS.org.

24. Benson J, Macklin P. 43. Cell-based modeling of mechanical and chemical stress in tissues during cryoprotocols. Cryobiology. 2015;71(1):176. Available from: http://dx.doi.org/10.1016/j.cryobiol.2015.05.049.

25. Kapoulkine A. pugixml: Light-weight, simple and fast XML parser for C++ with XPath support; 2015. Available from: https://github.com/zeux/pugixml.

26. Simpson MJ, Towne C, McElwain DLS, Upton Z. Migration of breast cancer cells: Understanding the roles of volume exclusion and cell-to-cell adhesion. Phys Rev E. 2010 Oct;82:041901. Available from: http://link.aps.org/doi/10.1103/PhysRevE.82.041901.

27. Drasdo D, Kree R, McCaskill JS. Monte Carlo approach to tissue-cell populations. Phys Rev E. 1995 Dec;52:6635–6657. Available from: http://dx.doi.org/10.1103/PhysRevE.52.6635.

28. Galle J, Loeffler M, Drasdo D. Modeling the Effect of Deregulated Proliferation and Apoptosis on the Growth Dynamics of Epithelial Cell Populations In Vitro. Biophys J. 2005;88(1):62–75. Available from: http://dx.doi.org/10.1529/biophysj.104.041459.

29. Macklin P, Lowengrub JS. Evolving interfaces via gradients of geometry-dependent interior Poisson problems: application to tumor growth. J Comput Phys. 2005;203(1):191–220. Available from: http://dx.doi.org/10.1016/j.jcp.2004.08.010.

30. Macklin P, Lowengrub JS. Nonlinear simulation of the effect of microenvironment on tumor growth. J Theor Biol. 2007;245(4):677–704. Available from: http://dx.doi.org/10.1016/j.jtbi.2006.12.004.

31. Macklin P, Lowengrub JS. A New Ghost Cell/Level Set Method for Moving Boundary Problems: Application to Tumor Growth. J Sci Comput. 2008;35(2-3):266–299. (invited author: J.S. Lowengrub). Available from: http://dx.doi.org/10.1007/s10915-008-9190-z.

32. Edgerton ME, Chuang YL, Macklin P, Yang W, Bearer EL, Cristini V. A novel, patient-specific mathematical pathology approach for assessment of surgical volume: Application to ductal carcinoma in situ of the breast. Anal Cell Pathol. 2011;34(5):247–63. Available from: http://dx.doi.org/10.3233/ACP-2011-0019.

33. Macklin P, Mumenthaler S, Lowengrub J. Modeling multiscale necrotic and calcified tissue biomechanics in cancer patients: application to ductal carcinoma in situ (DCIS). In: Gefen A, editor. Multiscale Computer Modeling in Biomechanics and Biomedical Engineering. Berlin, Germany: Springer; 2013. p. 349–80. (invited author: P. Macklin). Available from: http://dx.doi.org/10.1007/8415_2012_150.

34. Majno G, Joris I. Apoptosis, oncosis, and necrosis. An overview of cell death. The American journal of pathology. 1995;146(1):3. Available from: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC1870771/.

35. Trump BE, Berezesky IK, Chang SH, Phelps PC. The Pathways of Cell Death: Oncosis, Apoptosis, and Necrosis. Toxicologic Pathology. 1997;25(1):82–88. Available from: http://tpx.sagepub.com/content/25/1/82.abstract.

36. Garvey CM, Spiller E, Lindsay D, Chiang CT, Choi NC, Agus DB, et al. A high-content image-based method for quantitatively studying context-dependent cell population dynamics. Sci Rep. 2016;6:29752. Available from: http://dx.doi.org/10.1038/srep29752.

37. McKeown SR. Defining normoxia, physoxia and hypoxia in tumours—implications for treatment response. The British Journal of Radiology. 2014;87(1035):20130676. PMID: 24588669. Available from: http://dx.doi.org/10.1259/bjr.20130676.

38. Ghosh S, Joshi MB, Ivanov D, Feder-Mengus C, Spagnoli GC, Martin I, et al. Use of multicellular tumor spheroids to dissect endothelial cell–tumor cell interactions: A role for T-cadherin in tumor angiogenesis. FEBS Letters. 2007;581(23):4523–4528. Available from: http://dx.doi.org/10.1016/j.febslet.2007.08.038.

39. Ma Hl, Jiang Q, Han S, Wu Y, Tomshine JC, Wang D, et al. Multicellular Tumor Spheroids as an in Vivo–Like Tumor Model for Three-Dimensional Imaging of Chemotherapeutic and Nano Material Cellular Penetration. Molecular Imaging. 2012;11(6). Available from: http://mix.sagepub.com/content/11/6/7290.2012.00012.abstract.

40. Kwon IK, Lee SC, Han B, Park K. Analysis on the current status of targeted drug delivery to tumors. Journal of Controlled Release. 2012;164(2):108 – 114. Drug Delivery and Cancer: Today's Challenges, Tomorrow's Directions. Available from: http://www.sciencedirect.com/science/article/pii/S0168365912005500.

41. Cukierman E, Khan DR. The benefits and challenges associated with the use of drug delivery systems in cancer therapy. Biochemical Pharmacology. 2010;80(5):762 – 770. Targeted Cancer Therapy. Available from: http://www.sciencedirect.com/science/article/pii/S0006295210002947.

42. Eales KL, Hollinshead KER, Tennant DA. Hypoxia and metabolic adaptation of cancer cells. Oncogenesis. 2016;5:e190. Available from: http://dx.doi.org/10.1038/oncsis.2015.50'.

43. Wilson WR, Hay MP. Targeting hypoxia in cancer therapy. Nat Rev Canc. 2011;11(6):393–410. Available from: http://dx.doi.org/10.1038/nrc3064.

44. Ichim CV. Revisiting immunosurveillance and immunostimulation: Implications for cancer immunotherapy. Journal of Translational Medicine. 2005 Feb;3(1):8. Available from: https://doi.org/10.1186/1479-5876-3-8.

45. Khanna R. Tumour surveillance: Missing peptides and MHC molecules. Immunol Cell Biol. 1998;76(1):20–6. Available from: http://dx.doi.org/10.1046/j.1440-1711.1998.00717.x.

46. Comber JD, Philip R. MHC class I antigen presentation and implications for developing a new generation of therapeutic vaccines. Therapeutic Advances in Vaccines. 2014;2(3):77–89. Available from: http://dx.doi.org/10.1177/2051013614525375.

47. Osborne JM, Flecther AG, Pitt-Francis JM, Maini PK, Gavaghan DJ. Comparing individual-based approaches to modelling the self-organization of multicellular tissues. PLoS Comput Biol. 2017;13(2):e1005387. Available from: http://dx.doi.org/10.1371/journal.pcbi.1005387.

48. Somogyi ET, Bouteiller JM, Glazier JA, K´onig M, Medley JK, Swat MH, et al. libRoadRunner: a high performance SBML simulation and analysis library. Bioinformatics. 2015;31(20):3315–21. Available from: http://dx.doi.org/10.1093/bioinformatics/btv363.

49. Byrne H, Drasdo D. Individual-based and continuum models of growing cell populations: a comparison. J Math Biol. 2009;58(4–5):657–87. Available from: http://dx.doi.org/10.1007/s00285-008-0212-0.

50. Hoops S, Sahle S, Gauges R, Lee C, Pahle J, Simus N, et al. COPASI—a COmplex PAthway SImulator. Bioinformatics. 2006;22(24):3067–3074. Available from: http://dx.doi.org/10.1093/bioinformatics/btl485.

51. Chu YS, Dufour S, Thiery JP, Perez E, Pincet F. Johnson-Kendall-Roberts Theory Applied to Living Cells. Phys Rev Lett. 2005 Jan;94:028102. Available from: https://link.aps.org/doi/10.1103/PhysRevLett.94.028102.

52. Buske P, Galle J, Barker N, Aust G, Clevers H, Loeffler M. A Comprehensive Model of the Spatio-Temporal Stem Cell and Tissue Organisation in the Intestinal Crypt. PLoS Comput Biol. 2011 01;7(1):1–13. Available from: http://dx.doi.org/10.1371/journal.pcbi.1001045.

53. National Science Foundation. Award Abstract 1720625: Network for Computational Nanotechnology-Engineered nanoBIO Node (PI: Fox; Co-PIs: Glazier, Douglas, Jadhao and Macklin); 2017–22. Available from: https://www.nsf.gov/awardsearch/showAward?AWD_ID=1720625.

Appendix: computational cost estimates (expanded)

In the main text, we discussed that most of the simulation steps have computational cost that scales linearly in the number of cells. The step that requires additional analysis (and relies upon PhysiCell's IDS) is the step where cell-cell mechanical interactions are used to set the cell velocities. Bounding this computational costs requires that we find a fixed upper bound on the number of cell-cell interactions, so that the computational cost is $\mathcal{O}(1)$ for single cells, and $\mathcal{O}(n)$ for all the cells.

We estimate an upper bound on the of cells in any voxel B_i by

$$N_{\max} \leq \frac{V_{\text{mech}}}{\min\{V_i : \mathbf{x}_i \in B_i\}},, \tag{25}$$

where V_{mech} is the fixed volume of the voxels in the interaction testing data structure. For cycling cells with "mature" volume V, we have $V_i \geq \frac{1}{2}V$. By default, dead cells are removed when $V_i \leq 20\ \mu\text{m}^3 (\approx \frac{1}{100}\,V)$. Since a typical $\mathcal{N}(i)$ is constructed from up to 27 such voxels, we have

$$N_{\max} \leq 27 \frac{V_{\text{mech}}}{\frac{1}{2}V} = 54 \frac{V_{\text{mech}}}{V} \tag{26}$$

for simulations dominated by live cells, and

$$N_{\max} \leq 27 \frac{V_{\text{mech}}}{\frac{1}{100}V} = 2700 \frac{V_{\text{mech}}}{V} \tag{27}$$

for simulations dominated by dead cells. Thus, the computational cost for a single cell's mechanical interactions is bounded by a fixed constant, and the total cost over all cell-cell mechanical interactions scales linearly in n. The slope of the cost-versus-n curve may be shallower for early, non-necrotic simulations, and it can be up to a factor of 100 steeper for necrosis-dominated simulations. In some cases, simulations may temporarily show a nonlinear relationship with n when transitioning from non-necrotic to necrotic.

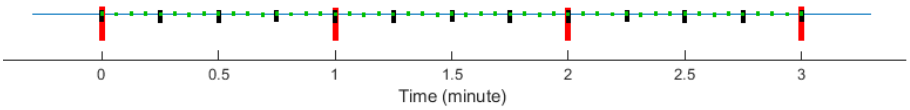

Figure 1. PhysiCell and multiple time scales: PhysiCell uses BioFVM to update the microenvironment at the short green tick marks, corresponding to Δt_{diff}. It updates cell mechanics (including cell position) less frequently at the medium black tick marks (Δt_{mech}), and it runs the cell volume and cycle/death models least frequently at the long red tick marks (Δt_{cell}). Note that the time steps shown are for illustrative purpose; the default step sizes are given in the supplementary materials.

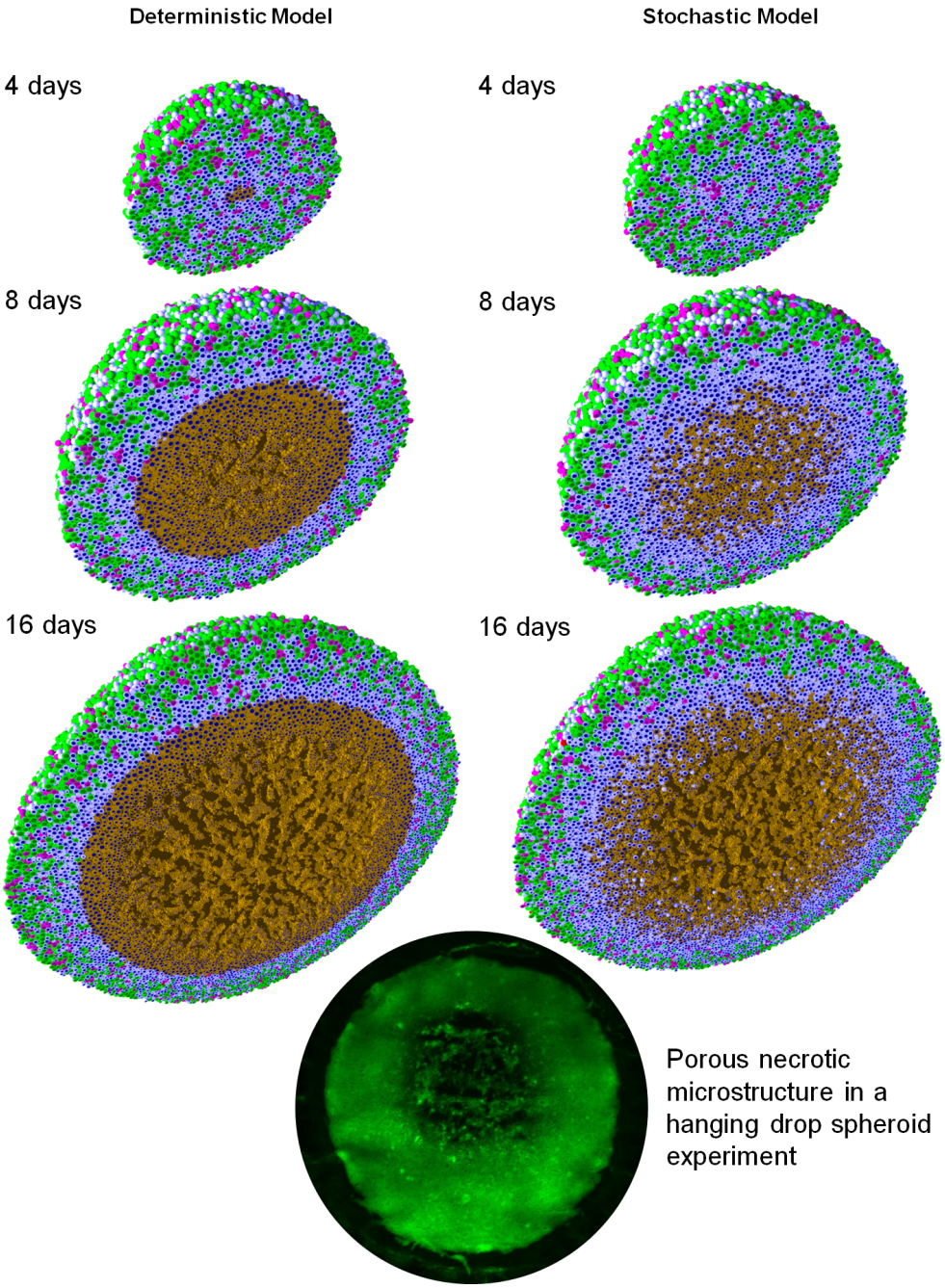

Figure 2. Hanging drop spheroid (HDS) simulations with deterministic necrosis (left) and stochastic necrosis (right), plotted at 4, 8, and 16 days. Videos are available at Video S1 and Video S2. ***Legend:*** Ki67+ cells are green before mitosis (K_1) and magenta afterwards (K_2). Pale blue cells are Ki67- (Q), dead cells are red (apoptotic) and brown (necrotic), and nuclei are dark blue. **Bottom:** Hanging drop spheroid experiment (HCC827 non-small cell lung carcinoma) showing a similar necrotic core microstructure. PhysiCell is the first simulation to predict this structure arising from cell-scale mechanical interactions. Image courtesy Mumenthaler lab, Lawrence J. Ellison Center for Transformative Medicine, University of Southern California.

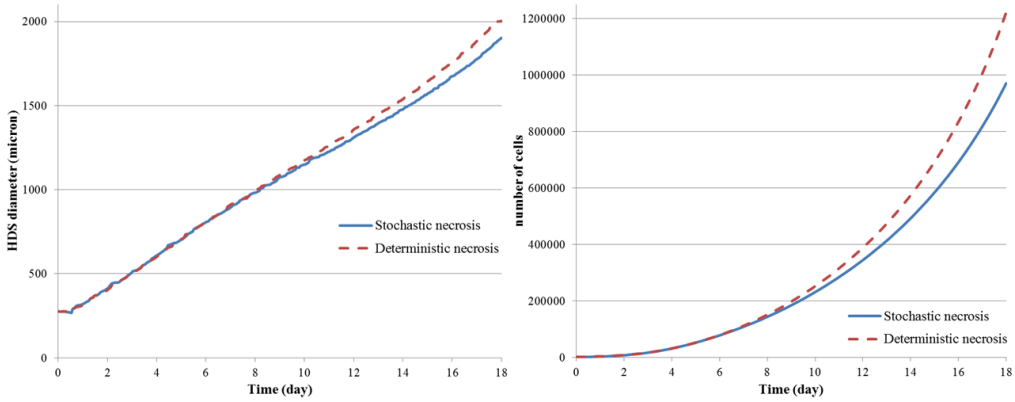

Figure 3. HDS growth: ***Left:*** The deterministic and stochastic necrosis models both give approximately linear growth (left), but the HDS with deterministic necrosis model grows faster ($\sim 5\%$ difference in diameter at day 18). ***Right:*** The HDS with stochastic necrosis has fewer cells than the deterministic model ($\sim 26\%$ difference in cell count at day 18), due to its delay in necrosis. The difference in cell count is larger than the difference in tumor diameter because most of the difference lies in the number of necrotic cells, and necrotic cells are smaller than viable cells.

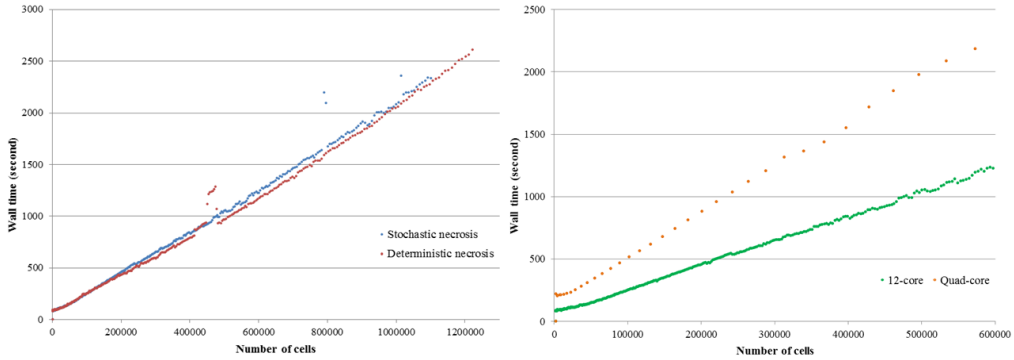

Figure 4. HDS computational cost scaling: ***Left:*** Wall-time vs. cell count for the stochastic (red) and deterministic (blue) necrosis necrosis models on a single HPC compute node. Both models show approximately linear cost scaling with the number of cell agents. ***right:*** Wall time vs. cell count for stochastic necrosis model on the desktop workstation (orange) and the single HPC node (green).

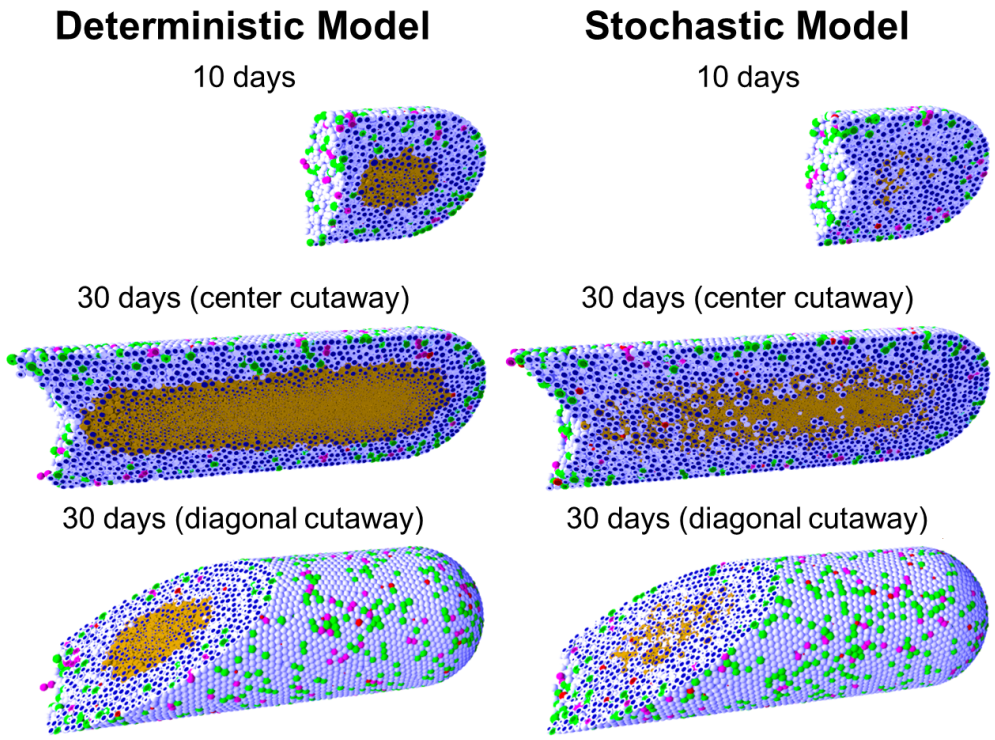

Figure 5. Ductal carcinoma in situ (DCIS) simulations with deterministic necrosis (left) and stochastic necrosis (right), plotted at 10 and 30 days (multiple views). Videos are available at Video S3 and Video S4. The figure legend is the same as Fig.2.

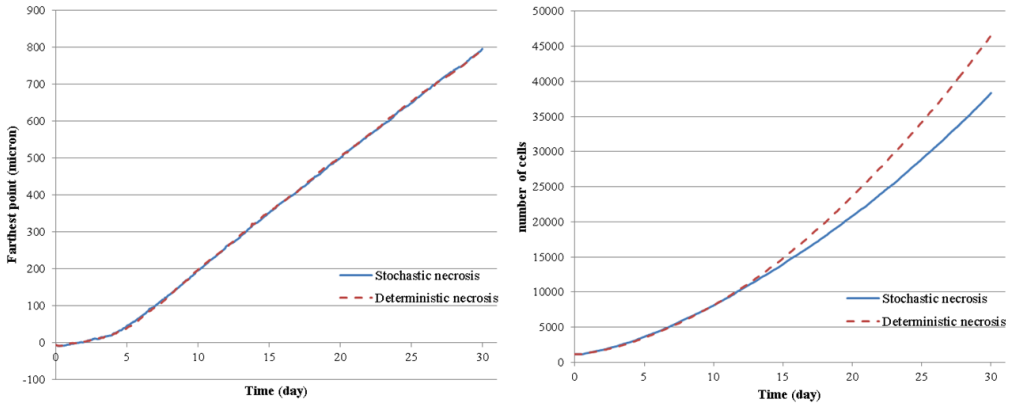

Figure 6. DCIS growth: The deterministic and stochastic necrosis models both result in linear DCIS growth at approximately 1 cm/year (left), even while their cell counts differ by 21% by the end of the simulations (right).

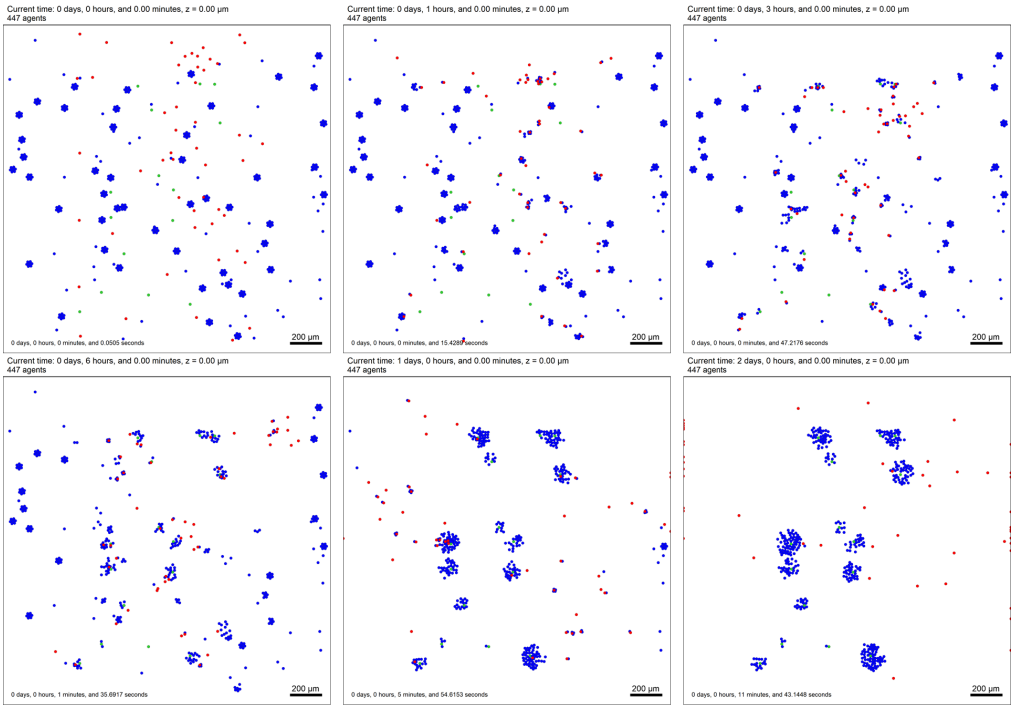

Figure 7. "Biorobots" example. Director cells (green) release a chemoattractant c_1 to guide worker cells (red). Cargo cells (blue) release a separate chemoattractant c_2. Unadhered worker cells chemotax towards ∇c_2, test for contact with cargo cells, form adhesive bonds, and then pull them towards the directors by following ∇c_1. If c_1 exceeds a threshold, the worker cells release the cargo and return to seek more cargo cells, repeating the cycle. A video is available at Video S5.

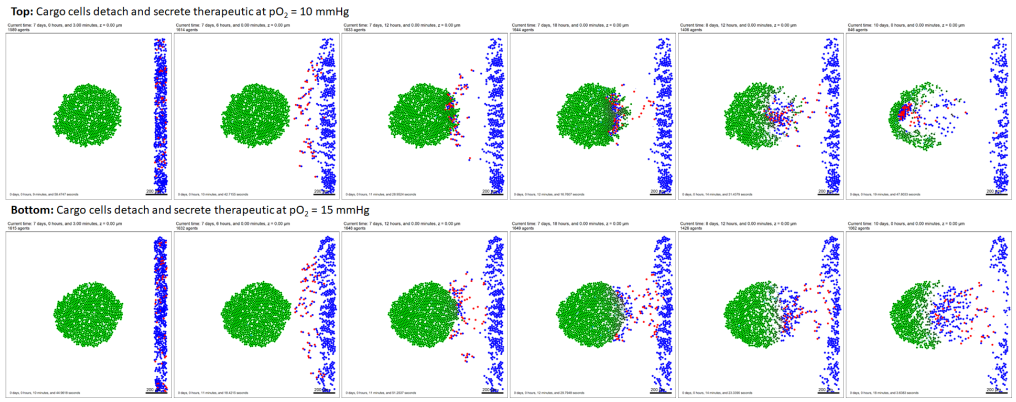

Figure 8. Anti-cancer "biorobots" example. By modifying the worker cells in the previous example (Fig. 7) to move up hypoxic gradients (along $-\nabla pO_2$) and drop their cargo in hypoxic zones, we can deliver cargo to a growing tumor. In this example, the cargo cells secrete a therapeutic that induces apoptosis in nearby tumor cells, leading to partial tumor regression. A video is available at Video S6.

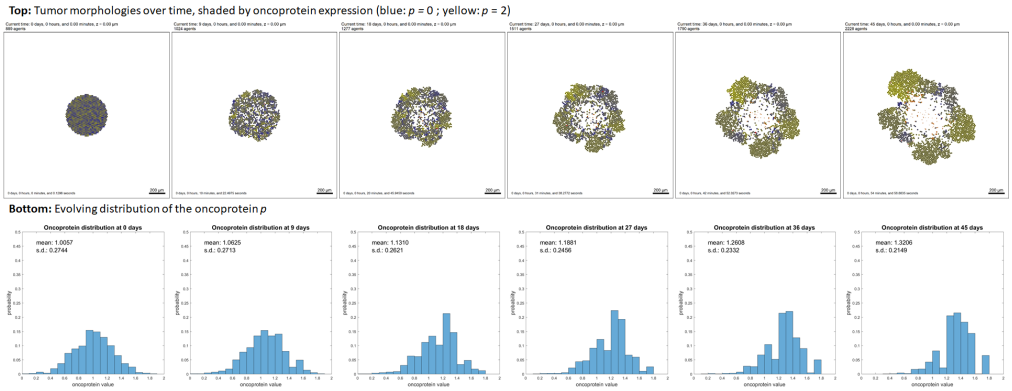

Figure 9. Cancer heterogeneity example. Each cell has an independent expression of a mutant "oncoprotein" p (dimensionless, bounded in [0,2]), which scales the oxygen-based rate of cell cycle entry. Blue cells have least p, and yellow cells have most. Initially, the population has normally distributed p with mean 1, standard deviation 0.3, and a "salt and pepper" mixed spatial arrangement. The proliferative advantage for cells with higher p leads to selection and enrichment of the most yellow cells. Stochastic effects lead to emergence of fast-growing foci and a loss of tumor symmetry. A video is available at Video S7.

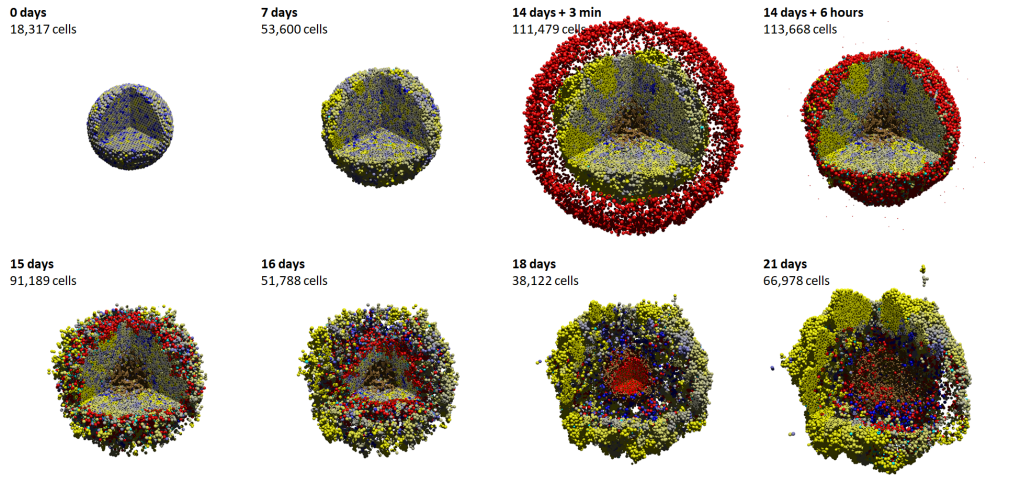

Figure 10. Cancer immunology example. In this 3-D example, each tumor cell secretes an immunostimulatory factor, and its immunogenicity is modeled as proportional to its mutant oncoprotein expression. (See the previous example in Fig. 9.) After 14 days, red immune cells perform a biased random walk towards the immunostimulatory factor, test for contact with cells, form adhesions, and attempt to induce apoptosis for cells with greater immunogenicity. The immune cells successfully attack the tumor initially, leading to partial regression; apoptotic cells are cyan. But strong homing towards gradients of the immunostimulatory factor causes immune cells to "pass" some cells at the outer edge, leading to tumor regrowth. Eventually, immune cells leave the necrotic regions and press their attack on the tumor. This highlights the importance of stochasticity in immune cell movement in mixing with the tumor cells for a more successful immune response. A video is available at Video S8.