BioFVM: an efficient, parallelized diffusive transport solver for 3-D biological simulations

Appendix: Computational details, example, convergence, and performance testing

Ahmadreza Ghaffarizadeh, Samuel H. Friedman, Paul Macklin*

December 3, 2015

Abstract

These are the supplementary materials for:

> A. Ghaffarizadeh, S.H. Friedman, and P. Macklin, **BioFVM: an efficient parallelized diffusive transport solver for 3-D biological simulations,** *Bioinformatics.* (2015, in review)

Any corrections will be posted online at BioFVM.MathCancer.org

Motivation: Computational models of multicellular systems require solving systems of PDEs for release, uptake, decay, and diffusion of multiple substrates in 3D, particularly when incorporating the impact of drugs, growth substrates, and signaling factors on cell receptors and subcellular systems biology.
Results: We introduce BioFVM, a diffusive transport solver tailored to biological problems. BioFVM can simulate release and uptake of many substrates by cell and bulk sources, diffusion, and decay in large 3D domains. It has been parallelized with OpenMP, allowing efficient simulations on desktop workstations or single supercomputer nodes. The code is stable even for large time steps, with linear computational cost scalings. Solutions are first-order accurate in time and second-order accurate in space. The code can be run by itself or as part of a larger simulator.
Availability: BioFVM is written in C++ with parallelization in OpenMP. It is maintained and available for download at http://BioFVM.MathCancer.org and http://BioFVM.sf.net under the Apache License (v2.0).
Contact: paul.macklin@usc.edu
Supplementary information: The full algorithm and convergence/performance testing are provided in supplementary materials and at BioFVM.MathCancer.org. A tutorial is included in each BioFVM download.

Contents

1 Introduction – Equations solved by BioFVM

In all the following sections, we solve the following system of partial differential equations on a computational domain Ω (with boundary $\partial\Omega$) for a vector of densities $\boldsymbol{\rho}$

$$\frac{\partial \boldsymbol{\rho}}{\partial t} = \overbrace{\nabla \cdot (\mathbf{D} \circ \nabla \boldsymbol{\rho})}^{\text{diffusion}} - \overbrace{\boldsymbol{\lambda} \circ \boldsymbol{\rho}}^{\text{decay}} + \overbrace{\mathbf{f}}^{\text{net source}} \quad \text{in } \Omega \tag{1}$$

$$(\mathbf{D} \circ \nabla \boldsymbol{\rho}) \cdot \mathbf{n} = \mathbf{0} \text{ on } \partial\Omega \tag{2}$$

$$\boldsymbol{\rho}(\mathbf{x}, t_0) = \mathbf{g} \text{ in } \Omega. \tag{3}$$

Above, $\mathbf{a} \circ \mathbf{b}$ denotes the Hadamard (termwise) product ($[\mathbf{a} \circ \mathbf{b}]_i = a_i b_i$ for each i). Similarly, we use $\mathbf{a} /\!/ \mathbf{b}$ to denote termwise division ($[\mathbf{a} /\!/ \mathbf{b}]_i = a_i / b_i$ for each i); $\mathbf{1}$ and $\mathbf{0}$ are vectors of all one and all zero, respectively, with same dimension as $\boldsymbol{\rho}$. Also,

$$\begin{aligned} \nabla \cdot (\mathbf{a} \circ \nabla \mathbf{b}) &= \nabla \cdot [a_1 \nabla b_1, \ldots, a_n \nabla b_n] = [\nabla \cdot (a_1 \nabla b_1), \nabla \cdot (a_2 \nabla b_2), \ldots, \nabla \cdot (a_n \nabla b_n)] \\ \nabla \mathbf{a} \cdot \mathbf{n} &= [\nabla a_1, \nabla a_2, \ldots, \nabla a_n] \cdot \mathbf{n} = [\nabla a_1 \cdot \mathbf{n}, \ldots, \nabla a_n \cdot \mathbf{n}] \end{aligned}$$

We shall use net sources of the following form:

$$\mathbf{f} = \mathbf{f}_{\text{bulk}}(\mathbf{x}, t) + \mathbf{f}_{\text{cells}}(\mathbf{x}, t). \tag{4}$$

We take the following form for net bulk sources (including reactions):

$$\mathbf{f}_{\text{bulk}} = \overbrace{\mathbf{S}(\mathbf{x}, t) \circ (\boldsymbol{\rho}^* - \boldsymbol{\rho})}^{\text{supply term}} - \overbrace{\mathbf{U}(\mathbf{x}, t) \circ \boldsymbol{\rho}}^{\text{uptake term}} + \overbrace{\mathbf{R}(\boldsymbol{\rho}, \mathbf{x}, t)}^{\text{remainder}}, \tag{5}$$

where $\mathbf{S}$ is the vector of supply rates (one for each substrate), $\boldsymbol{\rho}^*$ are the saturation densities, and $\mathbf{U}$ are the uptake rates. In the formulation above, reactions can be included in the "remainder" term $\mathbf{R}$ or in $\mathbf{U}$ when quasi-linearized; for the first version of the software, we shall set $\mathbf{R} \equiv \mathbf{0}$, but the nonzero case is readily addressed by adding an additional solver. See Section 3.3 on operator splitting below. Also, in the first version of the solver, we assume $\mathbf{D}$ is constant; the more general case will be handled in future versions of the software.

The cell-based net source term $\mathbf{f}_{\text{cells}}$ models substrate secretion and uptake across the cells' volumes:

$$\mathbf{f}_{\text{cells}} = \sum_k 1_k(\mathbf{x}) \left[\overbrace{\overline{\mathbf{S}}_k \circ (\boldsymbol{\rho}_k^* - \boldsymbol{\rho})}^{\text{supply term}} - \overbrace{\overline{\mathbf{U}}_k \circ \boldsymbol{\rho}}^{\text{uptake term}} \right], \tag{6}$$

where $\left\{ \mathbf{x}_k, W_k, \overline{\mathbf{S}}_k, \overline{\mathbf{U}}_k, \boldsymbol{\rho}_k^* \right\}$ is a set of "cells" centered at position $\mathbf{x}_k$, with volume W_k, uptaking $\boldsymbol{\rho}$ at rates $\overline{\mathbf{U}}_k$ and secreting $\boldsymbol{\rho}$ with saturation densities $\boldsymbol{\rho}_k^*$ at rates $\overline{\mathbf{S}}_k$. For any cell k, $1_k(\mathbf{x})$ is an indicator function satisfying

$$1_k(\mathbf{x}) = \begin{cases} 1 & \text{if } \mathbf{x} \text{ is in cell } k \\ 0 & \text{elsewhere.} \end{cases} \tag{7}$$

2 Example: Oxygen and VEGF distribution in a large 3-D tissue

To demonstrate the code, we simulated oxygen release by blood vessels, consumption by a large number of tumor cells, and a simplified hypoxic response by the tumor cells. Oxygen is released by the vessels at a large rate so that $pO_2 \sim \rho_{O2}^* = 70$ mmHg on the vessels and approximately 50-60 mmHg nearby (typical of normal breast tissue) [23, 16]; it diffuses through the stroma ($D = 10^5$ μm^2/min [8] and $\lambda = 0.1$ min^{-1} for a 1 mm diffusion length scale ($\sqrt{D/\lambda}$) in low-density stroma [14]), and is consumed by tumor cells. We set the tumor cell uptake rate U_k according to the oxygen diffusion length scale $L = \sqrt{D/U_k}$, using an L with order of magnitude ~ 100 μm in dense tumor tissue [14]. For $L = 100$ μm, $\overline{U}_k = 10$ min^{-1}; we used this value in the example.

More precise measurements find the diffusion length scale to be 140-190 μm [18, 7, 22]. These more precise diffusion lengths can be directly related the tumor cell oxygen consumption rates using recent work by Grimes

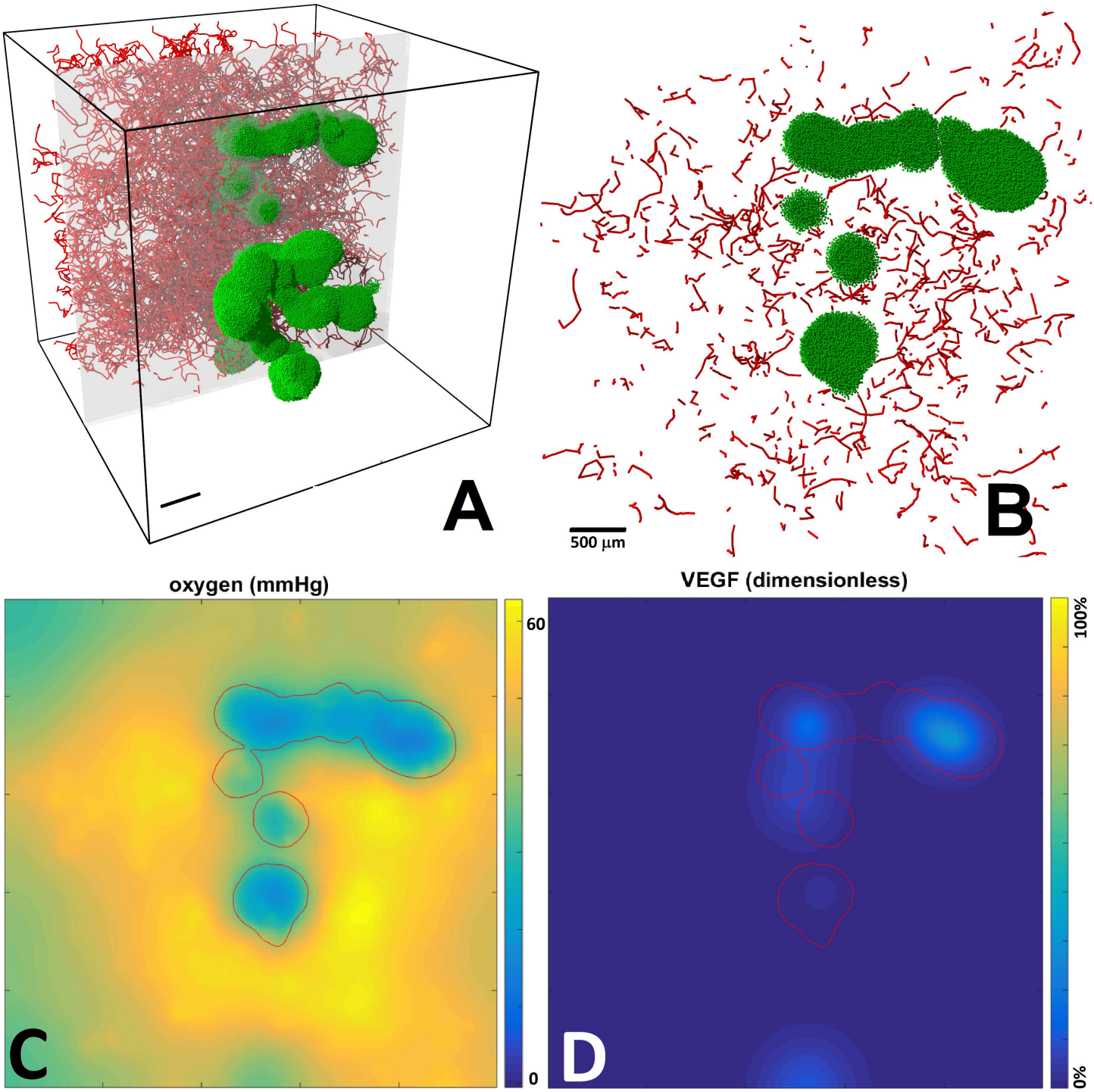

Figure 1) Simulation of oxygen and VEGF diffusion in a highly vascularized tissue with a multifocal tumor lesion; vasculature is rendered up to the gray clipping plane (panel A). Vessels and tumor cells in the gray clipping plane (panel B). Oxygen distribution in the (panel C) shows significant hypoxia (blue areas, $pO_2 < 15$ mmHg) within the tumor (red outline). Hypoxic tumor cells release VEGF to stimulate further vascularization (panel D).

et al. [6, 5], which established a steady-state model for oxygen diffusion in both cylindrical and spherical configurations. The increased diffusion distances would yield lower oxygen consumption rates. We note that lower oxygen consumption rates may allow computations with a larger time step size. See Sections 4.2 and 4.3.

Hypoxic tumor cells accumulate HIF-1α in low oxygen conditions [16]. (HIF-1α is a cellular hypoxia "sensor" molecule that stabilizes and accumulates in the absence of oxygen [10].) HIF-1α accumulation reaches half maximum at $pO_2 \approx$ 11-15 mmHg (1.5-2% oxygen [16]), and reaches its maximum value at $pO_2 \approx 4$ mmHg (0.5% oxygen [16]). Increased HIF-1α levels can trigger numerous downstream phenotype changes, including secretion of VEGF (vascular endothelial growth factor). Hypoxic phenotype changes (e.g., VEGF secretion to induce angiogenesis

[16, 23]) can be observed through gene expression and proteomic changes at 7-8 mmHg [23].

For technical illustration, we use a simplified hypoxic response model where VEGF secretion begins at $pO_2 = 15$ mmHg (where HIF-1α expression is first observed), and increases as oxygen tension is decreased towards the onset of phenotypic changes (7-8 mmHg), using the following form:

$$S_i = \begin{cases} \max\left(\frac{15-pO_2}{15-8}, 1\right) & \text{if } pO_2 < 15 \text{ mmHg} \\ 0 & \text{otherwise.} \end{cases} \tag{8}$$

In the simulation, the VEGF diffuses into the tissue with an ~ 300 μm diffusion length scale (chosen for this example to exceed the oxygen diffusion length scale).

In Figure 1, we show a simulation of 1 hour for this example in 125 mm^3 of vascularized tissue with a large irregular tumor at 20 μm resolution (15,625,000 voxels) and using and $\Delta t = 0.01$ min. Panels A and B show the blood vessels (red curves) and tumor cells (green spheres) in this large domain. In panel A, the vasculature is rendered up to a gray clipping plane for clearer illustration. Panel B shows the tumor cells and vessels contained in the gray clipping plane. Panels C-D shows the concentration of oxygen and VEGF in this plane. The red contour marks the tumor boundary. This simulation—with 2.8 million cell source/sink terms—required approximately 80 minutes on a quad-core desktop computer (Intel i7-4790, 3.60 GHz, 16 GB of memory). As we shall show below, simulations on smaller domains in the 1-1.5 million voxel size range take approximately 5-10 minutes to complete 60 minutes of diffusion.

The code to replicate this result is included in every BioFVM download as `main_experiment.cpp`. Users can either create their own initial cell and vessel positions or use the positions from this example. These initial cell and vessel positions are available at the BioFVM SourceForge site, under "Benchmark Datasets/Bioinformatics_main_example." The parameters used in this simulation are provided in Table 1.

Table 1: Parameters used in the simulation of Example 1.

Parameter	Description	Value	Units
D_{O_2}	oxygen diffusion coefficient	1.0e5	μm^2/min
D_{VEGF}	VEGF diffusion coefficient	1.0e3	μm^2/min
λ_{O_2}	oxygen "decay" rate	0.01	min^{-1}
λ_{VEGF}	VEGF "decay" rate	0.1	min^{-1}
$\boldsymbol{\rho}^*_{O_2}$	saturation oxygenation	70	mmHg
$\boldsymbol{\rho}^*_{VEGF}$	saturation VEGF value	1.0	dimensionless
U_{O_2}	oxygen uptake rate	10.0	min^{-1}
S_{O_2}	oxygen release rate	10.0	min^{-1}
S_{VEGF}	VEGF release reate	1.0	min^{-1}
Δt	time step size	0.01	min

2.1 Further discussion on blood vessel sources

Similar to many existing models (e.g., Frieboes et al. [3], Powathil et al. [19], Frieboes et al. [4] and Robertson-Tessi et al. [20]), we simulated blood vessels as line sources, approximated as a series of point sources (with Dirac delta functional form), using cell-centered sources as discussed in the previous section. This simplified the example and allows us to model substrate densities in tissues near vessels. However, blood vessels often have cross-sectional area larger than cells (potentially spanning multiple voxels).

Thus, it would be more ideal to model vessels as bulk sources. In this approach, we would set the "target" $\boldsymbol{\rho}^*$ substrate densities equal to physiological value on the vessel outer walls, but note that any voxels "inside" a vessel may have non-physiologic values. Alternatively, one could set aside specific voxels for the vessel interiors, set the substrate concentrations to the arterial/venous value in the vessel voxels (e.g., from a separate flow code), and then change the diffusion coefficient along the boundaries of these vessel voxels to simulate perfusion across vessel walls. The first approach can be implemented in the current version of BioFVM, using bulk sources. The second approach ideally would require more general, non-Cartesian volumetric meshes, and a spatially varying diffusion coefficient. Variable diffusion coefficients are planned for later versions of BioFVM, but this will require a much

more complex Thomas solver implementation with greater memory cost. Non-Cartesian lattices are also planned for future releases, but de-coupling dimensions is no longer possible, so either iterative matrix solvers are required (for implicit diffusion discretizations) or very small time steps (for explicit discretizations).

3 Computational details

The code is implemented in C++11, with parallelization in OpenMP. We have tested and support gcc (32-bit and 64-bit) Version 4.9.0 or later; in Windows, we use MinGW-W64 [17] (gcc Version 4.90 or better). The code does not require any installation or alteration of system environment variables (e.g., path); `BioFVM.h` can be included in the project, and the `BioFVM*.h` and `BioFVM*.cpp` files can simply be copied to the project's directory. The only external dependency is pugixml [9] (for XML parsing); a compatible version of this dependency is included with the BioFVM download.

3.1 Design goals and philosophy

- **Goal:** Simulate at least 5-10 diffusing substrates in 3-D, with a desktop workstation/supercomputer node.
- **Goal:** Simulate 3-D domains with at least 1 million voxels, with a desktop workstation/supercomputer node.
- Emphasize solving cell sources and sinks most commonly expected in biological problems.
- Use schemes that are stable but simple to implement.
- Reduce complexity by minimizing external dependencies. Never require an entire library if only using one or two functions.
- Prioritize earlier software release at first-order time, second-order spatial accuracy, rather than later release on higher-order time accuracy. Later releases can increase accuracy as needed.
- Prioritize earlier release on simple Cartesian meshes over full Voronoi mesh generality. Later releases can support more general meshes as needed.
- Simplify diffusion discretization to allow straightforward OpenMP parallelization.
- Prioritize execution speed over memory footprint.

3.2 Domain discretization and notation

Let $\{\Omega_i\}_{i=0}^N$ be a set of voxels satisfying $\bigcup_{i=0}^N \Omega_i = \Omega$, with centroids $\{\mathbf{x}_i\}_{i=0}^N$, volumes $\{V_i = |\Omega_i|\}_{i=0}^N$, and boundaries $\{\Sigma_i = \partial\Omega_i\}_{i=0}^N$. For each voxel i, let N_i be the index set of neighboring voxels. For any voxel i and for each $j \in N_i$, let $\Sigma_{ij} = \Sigma_i \cap \Sigma_j$ be the shared boundary between Ω_i and Ω_j, with centroid $\mathbf{x}_{ij}$, outward normal vector (into Ω_j) $\mathbf{n}_{ij}$, and surface area $S_{ij} = |\Sigma_{ij}|$. Define $\Delta\mathbf{x}_{ij} = \mathbf{x}_j - \mathbf{x}_i$ and $\Delta x_{ij} = |\Delta\mathbf{x}_{ij}|$. Let $\mathbf{D}_{ij} = \mathbf{D}(\mathbf{x}_{ij})$.

For a fixed time step size Δt, let $t_n = t_0 + n\Delta t$. For any voxel Ω_i and any time $t \geq t_0$, define $\mathbf{u}_i^n = \int_{\Omega_i} \boldsymbol{\rho}^n(\mathbf{x})dV$, and denote the mean density $\boldsymbol{\rho}$ at time t_n in voxel Ω_i by

$$\boldsymbol{\rho}_i^n = \frac{\mathbf{u}_i^n}{V_i} = \frac{\int_{\Omega_i} \boldsymbol{\rho} dV}{V_i}. \tag{9}$$

For any other function $\mathbf{f}(\mathbf{x}, t)$ (e.g., bulk sources), denote $\mathbf{f}_i^n = \mathbf{f}(\mathbf{x}_i, t_n)$ for any voxel Ω_i and discretized time t_n.

3.3 Operator splitting and the overall algorithm

We solve Eqn. 1 by splitting the overall operator into several simpler operators, each of which can individually be solved by tailored, optimized algorithms [15]. This allows extra flexibility in future performance/accuracy tradeoff decisions, and allows the introduction of future operators (e.g., advective terms) without rewriting the main solvers. As noted in section 1, we set $\mathbf{R} \equiv \mathbf{0}$. Adding a reaction term is straightforward by generalizing the

operator splitting: follow the other steps by an additional solver for the reaction term, such as by quasi-linearizing the operator and using an implicit solver.

To advance the solution from $\boldsymbol{\rho}^n$ at time t_n to $\boldsymbol{\rho}^{n+1}$ at time $t_n + \Delta t$, we shall use a first-order splitting [15]:

1. **Diffusion-decay:** Solve for $\boldsymbol{\sigma}$:
$$\frac{\boldsymbol{\sigma} - \boldsymbol{\rho}^n}{\Delta t} = \nabla \cdot (\mathbf{D} \circ \nabla \boldsymbol{\sigma}) - \boldsymbol{\lambda} \circ \boldsymbol{\sigma} \tag{10}$$
See Section 3.4 for details on this solver (and particularly Section 3.4.1 on our LOD implementation for Cartesian meshes). Note that this uses a stable, implicit time discretization.

2. **Bulk supply/uptake:** Solve for $\boldsymbol{\sigma}^*$ with the implicit time discretization:
$$\frac{\boldsymbol{\sigma}^* - \boldsymbol{\sigma}}{\Delta t} = \mathbf{S}(\mathbf{x}, t) \circ (\boldsymbol{\rho}^* - \boldsymbol{\sigma}^*) - \mathbf{U}(\mathbf{x}, t) \circ \boldsymbol{\sigma}^*$$
which can then be solved for $\boldsymbol{\sigma}_i^*$ at each voxel Ω_i by:
$$\boldsymbol{\sigma}_i^* = \Big(\boldsymbol{\sigma}_i + \Delta t \boldsymbol{S}_i(t) \circ \boldsymbol{\rho}^*\Big) /\!\!/ \Big(\mathbf{1} + \Delta t \mathbf{S}_i(t) + \Delta t \mathbf{U}_i(t)\Big) \tag{11}$$
In practice, $\boldsymbol{\sigma}^*$ can overwrite $\boldsymbol{\sigma}$ on the same data array, thus reducing memory allocation and copy time costs and accelerating the solver. These operations can be safely parallelized across the processor cores by OpenMP.

3. **Cell supply/uptake:** Solve for $\boldsymbol{\rho}^{n+1}$ with the implicit time discretization:
$$\frac{\boldsymbol{\rho}^{n+1} - \boldsymbol{\sigma}^*}{\Delta t} = \sum_k 1_k(\mathbf{x}) \left[\overline{\mathbf{S}}_k \circ \left(\boldsymbol{\rho}_k^* - \boldsymbol{\rho}^{n+1}\right) - \overline{\mathbf{U}}_k \circ \boldsymbol{\rho}^{n+1}\right]. \tag{12}$$
See Section 3.5 for details on this solver. In practice, $\boldsymbol{\rho}^{n+1}$ can overwrite $\boldsymbol{\sigma}^*$ on the same data array, thus reducing memory allocation and copy time costs and accelerating the solver.

3.4 Finite volume method (FVM): Voronoi meshes and Cartesian meshes

We use the finite volume method (FVM) [2] to solve Eqn. 10 with Neumann boundary conditions. We will assume a Voronoi mesh, so that in particular, $\mathbf{x}_{ij} = \frac{1}{2}(\mathbf{x}_i + \mathbf{x}_j)$ and $\mathbf{n}_{ij} = \Delta \mathbf{x}_{ij} / \Delta x_{ij}$ for any neighboring voxels i and j. For any voxel i, we integrate Eqn. 10 over Ω_i and apply the divergence theorem, obtaining:
$$\begin{aligned} \frac{1}{\Delta t} \int_{\Omega_i} (\boldsymbol{\sigma}^* - \boldsymbol{\rho}^n)\, dV &= \int_{\partial \Omega_i} (\mathbf{D} \circ \nabla \boldsymbol{\sigma}^*) \cdot \mathbf{n} dS - \int_{\Omega_i} \boldsymbol{\lambda} \circ \boldsymbol{\sigma}^* dV \\ \implies \frac{1}{\Delta t} (\boldsymbol{\sigma}_i^* V_i - \mathbf{u}_i^n) &\approx \sum_{j \in N_i} \mathbf{D}_{ij} \circ \left(\frac{\boldsymbol{\sigma}_j^* - \boldsymbol{\sigma}_i^*}{\Delta x_{ij}}\right) S_{ij} - \boldsymbol{\lambda} \circ \boldsymbol{\sigma}_i^* V_i \end{aligned} \tag{13}$$
Finally, we divide by V_i to obtain the implicit discretization of the finite volume method:
$$\left(\mathbf{1} + \Delta t \boldsymbol{\lambda} + \Delta t \sum_{j \in N_i} \frac{S_{ij}}{\Delta x_{ij} V_i} \mathbf{D}_{ij}\right) \circ \boldsymbol{\sigma}_i^* - \Delta t \sum_{j \in N_i} \frac{S_{ij}}{\Delta x_{ij} V_i} \mathbf{D}_{ij} \circ \boldsymbol{\sigma}_j^* = \boldsymbol{\rho}_i^n, \tag{14}$$
which requires solution as a large but sparse linear system.

3.4.1 Locally-one dimensional (LOD) method for Cartesian meshes

Consider the special case of the domain $\Omega = [x_L, x_U] \times [y_L, y_U] \times [z_L, z_U]$ with a regular Cartesian discretization with x, y, and z step sizes given by Δx, Δy, and Δz, respectively, with N_x x-nodes, N_y y-nodes, and N_z z-nodes. Then for each $(i, j, k) \in [0, N_x - 1] \times [0, N_y - 1] \times [0, N_z - 1]$, the voxels are defined by
$$m(i, j, k) = i + jN_x + kN_xN_y, \tag{15}$$
$$\mathbf{x}_m = \mathbf{x}_{m(i,j,k)} = \mathbf{x}_{i,j,k} = [x_i, y_j, z_k] \tag{16}$$
$$\Omega_m = \Omega_{m(i,j,k)} = \Omega_{i,j,k} = \mathbf{x}_m + \left[-\frac{\Delta x}{2}, \frac{\Delta x}{2}\right] \times \left[-\frac{\Delta y}{2}, \frac{\Delta y}{2}\right] \times \left[-\frac{\Delta z}{2}, \frac{\Delta z}{2}\right]. \tag{17}$$

Notice then that $V_m = \Delta x \Delta y \Delta z$ for all m, and

$$S_{m,m\pm1} = \Delta y \Delta z \qquad \Delta x_{m,m\pm1} = \Delta x \tag{18}$$
$$S_{m,m\pm N_x} = \Delta x \Delta z \qquad \Delta x_{m,m\pm N_x} = \Delta y \tag{19}$$
$$S_{m,m\pm N_x N_y} = \Delta x \Delta y \qquad \Delta x_{m,m\pm N_x N_y} = \Delta z. \tag{20}$$

For this mesh structuring, we can use the locally one-dimensional (LOD) method to obtain fast and accurate solutions [24, 15]. This method (like the related alternating directions implicit (ADI) method) splits a higher-dimensional PDE into a series of related one-dimensional PDEs to be solved with fast matrix solvers. We do this by splitting the operator in Eqn. 10 as

$$\frac{\boldsymbol{\eta} - \boldsymbol{\rho}^n}{\Delta t} = \partial_x \left(\mathbf{D} \circ \partial_x \boldsymbol{\eta} \right) - \frac{1}{3} \boldsymbol{\lambda} \circ \boldsymbol{\eta} \tag{21}$$
$$\frac{\boldsymbol{\eta}^* - \boldsymbol{\eta}}{\Delta t} = \partial_y \left(\mathbf{D} \circ \partial_y \boldsymbol{\eta}^* \right) - \frac{1}{3} \boldsymbol{\lambda} \circ \boldsymbol{\eta}^* \tag{22}$$
$$\frac{\boldsymbol{\sigma} - \boldsymbol{\eta}^*}{\Delta t} = \partial_z \left(\mathbf{D} \circ \partial_z \boldsymbol{\sigma} \right) - \frac{1}{3} \boldsymbol{\lambda} \circ \boldsymbol{\sigma} \tag{23}$$

Applying the finite volume method from Eqn. 14 to Eqn. 21 and setting $\mathbf{D}$ constant, for each voxel i we have:

$$\left(\mathbf{1} + \frac{1}{3} \Delta t \boldsymbol{\lambda} + \frac{\Delta t \# N_i}{\Delta x^2} \mathbf{D} \right) \circ \boldsymbol{\eta}_i - \sum_{j \in N_i} \frac{\Delta t}{\Delta x^2} \mathbf{D} \circ \boldsymbol{\eta}_j \quad = \quad \boldsymbol{\rho}_i^n. \tag{24}$$

For any fixed $0 \le j < N_y$ and $0 \le k < N_z$, and over the range $0 \le i < N_x$ (that is, for m from $m_L = m(0, j, k)$ to $m_U = m(N_x - 1, j, k)$, we obtain the tridiagonal vector system:

$$\left(\mathbf{1} + \frac{1}{3} \Delta t \boldsymbol{\lambda} + \frac{\Delta t}{\Delta x^2} \mathbf{D} \right) \circ \boldsymbol{\eta}_{m_L} - \frac{\Delta t}{\Delta x^2} \mathbf{D} \circ \boldsymbol{\eta}_{m_L+1} \quad = \quad \boldsymbol{\rho}_{m_L}^n \tag{25}$$
$$-\frac{\Delta t}{\Delta x^2} \mathbf{D} \circ \boldsymbol{\eta}_{m-1} + \left(\mathbf{1} + \frac{1}{3} \Delta t \boldsymbol{\lambda} + 2 \frac{\Delta t}{\Delta x^2} \mathbf{D} \right) \circ \boldsymbol{\eta}_m - \frac{\Delta t}{\Delta x^2} \mathbf{D} \circ \boldsymbol{\eta}_{m+1} \quad = \quad \boldsymbol{\rho}_m^n \qquad m_L < m < m_U \tag{26}$$
$$-\frac{\Delta t}{\Delta x^2} \mathbf{D} \circ \boldsymbol{\eta}_{m_U-1} + \left(\mathbf{1} + \frac{1}{3} \Delta t \boldsymbol{\lambda} + \frac{\Delta t}{\Delta x^2} \mathbf{D} \right) \circ \boldsymbol{\eta}_{m_U} \quad = \quad \boldsymbol{\rho}_{m_U}^n \tag{27}$$

This tridiagonal linear system that can be solved efficiently and directly by the Thomas algorithm (applied vectorially) [21]. Similar calculations in the $y-$ and $z-$ directions (Equations 22-23) give additional tridiagonal linear systems to solve. Thus, our overall algorithm for the diffusion-decay equation is:

1. **x-diffusion:** For each $0 \le j < N_y$ and each $0 \le k < N_z$, solve the tridiagonal system (Equations 25-27) along the strip $0 \le i < N_x$ ($m(0, j, k) = m_L \le m \le m_U = m(N_x - 1, j, k)$). Solving along each strip is a serial operation, but the iteration across strips can be parallelized with OpenMP (a bundle of independent, simultaneous Thomas solvers). In practice, the updated solutions ($\boldsymbol{\eta}$) can overwrite the previous solutions ($\boldsymbol{\rho}^n$), thus reducing memory allocation and copy costs.

2. **y-diffusion:** For each $0 \le i < N_x$ and each $0 \le k < N_z$, solve the analogous tridiagonal system along the strip $0 \le j < N_y$. Each strip can be solved with a separate Thomas solver, and thus parallelized with OpenMP as in x-diffusion. In practice, the updated solutions ($\boldsymbol{\eta}^*$) can overwrite the previous solutions ($\boldsymbol{\eta}$), thus reducing memory allocation and copy costs.

3. **z-diffusion:** For each $0 \le i < N_x$ and each $0 \le j < N_y$, solve the analogous tridiagonal system along the strip $0 \le z < N_z$. Each strip can be solved with a separate Thomas solver, and thus parallelized with OpenMP as in x-diffusion. In practice, the updated solutions ($\boldsymbol{\sigma}$) can overwrite the previous solutions ($\boldsymbol{\eta}^*$), thus reducing memory allocation and copy costs.

Thus, Cartesian meshes with an additional locally one-dimensional (LOD) operator splitting allows us the stability of an implicit scheme without the difficulty of solving large linear systems; instead we directly solve simple tridiagonal systems. Morever, this formulation allows us to readily apply OpenMP to parallelize the code (by using independent Thomas solvers for each "strip").

3.4.2 Accelerations

To further accelerate the algorithm, we apply the Thomas solvers to the entire vector of substrates simultaneously, taking advantage of SIMD (single inststruction multiple data) optimizations on modern CPUs. Similarly, we use operator overloading (e.g., `operator+=`) on the vectors of substrates and vectors of coefficients to minimize the use of hidden, temporary variables that incur substantial memory allocation, copy, and dealloation costs. (For instance, $\mathbf{x} = \mathbf{x} + \mathbf{y}$ temporarily allocates memory for hidden variable $\mathbf{z}$, stores the result of $\mathbf{x} + \mathbf{y}$ in $\mathbf{z}$, copies the result back onto $\mathbf{x}$, and then de-allocates $\mathbf{z}$. Using $\mathbf{x}$ += $\mathbf{y}$ eliminates these costs by directly overwriting each component x_i with $x_i + y_i$.)

We optimized custom level 1 BLAS "axpy" operators ($\mathbf{y} = \mathbf{a} \circ \mathbf{x} + \mathbf{y}$ and $\mathbf{y} = a\mathbf{x} + \mathbf{y}$) to streamline common operations in the Thomas and other solvers, without requiring an entire library for these simple instructions. We implemented a specialized vectorized Thomas solver tailored to the diffusion-decay system using these overloaded and axpy instructions, with further optimizations by pre-computing the forward sweeps and storing them for the x-, y-, and z- diffusion operators. Note that separating diffusion and decay terms from the overall PDE makes this optimization possible, as $\mathbf{D}$ and $\boldsymbol{\lambda}$ do not change from iteration to iteration, but the supply and uptake terms do.

3.5 Cell supply/uptake solver

In the first release of the solver, we approximate the cell supply/uptake term by "concentrating" it to a single voxel with a Dirac delta function. For each cell k, let i_k be the index of the voxel containing the cell center $\mathbf{x}_k$; i.e., $\mathbf{x}_k \in \Omega_{i_k}$. Then we approximate Eqn. 12 by

$$\frac{\boldsymbol{\rho}^{n+1} - \boldsymbol{\sigma}^*}{\Delta t} = \sum_k W_k \delta\left(\mathbf{x} - \mathbf{x}_k\right) \left[\overline{\mathbf{S}}_k \circ \left(\boldsymbol{\rho}_k^* - \boldsymbol{\rho}^{n+1}\right) - \overline{\mathbf{U}}_k \circ \boldsymbol{\rho}^{n+1})\right], \tag{28}$$

where $\delta\left(\mathbf{x}\right)$ is the Dirac delta function, which we approximate by

$$\delta\left(\mathbf{x} - \mathbf{x}_k\right) \approx \begin{cases} \frac{1}{V_{i_k}} & \text{if } \mathbf{x} \in \Omega_{i_k} \\ 0 & \text{elsewhere.} \end{cases} \tag{29}$$

To solve this, we iterate over all cells k with:

$$\boldsymbol{\rho}_{i_k}^{n+1} = \left(\boldsymbol{\sigma}_{i_k}^* + \Delta t \frac{W_k}{V_{i_k}} \overline{\mathbf{S}}_k \circ \boldsymbol{\rho}_k^*\right) /\!/ \left(\mathbf{1} + \Delta t \frac{W_k}{V_{i_k}} \left(\overline{\mathbf{S}}_k + \overline{\mathbf{U}}_k\right)\right) \tag{30}$$

These operations can be safely parallelized across the processor cores by OpenMP.

Note that this discretization can be less accurate for cells that are the same size of typical computational voxel or larger. In this case, accuracy can be improved by either (1) "rasterizing" the cell-based supply and uptake onto individual voxels, effectively by calculating the intersection of each cell's volume on each voxel, or (2) sub-dividing the cell into smaller "subcells" and computing their contribution to transport directly, using the same algorithm as in Eqn. 30. We plan to add this capability in future releases.

3.6 Approximating Dirichlet conditions on selected voxels

To approximate Dirichlet conditions on one or more selected voxels, the code can overwrite the data stored in any voxel with a prescribed vector of values. The code performs this step after each operator of of the LOD algorithm (i.e., after x-diffusion, after y-diffusion, and after z-diffusion; see Section 3.4.1 above). However, to improve accuracy, we would need to modify the computational stencil of the diffusion operator anywhere a specified Dirichlet node appears. For the LOD algorithm, this would require individual Thomas solvers for each x-strip, for each y-strip, and each z-strip. We defer this development until a later release, which will also facilitate a discretization of $\nabla \cdot (\mathbf{D} \circ \nabla \boldsymbol{\rho})$ rather than $\mathbf{D} \circ \nabla^2 \boldsymbol{\rho}$.

4 Convergence testing

4.1 Example 1: 1-D diffusion (with analytical solution)

We first test the convergence of the diffusion-decay solver against a 1-D problem with a known analytical solution:

$$\frac{\partial \rho}{\partial t} = D\partial_x^2 \rho \qquad -L_0 < x < L_0, t > 0 \tag{31}$$

$$\frac{\partial \rho}{\partial x}(x,t) = 0 \qquad x \in \{-L_0, L_0\}, t > 0 \tag{32}$$

$$\rho(x,0) = 1 + \cos\left(\frac{\pi}{L_0}x\right) \qquad -L_0 \leq x \leq L_0, \tag{33}$$

with exact solution

$$\rho(x,t) = 1 + \mathrm{e}^{-\beta t}\cos\left(\frac{\pi}{L_0}x\right), \text{ where } \beta = \frac{\pi^2 D}{L_0^2}, \tag{34}$$

and where we used $L_0 = 500\,\mu$m and $D = 10^5 \mu\text{m}^2/\text{min}$ ($D = 1.75 \times 10^{-5}\text{cm}^2/\text{sec} = 1.05 \times 10^5 \mu\text{m}^2/\text{min}$ for oxygen in tissues as measured in [8]). The solution is plotted for several times in Fig. 2. The source code for this problem can be found in the `examples` directory of any BioFVM download (file: `convergence_test1.cpp`).

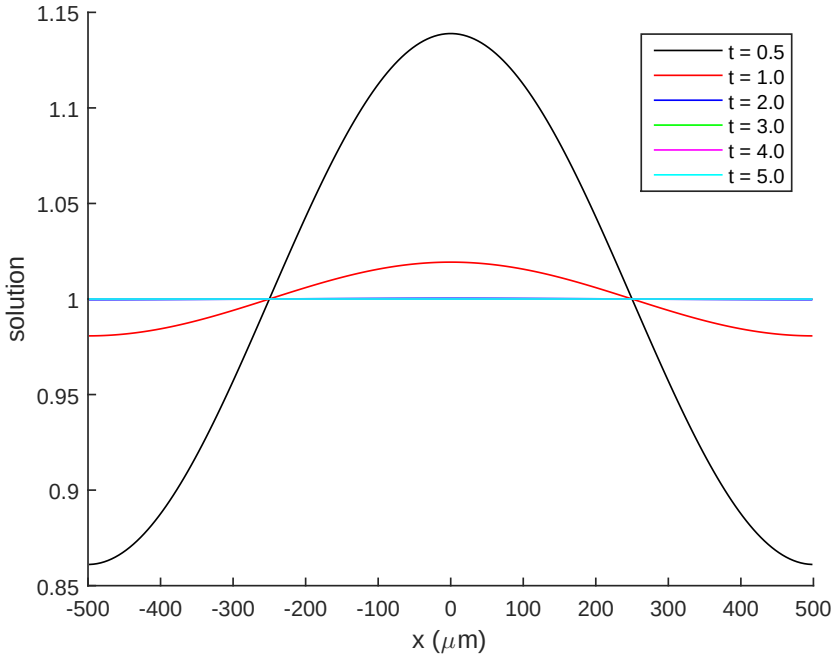

Figure 2) Solution for Example 1 (Section 4.1), plotted at $t = 0.5, 1, 2, 3, 4, 5$ minutes.

4.1.1 Convergence in time

Let $\rho_{\Delta t, \Delta x}$ denote the numerical solution simulated with time step size Δt and spatial step size Δx. For any given norm $||\cdot||$, the error for the algorithm should take the form

$$\text{Err}(\Delta t, \Delta x) = ||\rho(\mathbf{x}, t) - \rho_{\Delta t, \Delta x}(\mathbf{x}, t)|| \sim A\Delta t^m + B\Delta x^n. \tag{35}$$

To test for the convergence in Δt, we must choose Δx sufficiently small that $B\Delta x^n \ll A\Delta t^m$. In this case,

$$\text{Err} \sim A\Delta t^m \Longrightarrow \log(\text{Err}) \sim \log(A) + m\log(\Delta t). \tag{36}$$

Thus, we calculate the order of convergence as the slope of the linear least squares fit of $\log(\text{Err})$ versus $\log(\Delta t)$. We use the ℓ_∞ norm (the maximum absolute error over all voxels). We perform convergence testing with $\Delta x = 5\,\mu$m,

with $\Delta t \in \{10^{-5}, 10^{-4}, 10^{-3}, 10^{-2}, 10^{-1}, 1\}$ min. The error values are plotted in Fig. 3 (left) and recorded in Table 2. In this example, we see first-order convergence in time at several solution times (showing good accuracy on both short and long time scales), and stability even for Δt much larger than the CFL condition for explicit discretizations (on the order of $\Delta t_{\text{CFL}} \sim 10^{-4}$ min). Note that the for any fixed Δt, the errors decrease over time; this shows that the solutions demonstrate better accuracy as solutions approach steady state.

The errors show evidence of saturating for very small Δt, where the error from the spatial discretization is likely to dominate, and so $B\Delta x^n \ll A\Delta t^m$ no longer holds for the convergence calculation. The error for $\Delta t = 1$ min (an unlikely choice of time step size in most applications) is comparatively large, giving a large drop in error between $\Delta t = 1$ min and $\Delta t = 0.1$ min; this large drop may unduly increase the computed convergence rate. For these reasons, we also calculated the convergence rate based upon the data for $10^{-4} \leq \Delta t \leq 10^{-1}$; see the rightmost column in Table 2.

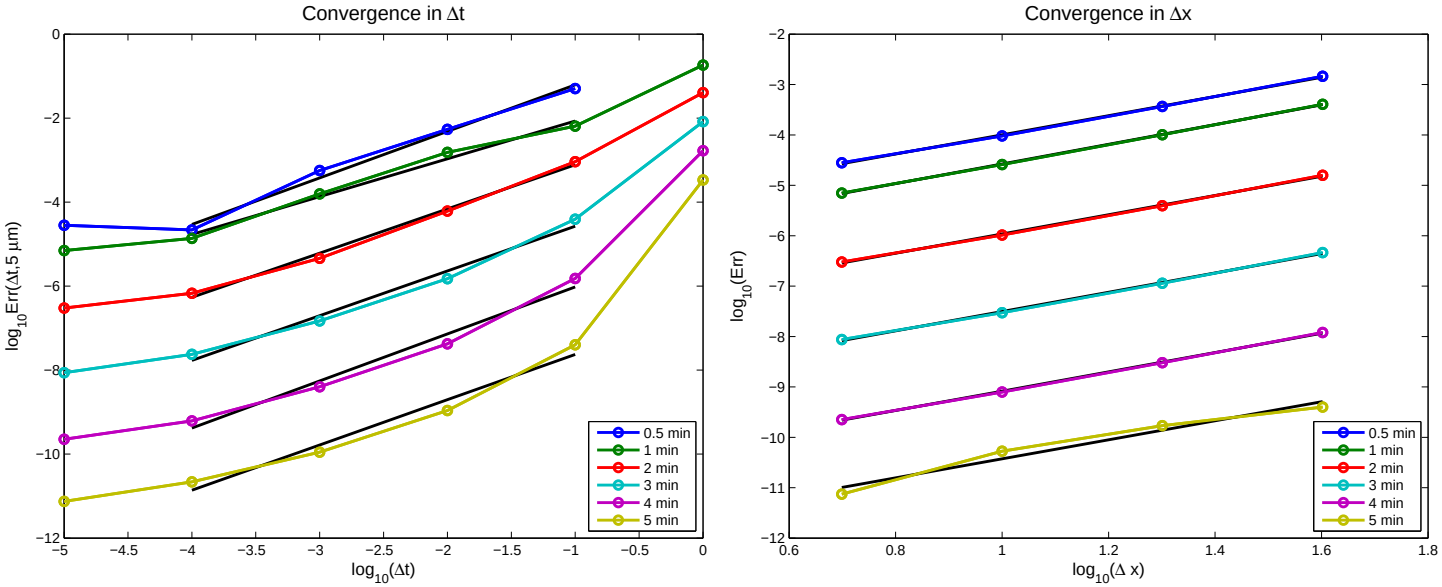

Figure 3) Left: Convergence in Δt for Example 1 (Section 4.1), with $\Delta x = 5\ \mu$m. Each curve gives the error across Δt at a different time (from top to bottom: t=0.5, 1, 2, 3, 4, and 5 minutes). Notice that for each fixed resolution Δt, the error improves in time as the solution approaches steady state. The linear least-squares fits are plotted for $10^{-4} \leq \Delta t \leq 10^{-1}$; the slope of each line gives the order of convergence at that time; see Table 2.
Right: Convergence in Δx for Example 1 (Section 4.1), with $\Delta t = 10^{-5}$ min. Each curve gives the error across Δx at a different time (from top to bottom: t=0.5, 1, 2, 3, 4, and 5 minutes). Notice that for each fixed resolution Δx, the error improves in time as the solution approaches steady state. The slope of each linear least squares fit (black curves) gives the order of convergence at that time; see Table 3.

Table 2: Convergence in Δt for Example 1 (Section 4.1), with $\Delta x = 5\ \mu$m. At each time, we observe first-order convergence. The solutions are stable even for large Δt, whereas the CFL condition for explicit solvers is $\Delta t < 1.25 \times 10^{-4}$ min.
*As we cannot compute $\rho_{1\text{ min},5\ \mu\text{m}}(0.5\text{ min})$, we used the data for $10^{-5} \leq \Delta t \leq 10^{-1}$ to compute the order of convergence at 0.5 min.
**At each time, the error approaches a saturating value when the spatial discretization errors begin to dominate (below $\sim \Delta t = 10^{-4}$). Moreover, the large errors for $\Delta t = 1$ min (a step size we would not anticipate using) artificially increase the order of convergence. Thus, we also computed the the order of convergence based on the data for $10^{-4} \leq \Delta t \leq 10^{-1}$.

Time	Err($\Delta t, 5\ \mu$m), where Δt is:							
(min)	1 min	0.1 min	10^{-2} min	10^{-3} min	10^{-4} min	10^{-5} min	order	order**
0.5	n.a.*	5.0542e-02	5.3974e-03	5.6336e-04	2.1800e-05	2.8000e-05	0.89*	1.11
1	1.8280e-01	6.4385e-03	1.5287e-03	1.5683e-04	1.3700e-05	7.0100e-06	0.88	0.90
2	4.0476e-02	9.1597e-04	6.1300e-05	4.5900e-06	6.7500e-07	3.0000e-07	1.03	1.05
3	8.2491e-03	3.9100e-05	1.5000e-06	1.4800e-07	2.3800e-08	8.6800e-09	1.16	1.07
4	1.6686e-03	1.5200e-06	4.2300e-08	4.0000e-09	6.1900e-10	2.2500e-10	1.30	1.12
5	3.3729e-04	4.0000e-08	1.0900e-09	1.1100e-10	2.1900e-11	7.4700e-12	1.40	1.08
mean							1.11	1.06

4.1.2 Convergence in space

To ensure that $A\Delta t^m \ll B\Delta x^m$ in Eqn. 35, we use $\Delta t = 10^{-5}$ min. Similarly to before,

$$\log(\text{Err}) \sim \log(B) + n\log(\Delta x). \tag{37}$$

and the order of convergence can be calculated as the slope of the linear least squares fit of log(Err) versus $\log(\Delta x)$. The errors are reported in Table 3 and plotted in Figure 3 (right). At all times, the solutions demonstrate approximately second-order convergence. Moreover, for any fixed Δx, the accuracy improves over time as the solution approaches steady state.

Table 3: Convergence in Δx for Example 1 (Section 4.1), with $\Delta t = 10^{-5}$ min. For all times, we observe approximately second-order convergence.

	Err(10^{-5} min, Δx)				
Time (min)	$\Delta x = 40\mu$m	$\Delta x = 20\mu$m	$\Delta x = 10\mu$m	$\Delta x = 5\mu$m	order
0.5	1.4530e-03	3.6600e-04	9.5600e-05	2.8000e-05	1.90
1	4.0500e-04	1.0100e-04	2.5800e-05	7.0100e-06	1.95
2	1.5800e-05	3.9400e-06	1.0300e-06	3.0000e-07	1.91
3	4.6300e-07	1.1400e-07	2.9700e-08	8.6800e-09	1.92
4	1.2000e-08	3.0300e-09	7.9300e-10	2.2500e-10	1.91
5	3.9800e-10	1.7000e-10	5.3000e-11	7.4700e-12	1.89
mean					1.91

4.1.3 Selecting Δt for reasonable accuracy at $\Delta x = 20\mu$m resolution

After investigating convergence in Δt and Δx, we also examined the relative accuracy ($\max |\rho_{\Delta t,\Delta x} - \rho| \,/\, |\rho|$) of solutions at $\Delta x = 20\ \mu$m (a typical spatial resolution for large multicellular systems problems in cancer and tissue engineering) at a variety of time step sizes. In Fig. 4 and Table 4, we see for the solutions are stable for all tested time steps, and that for $\Delta t = 0.01$ min, the relative error never exceeds approximately 0.5% at any of the tested times; with $\Delta t = 0.1$ min, the error remains below approximately 5% all tested times, and below approximately 0.7% for all $t \geq 1$ min (close to quasi-steady conditions that may be expected in many cancer and tissue engineering problems). This means that for diffusion-decay problems with parameters of this order of magnitude, we can use step sizes that are 10-100 times larger than the explicit CFL condition (1D: $(20\ \mu\text{m})^2/(2 \times 10^5\ \mu\text{m}^2/\text{min}) \sim 0.002$ min) and still obtain reasonable accuracy, especially as systems approach a quasi-steady state.

Table 4: Relative accuracy versus Δt for $\Delta x = 20\ \mu$m in Example 1 (Section 4.1). At this spatial resolution, $\Delta t = 0.01$ or $\Delta t = 0.1$ min may be sufficient for many common problems.

time	Relative Err($\Delta t, 20\ \mu$m), where Δt is:					
(min)	1 min	0.1 min	10^{-2} min	10^{-3} min	10^{-4} min	10^{-5} min
0.5	n.a.	0.058693982	0.004739153	0.000654229	1.92e-05	3.25e-05
1	0.186399099	0.006565157	0.001499716	0.000159917	1.34e-05	7.15e-06
2	0.040490869	0.000915632	6.13e-05	4.59e-06	6.75e-07	3.00e-07
3	0.008249157	3.91e-05	1.50e-06	1.48e-07	2.38e-08	8.68e-09
4	0.001668634	1.52e-06	4.23e-08	4.00e-09	6.19E-10	2.25E-10
5	0.000337292	4.00e-08	1.09e-09	1.11E-10	2.19E-11	7.47E-12

4.2 Example 2: 3-D diffusion-reaction with bulk sources

We next tested the convergence of the overall 3-D simulator, including "bulk sources", using the problem

$$\frac{\partial \rho}{\partial t} = D\nabla^2\rho - \lambda\rho + f(\mathbf{x}) \cdot (\rho^* - \rho) \qquad \mathbf{x} \in \Omega \tag{38}$$

$$D\nabla\rho \cdot \mathbf{n} = 0 \qquad \mathbf{x} \in \partial\Omega \tag{39}$$

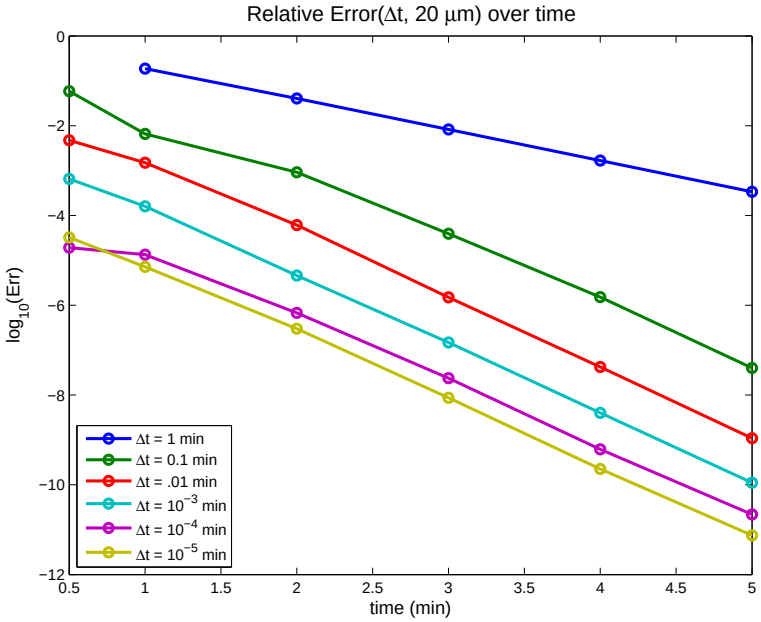

Figure 4) Relative accuracy versus Δt for $\Delta x = 20\ \mu$m in Example 1 (Section 4.1). At this spatial resolution, $\Delta t = 0.01$ or $\Delta t = 0.1$ min may be sufficient for many common problems.

where $\Omega = [-L_0, L_0] \times [-L_0, L_0] \times [-L_0, L_0]$ with $L_0 = 500\ \mu$m, and f is defined by

$$f(\mathbf{x}) \quad = \quad \begin{cases} r & \text{if } ||x| - L_0| \leq 40\ \mu\text{m or } ||y| - L_0| \leq 40\ \mu\text{m or } ||z| - L_0| \leq 40 \\ 0 & \text{elsewhere.} \end{cases} \tag{40}$$

As in the prior example, we set $D = 10^5\ \mu\text{m}^2/\text{min}$, and we choose $\lambda = 0.1\ \text{min}^{-1}$ so that the diffusional length scale $L = \sqrt{D/\lambda} = 1000\ \mu$m in "background tissue" with no cell uptake sources (similarly to our work in [14]). We set $\rho^* = 38$ mmHg (corresponding to 5% oxygenation conditions, typical of physiologic conditions of 1-11% oxygenation in tissue [1, 16]). We set a large value of r (for simplicity, $r = 38\ \text{min}^{-1}$) to simulate a very strong source that maintains the nearest tissue close to ρ^*. A typical solution ($\Delta x = 10\,\mu$m, $\Delta t = 10^{-4}$ min) at $t = 4$ min is plotted in Figure 5. The source code for this problem can be found in the `examples` directory of any BioFVM download (file: `convergence_test2.cpp`).

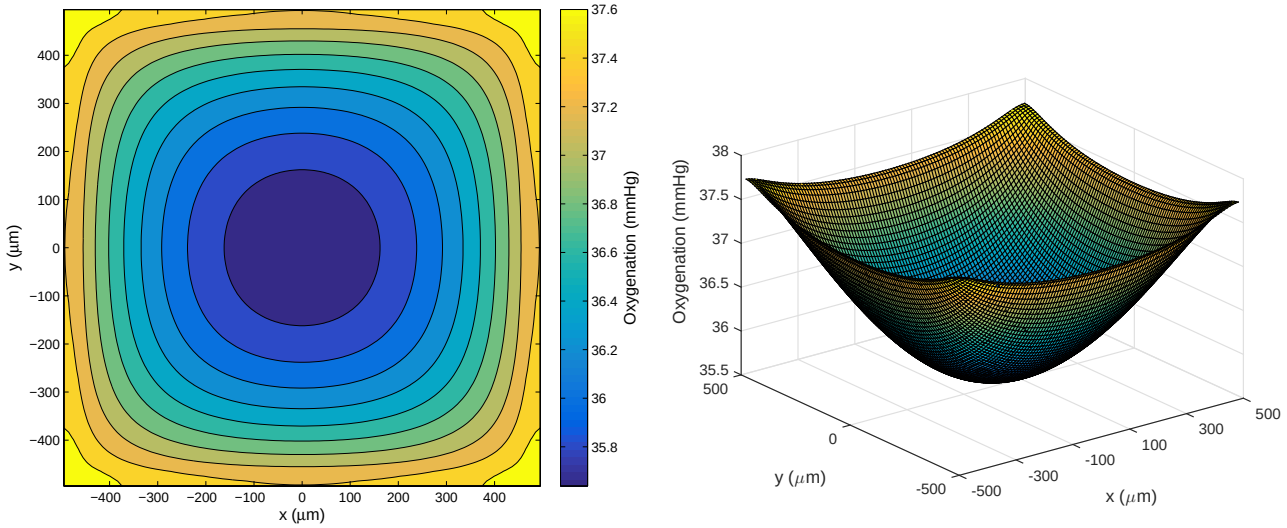

Figure 5) Solution for Example 2 (Section 4.2) after 4 minutes, computed with $\Delta x = 10\,\mu$m and $\Delta t = 10^{-4}$ min. **left:** Contour plot through $z = 5\,\mu$m. **right:** Surface plot through the same cross-section.

4.2.1 Convergence in time

Unlike the previous example, we do not have an analytical solution. Following our prior work [11, 12, 13], we define the numerical rate of convergence by first denoting

$$\rho_{\Delta t,\Delta x}(\mathbf{x},t) \quad = \quad \text{numerical solution at } (\mathbf{x},t) \text{ obtained with } \Delta t \text{ and } \Delta x \tag{41}$$

$$\text{Err}(\Delta t,\Delta x,t) \quad = \quad \max_{\text{all voxels } i} \left|\rho^*(\mathbf{x}_i,t) - \rho_{\Delta t,\Delta x}(\mathbf{x}_i,t)\right|, \tag{42}$$

where $\rho(\mathbf{x},t)$ is an approximation to the true solution, typically computed with very fine Δx and Δt. Here we use $\rho = \rho_{10^{-4},10}$ (Δt in units of minutes, and Δx in units of μm); that is, at each time t, we compare the simulated solution at larger time step sizes Δt to the simulated solution at the finest time step size to estimate the errors at each time. With these definitions, we calculate the numerical order of convergence by

$$\text{order} \quad = \quad \frac{\log\left(\frac{\text{Err}(\Delta t_1,10)}{\text{Err}(\Delta t_2,10)}\right)}{\log\left(\frac{\Delta t_1}{\Delta t_2}\right)} \tag{43}$$

The convergence results are stated in Table 5 (left) and plotted in Fig. 6 (left). First-order convergence is observed at all computed times.

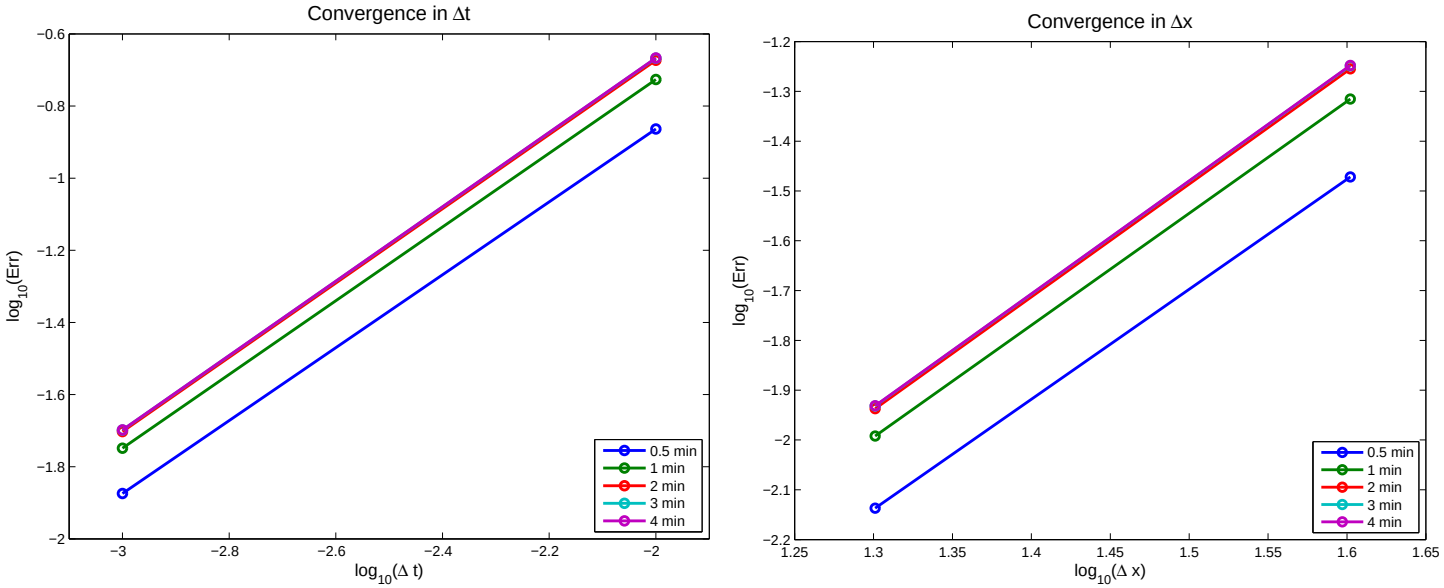

Figure 6) Convergence for Example 2 (Section 4.2).
Left: Convergence in Δt, with $\Delta x = 10\ \mu$m. Each curve gives the error across Δt at a different time (from top to bottom: t=0.5, 1, 2, 3, and 4 minutes). The convergence rates are computed as in Eqn. 43. See Table 5.
Right: Convergence in Δx, with $\Delta t = 10^{-4}$ min. Each curve gives the error across Δx at a different time (from top to bottom: t=0.5, 1, 2, 3, and 4 minutes). The convergence rates are computed as in Eqn. 46. See Table 5.

4.2.2 Convergence in space

To test convergence in space, we set $\Delta t = 10^{-4}$ min and simulated 4 minutes of diffusion at spatial resolutions $\Delta x = 10\ \mu$m (high or fine resolution), 20 μm (medium resolution), and 40 μm (low resolution). As before, we will use $\rho_{10^{-4},10}$ as an estimate of the exact solution. Because the voxel centers do not align for the simulated spatial resolutions, we cannot simply compute the ℓ_∞ norm voxel by voxel as before. Instead, we define a norm to help quantify the errors. First, noting that ρ_i^n is the simulated mean value of ρ in the i^{th} voxel at resolution Δx (hereafter denoted by $\Omega_i^{\Delta x}$), we first coarse grain the solution on the finest mesh to the simulation resolution:

$$\rho^{\text{coarse}}_{\Delta t,\Delta x}(\mathbf{x}_i,t) \quad = \quad \text{mean}\left\{\rho_{\Delta t,10}(\mathbf{x}_k,t) \text{ for } k \text{ satisfying } \Omega_k^{10} \subset \Omega_i^{\Delta x}\right\} \tag{44}$$

For $\Delta x = 20\ \mu$m, this average is computed over 8 voxels from the fine-grained mesh for each voxel in the medium resolution mesh; for $\Delta x = 40\ \mu$m, the average is computed over 64 voxels from the fine-grained mesh for each

Table 5: Convergence for Example 2 (Section 4.2).
Left: Convergence in Δt, with $\Delta x = 10\ \mu$m. For all times, we observe approximately first-order convergence.
Right: Convergence in Δx, with $\Delta t = 10^{-4}$ min. For all times, we observe approximately second-order convergence.

Time (min)	Err(Δt, 10 μm) $\Delta t = 10^{-2}$ min	$\Delta t = 10^{-3}$ min	order
0.5	1.3689e-01	1.3361e-02	1.01
1	1.8778e-01	1.7843e-02	1.02
2	2.1235e-01	1.9826e-02	1.03
3	2.1514e-01	2.0024e-02	1.03
4	2.1543e-01	2.0042e-02	1.03
mean			1.02

Time (min)	Err(10^{-4} min, Δx) $\Delta x = 40\ \mu$m	$\Delta x = 20\ \mu$m	order
0.5	3.3739e-02	7.2974e-03	2.21
1	4.8367e-02	1.0187e-02	2.25
2	5.5621e-02	1.1559e-02	2.27
3	5.6424e-02	1.1704e-02	2.27
4	5.6503e-02	1.1718e-02	2.27
mean			2.25

voxel in the low resolution mesh. With this coarse-grained projection of the fine solution on the medium- and low-resolution meshes, we can define:

$$\text{Err}_{\text{coarse}}(\Delta t, \Delta x) \quad = \quad \left|\left|\rho_{\Delta t,\Delta x} - \rho^{\text{coarse}}_{\Delta t,\Delta x}\right|\right|_{\infty} \tag{45}$$

We then compute the order of convergence via

$$\text{order} = \frac{\log\left(\frac{\text{Err}_{\text{coarse}}(\Delta t,40)}{\text{Err}_{\text{coarse}}(\Delta t,20)}\right)}{\log\left(\frac{40}{20}\right)} \tag{46}$$

The convergence results are stated in Table 5 (right) and plotted in Fig. 6 (right). Approximately second-order convergence is observed at all computed times.

4.2.3 Selecting Δt for reasonable accuracy at $\Delta x = 20\ \mu$m resolution

As before, we next investigated the relative accuracy of solutions at $\Delta x = 20\ \mu$m at a variety of time step sizes. In Fig. 7 and Table 6, we see for the solutions are stable for all tested time steps, and that for $\Delta t = 0.01$ min, the relative error never exceeds 0.6% at any of the tested times, and with $\Delta t = 0.1$ min, the error remains below approximately 5% all tested times. This means that for diffusion-decay problems with parameters of this order of magnitude, we can use step sizes that are 10-100 times larger than the explicit CFL condition (3D: $(20\ \mu\text{m})^2/(2 \times 3 \times 10^5\ \mu\text{m}^2/\text{min}) \sim 6.7 \times 10^{-4}$ min) and still obtain reasonable accuracy.

Table 6: Relative accuracy versus Δt for $\Delta x = 20\ \mu$m in Example 2 (Section 4.2). At this spatial resolution, $\Delta t = 0.01$ or $\Delta t = 0.1$ min may be sufficient for many common problems.

time (min)	Relative Err(Δt, 20 μm), where Δt is: 1 min	0.1 min	10^{-2} min	10^{-3} min
0.5	n.a.	0.02130426	0.003622101	0.000353547
1	0.08972731	0.034661628	0.004994918	0.000474698
2	0.148955005	0.045961999	0.005664208	0.00052887
3	0.195526889	0.048989694	0.00574096	0.000534346
4	0.230851486	0.049768124	0.005749059	0.000534856
mean	1.6627e-01	4.0137e-02	5.1542e-03	4.8526e-04

4.3 Example 3: 3-D diffusion-reaction with bulk sources, grid-aligned cell uptake

Next, we added grid-aligned cells to uptake substrate, as would be expected in cellular automata and cellular potts methods. The modified problem took the form

$$\frac{\partial \rho}{\partial t} \quad = \quad D\nabla^2\rho - \lambda\rho + f(\mathbf{x}) \cdot (\rho^* - \rho) - \sum_k 1_k(\mathbf{x}) U_k \rho \qquad \mathbf{x} \in \Omega \tag{47}$$

$$D\nabla\rho \cdot \mathbf{n} \quad = \quad 0 \qquad \mathbf{x} \in \partial\Omega \tag{48}$$

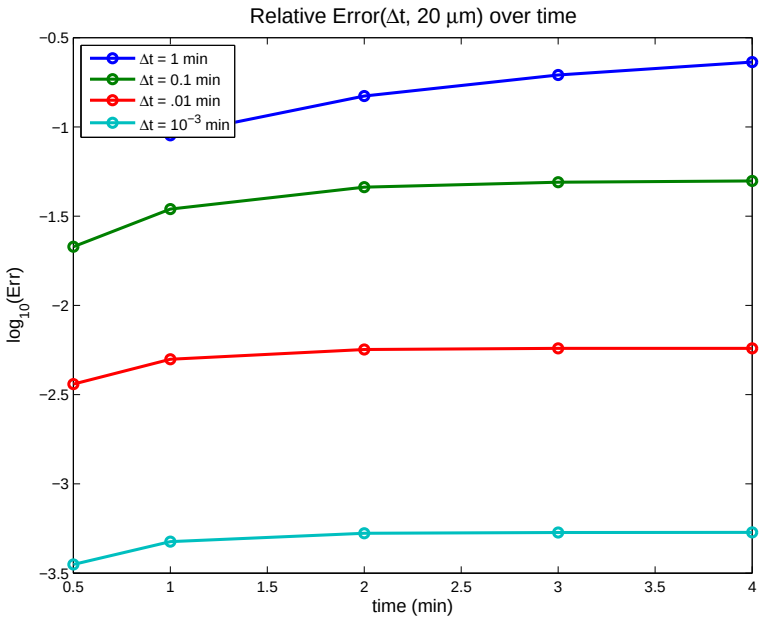

Figure 7) Relative accuracy versus Δt for $\Delta x = 20\ \mu$m in Example 2 (Section 4.2). At this spatial resolution, $\Delta t = 0.01$ or $\Delta t = 0.1$ min may be sufficient for many common problems.

where $\Omega = [-L_0, L_0] \times [-L_0, L_0] \times [-L_0, L_0]$ with $L_0 = 500\ \mu$m, and f is defined by

$$f(\mathbf{x}) \quad = \quad \begin{cases} r & \text{if } ||x| - L_0| \leq 40\ \mu\text{m or } ||y| - L_0| \leq 40\ \mu\text{m or } ||z| - L_0| \leq 40 \\ 0 & \text{elsewhere.} \end{cases} \tag{49}$$

All parameters are as in Example 4.2, and we set $W_k = 10^3 \mu\text{m}^3$ and $\overline{U}_k = 10\ \text{min}^{-1}$ for each cell, so that the diffusion length scale satisifes $L = \sqrt{D/\overline{U}_k} = 100\ \mu$m in areas densely filled with cells [14]. The cells were introduced to fill a spherical tumor focus centered at $(0, 0, 0)$ with radius $400\ \mu$m. That is, we introduced all cells that satisfied

$$\mathbf{x}_n = (x_{i_n}, y_{j_n}, z_{k_n}) \ \text{ with } \ ||\mathbf{x_n}|| \leq 400\ \mu\text{m} \tag{50}$$

and where $x_{i_n}, y_{j_n}, z_{k_n} \in \{-495, -485, ..., 485, 495\}\ \mu$m. (That is, the cells are centered at voxel centers at 10 μm resolution.) A typical solution ($\Delta x = 10\,\mu$m, $\Delta t = 10^{-4}$ min) at $t = 4$ min is plotted in Figure 8. The source code for this problem can be found in the `examples` directory of any BioFVM download (file: `convergence_test3.cpp`).

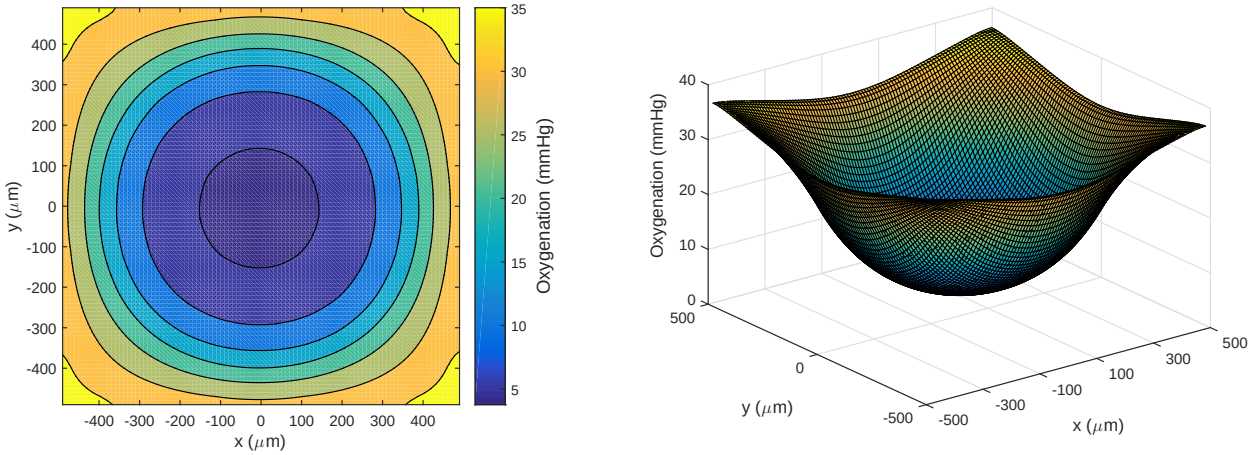

Figure 8) Solution for Example 3 (Section 4.3) after 4 minutes, computed with $\Delta x = 10\ \mu$m and $\Delta t = 10^{-4}$ min. **left:** Contour plot through $z = 5\ \mu$m. **right:** Surface plot through the same cross-section.

4.3.1 Convergence in time

We calculated the convergence rate in time as defined in Section 4.2.1, using $\Delta x = 10\,\mu$m and $\Delta t = 10^{-4}$ min to calculate ρ. The convergence results are stated in Table 7 (left) and plotted in Fig. 9 (left). First-order convergence is observed at all computed times.

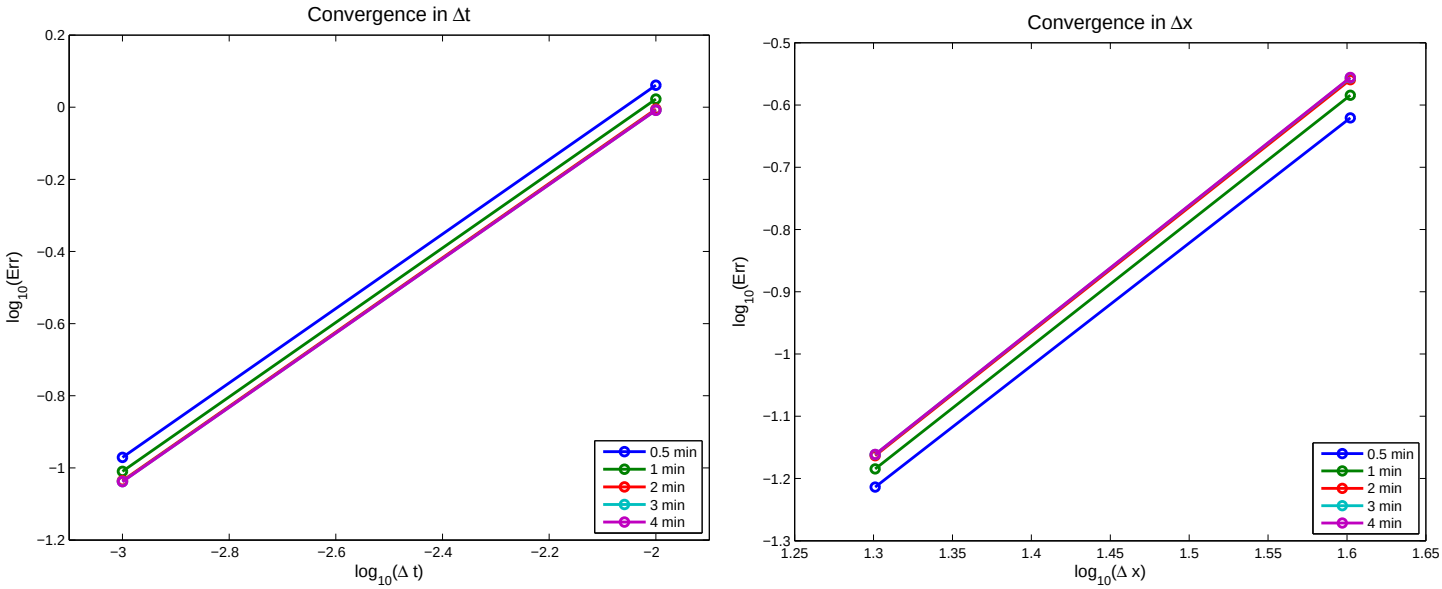

Figure 9) Convergence for Example 3 (Section 4.3).
Left: Convergence in Δt, with $\Delta x = 10\,\mu$m. Each curve gives the error across Δt at a different time (from top to bottom: t=0.5, 1, 2, 3, and 4 minutes). The convergence rates are computed as in Eqn. 43. See Table 7.
Right: Convergence in Δx, with $\Delta t = 10^{-4}$ min. Each curve gives the error across Δx at a different time (from top to bottom: t=0.5, 1, 2, 3, and 4 minutes). The convergence rates are computed as in Eqn. 46. See Table 7.

Table 7: Convergence for Example 3 (Section 4.3).
Left: Convergence in Δt, with $\Delta x = 10\,\mu$m. For all times, we observe approximately first-order convergence.
Right: Convergence in Δx, with $\Delta t = 10^{-4}$ min. For all times, we observe approximately second-order convergence.

Time	Err(Δt, 10 μm)		
(min)	$\Delta t = 10^{-2}$ min	$\Delta t = 10^{-3}$ min	order
0.5	1.150889467	0.106925713	1.03
1	1.053258117	0.097784284	1.03
2	0.987859439	0.092132075	1.03
3	0.981200055	0.091633895	1.03
4	0.980696623	0.091601635	1.03
mean			1.03

Time	Err(10^{-4} min, Δx)		
(min)	$\Delta x = 40\,\mu$m	$\Delta x = 20\,\mu$m	order
0.5	2.3949e-01	6.1133e-02	1.97
1	2.6043e-01	6.5372e-02	1.99
2	2.7586e-01	6.8652e-02	2.01
3	2.7777e-01	6.8959e-02	2.01
4	2.7807e-01	6.8979e-02	2.01
mean			2.00

4.3.2 Convergence in space

To test convergence in space, we set $\Delta t = 10^{-4}$ min and simulated 4 minutes of diffusion at spatial resolutions $\Delta x = 10\mu$m (high or fine resolution), 20μm (medium resolution), and 40μm (low resolution). As before, we will use $\rho_{10^{-4},10}$ as an estimate of the exact solution, with the errors and order of convergence defined as in Section 4.2.2. The convergence results are stated in Table 7 (right) and plotted in Fig. 9 (right). Approximately second-order convergence is observed at all computed times.

4.3.3 Selecting Δt for reasonable accuracy at $\Delta x = 20\,\mu$m resolution

As before, we next investigated the relative accuracy of solutions at $\Delta x = 20\,\mu$m at a variety of time step sizes. In Fig. 10 and Table 8, we see for the solutions are stable for all tested time steps, and that for $\Delta t = 0.01$ min,

the relative error never exceeds 4% at any of the tested times. This means that for diffusion-decay problems with parameters of this order of magnitude, we can use step sizes that are ~ 10 times larger than the explicit CFL condition (3D: $(20\ \mu\text{m})^2/(2 \times 3 \times 10^5\ \mu\text{m}^2/\text{min}) \sim 6.7 \times 10^{-4}$ min) and still obtain reasonable accuracy.

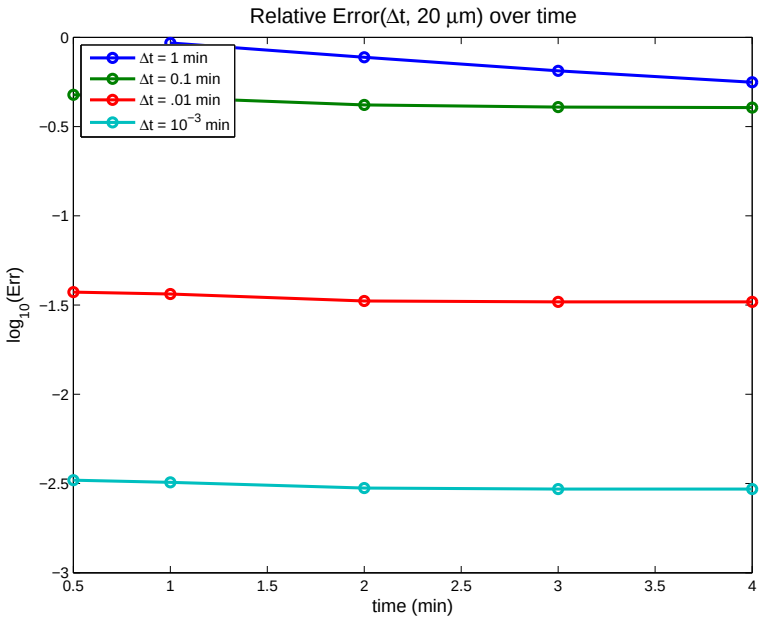

Figure 10) Relative accuracy versus Δt for $\Delta x = 20\ \mu$m in Example 3 (Section 4.3). At this spatial resolution, $\Delta t = 0.01$ or $\Delta t = 0.1$ min may be sufficient for many common problems.

Table 8: Relative accuracy versus Δt for $\Delta x = 20\ \mu$m in Example 3 (Section 4.3). At this spatial resolution, $\Delta t = 0.01$ or $\Delta t = 0.1$ min may be sufficient for many common problems.

time	Relative Err($\Delta t, 20\ \mu$m), where Δt is:			
(min)	1 min	0.1 min	10^{-2} min	10^{-3} min
0.5	n.a.	0.476524012	0.037360947	0.00330152
1	0.929100792	0.458497393	0.036475153	0.003209583
2	0.773244293	0.418323023	0.033328745	0.00298451
3	0.64869279	0.406707729	0.032957057	0.002944096
4	0.55996833	0.404075957	0.032929183	0.00294212
mean	0.72	0.433	0.0346	0.00308

4.4 Example 4: 3-D diffusion-reaction with bulk sources, off-lattice cell uptake

Next, we tested off-lattice cells to uptake substrate, as would be expected in cell-centered agent-based models. The modified problem took the form

$$\frac{\partial \rho}{\partial t} = D\nabla^2\rho - \lambda\rho + f(\mathbf{x}) \cdot (\rho^* - \rho) - \sum_k 1_k(\mathbf{x}) U_k \rho \qquad \mathbf{x} \in \Omega \tag{51}$$

$$D\nabla\rho \cdot \mathbf{n} = 0 \qquad \mathbf{x} \in \partial\Omega \tag{52}$$

where $\Omega = [-L_0, L_0] \times [-L_0, L_0] \times [-L_0, L_0]$ with $L_0 = 500\ \mu$m, and f is defined by

$$f(\mathbf{x}) = \begin{cases} r & \text{if } ||x| - L_0| \le 40\ \mu\text{m or } ||y| - L_0| \le 40\ \mu\text{m or } ||z| - L_0| \le 40 \\ 0 & \text{elsewhere.} \end{cases} \tag{53}$$

All parameters are as in Example 4.2. 417,000 cells with radius 5 μm (volume: $W_k = \frac{4}{3}\pi 5^3\ \mu\text{m}^3$) were introduced with a hexagonal arrangement to fill the same spherical tumor focus centered at $(0, 0, 0)$ with radius 400 μm. See

the source code for further details on this cell packing. Each cell had the same uptake rate as in Example 3 (Section 4.3). A typical solution ($\Delta x = 10\ \mu$m, $\Delta t = 10^{-4}$ min) at $t = 4$ min is plotted in Figure 11. The source code for this problem can be found in the `examples` directory of any BioFVM download (file: `convergence_test4.cpp`).

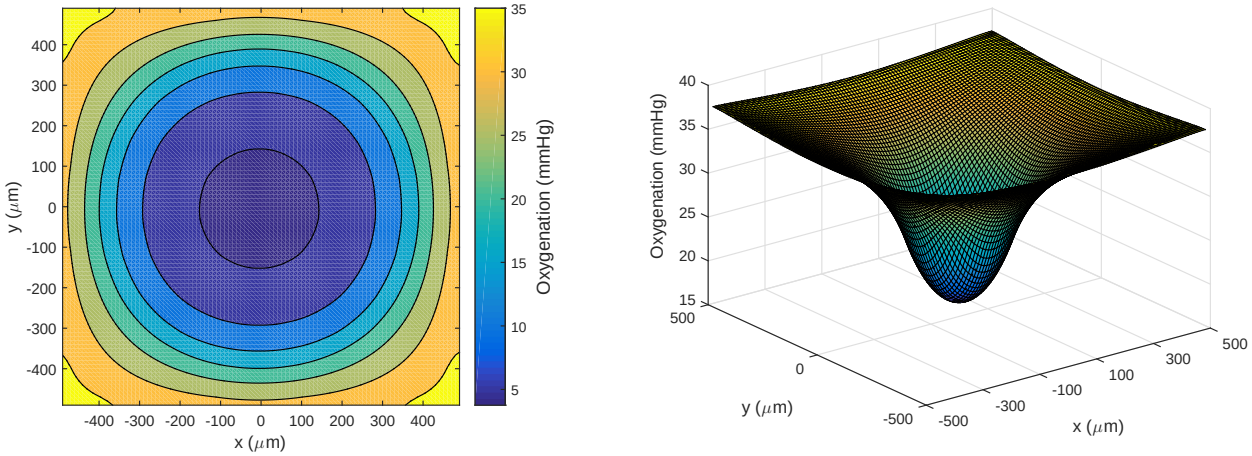

Figure 11) Solution for Example 4 (Section 4.4) after 4 minutes, computed with $\Delta x = 10\ \mu$m and $\Delta t = 10^{-4}$ min. **left:** Contour plot through $z = 5\ \mu$m. **right:** Surface plot through the same cross-section.

4.4.1 Convergence in time

We calculated the convergence rate in time as defined in Section 4.2.1, using $\Delta x = 10\ \mu$m and $\Delta t = 10^{-4}$ min to calculate ρ. The convergence results are stated in Table 9 (left) and plotted in Fig. 12 (left). First-order convergence is observed at all computed times.

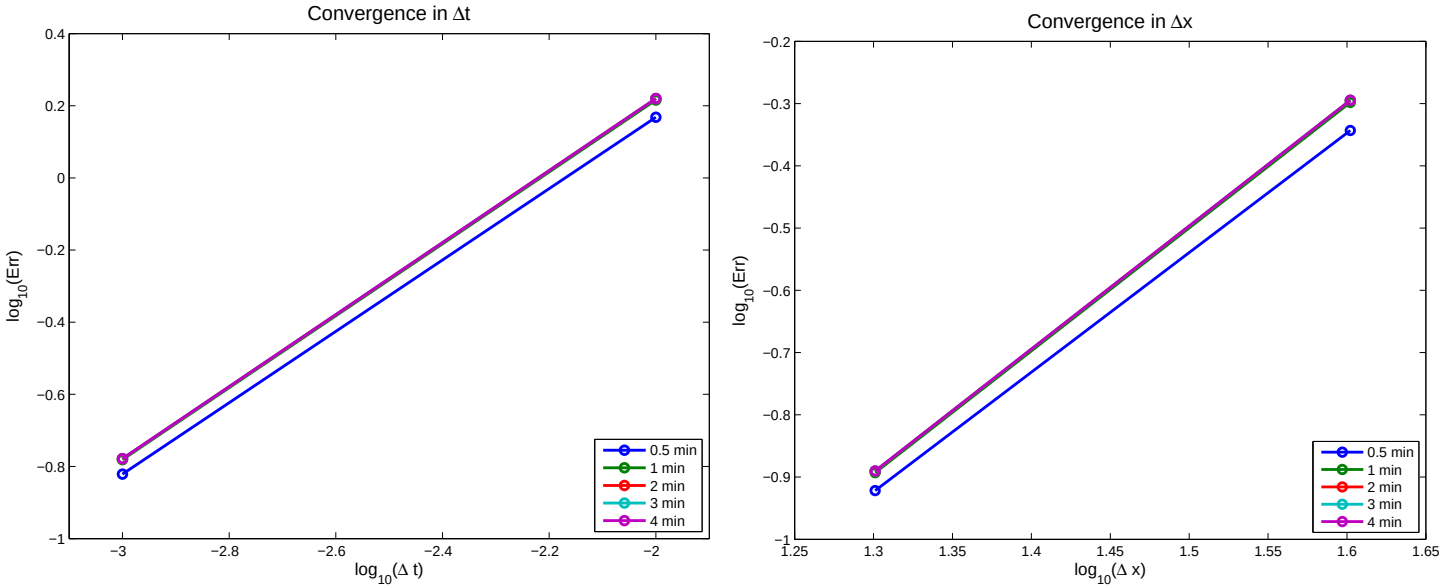

Figure 12) Left: Convergence in Δt for Example 4 (Section 4.4), with $\Delta x = 10\ \mu$m. Each curve gives the error across Δt at a different time (from top to bottom: t=0.5, 1, 2, 3, and 4 minutes). The convergence rates are computed as in Eqn. 43. See Table 9.

Right: Convergence in Δx for Example 4 (Section 4.4), with $\Delta t = 10^{-4}$ min. Each curve gives the error across Δx at a different time (from top to bottom: t=0.5, 1, 2, 3, and 4 minutes). The convergence rates are computed as in Eqn. 46. See Table 9.

4.4.2 Convergence in space

To test convergence in space, we set $\Delta t = 10^{-4}$ min and simulated 4 minutes of diffusion at spatial resolutions $\Delta x = 10\mu$m (high or fine resolution), 20μm (medium resolution), and 40μm (low resolution). As before, we will use

Table 9: Convergence for Example 4 (Section 4.4).
Left: Convergence in Δt, with $\Delta x = 10\,\mu$m. For all times, we observe approximately first-order convergence.
Right: Convergence in Δx, with $\Delta t = 10^{-4}$ min. For all times, we observe approximately second-order convergence.

Time (min)	Err(Δt, 10 μm) $\Delta t = 10^{-2}$ min	$\Delta t = 10^{-3}$ min	order
0.5	1.472985091	0.150945545	0.989378051
1	1.643977639	0.165465049	0.997189635
2	1.659215736	0.1665307	0.998408552
3	1.659289769	0.166535021	0.998416659
4	1.659289104	0.166534098	0.998418892
mean			0.996362358

Time (min)	Err(10^{-4} min, Δx) $\Delta x = 40\,\mu$m	$\Delta x = 20\,\mu$m	order
0.5	4.5359e-01	1.1975e-01	1.92
1	5.0324e-01	1.2798e-01	1.98
2	5.0722e-01	1.2873e-01	1.98
3	5.0724e-01	1.2873e-01	1.98
4	5.0723e-01	1.2873e-01	1.98
mean			1.97

$\rho_{10^{-4},10}$ as an estimate of the exact solution, with the errors and order of convergence defined as in Section 4.2.2. The convergence results are stated in Table 9 (right) and plotted in Fig. 12 (right). Approximately second-order convergence is observed at all computed times.

4.4.3 Selecting Δt for reasonable accuracy at $\Delta x = 20\,\mu$m resolution

As before, we next investigated the relative accuracy of solutions at $\Delta x = 20\,\mu$m at a variety of time step sizes. In Fig. 13 and Table 10, we see for the solutions are stable for all tested time steps, and that for $\Delta t = 0.01$ min, the relative error never exceeds $\sim 5\%$ at any of the tested times. This means that for diffusion-decay problems with parameters of this order of magnitude, we can use step sizes that are approximately 15 times larger than the explicit CFL condition (3D: $(20\,\mu\text{m})^2/(2 \times 3 \times 10^5\,\mu\text{m}^2/\text{min}) \sim 6.7 \times 10^{-4}$ min) and still obtain reasonable accuracy.

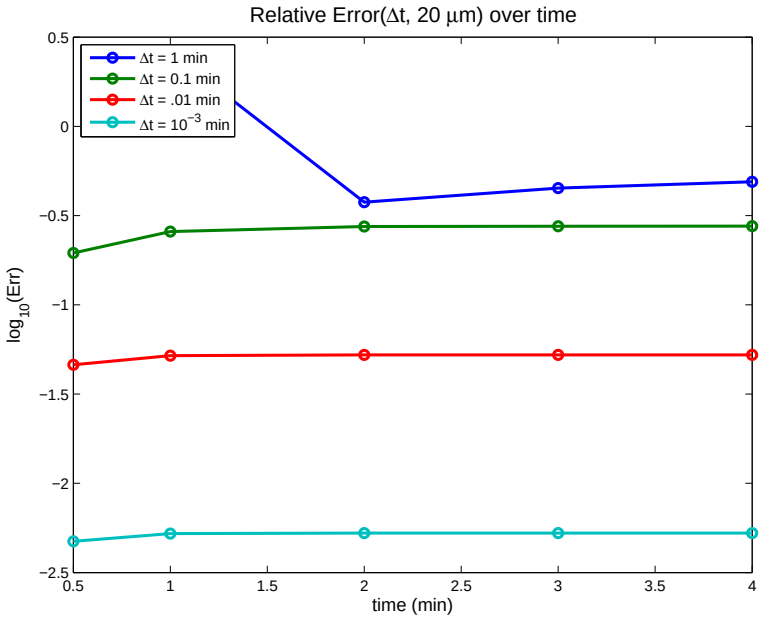

Figure 13) Relative accuracy versus Δt for $\Delta x = 20\,\mu$m in Example 4 (Section 4.4). At this spatial resolution, $\Delta t = 0.01$ min may be sufficient for many common problems.

Table 10: Relative accuracy versus Δt for $\Delta x = 20\,\mu$m in Example 4 (Section 4.4). At this spatial resolution, $\Delta t = 0.01$ min may be sufficient for many common problems.

time	Relative Err(Δt, 20 μm), where Δt is:			
(min)	1 min	0.1 min	10^{-2} min	10^{-3} min
0.5	n.a.	0.195251358	0.046145886	0.004730137
1	2.608209824	0.257376363	0.051954152	0.005230567
2	0.375719263	0.274877019	0.052455745	0.005266263
3	0.451076525	0.275756818	0.05245812	0.005266404
4	0.489548456	0.27580093	0.052458101	0.005266375
mean	0.981138517	0.255812498	0.051094401	0.005151949

4.5 Example 5: 3-D diffusion-reaction with bulk sources, off-lattice cell uptake and supply

Next, we tested off-lattice cells to uptake substrate, as would be expected in cell-centered agent-based models. The modified problem took the form

$$\frac{\partial \rho}{\partial t} = D\nabla^2\rho - \lambda\rho + f(\mathbf{x})\cdot(\rho^* - \rho) - \sum_{k\in\mathcal{K}_1} 1_k(\mathbf{x})\overline{U}_k\rho + \sum_{k\in\mathcal{K}_2} 1_k(\mathbf{x})\overline{S}_k\,(\rho_k^* - \rho) \qquad \mathbf{x}\in\Omega \tag{54}$$

$$D\nabla\rho\cdot\mathbf{n} = 0 \qquad \mathbf{x}\in\partial\Omega \tag{55}$$

where $\Omega = [-L_0, L_0]\times[-L_0, L_0]\times[-L_0, L_0]$ with $L_0 = 500\,\mu$m, and f is defined by

$$f(\mathbf{x}) = \begin{cases} r & \text{if } ||x| - L_0| \le 40\,\mu\text{m or } ||y| - L_0| \le 40\,\mu\text{m or } ||z| - L_0| \le 40 \\ 0 & \text{elsewhere.} \end{cases} \tag{56}$$

All parameters are as in Example 4.2. $\mathcal{K}_1$ is the set of 417,000 cells uptaking ρ within the 400 μm tumor focus introduced in Example 4 (Section 4.4), and $\mathcal{K}_2$ is a set of 200 randomly-placed cell-centered sources, with $\mathcal{S}_k = 10$ min^{-1}, $\rho^* = 38$ mmHg, and same size as in Example 4. A typical solution ($\Delta x = 10\,\mu$m, $\Delta t = 10^{-4}$ min) at $t = 4$ min is plotted in Figure 14. The source code for this problem can be found in the `examples` directory of any BioFVM download (file: `convergence_test5.cpp`).

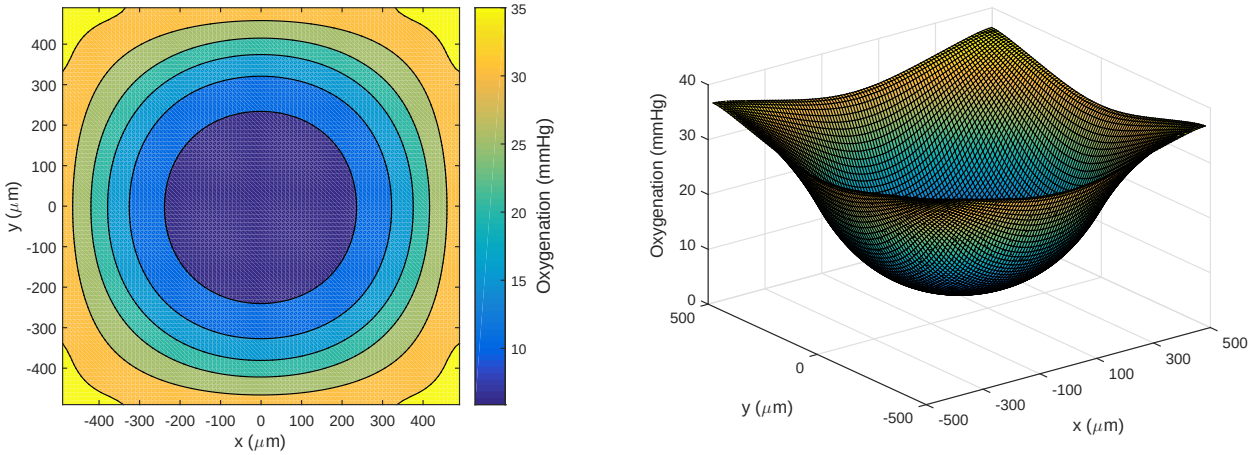

Figure 14) Solution for Example 5 (Section 4.5) after 4 minutes, computed with $\Delta x = 10\,\mu$m and $\Delta t = 10^{-4}$ min. **left:** Contour plot through $z = 5\,\mu$m. **right:** Surface plot through the same cross-section.

4.5.1 Convergence in time

We calculated the convergence rate in time as defined in Section 4.2.1, using $\Delta x = 10\,\mu$m and $\Delta t = 10^{-4}$ min to calculate ρ. The convergence results are stated in Table 11 (left) and plotted in Fig. 15 (left). First-order convergence is observed at all computed times.

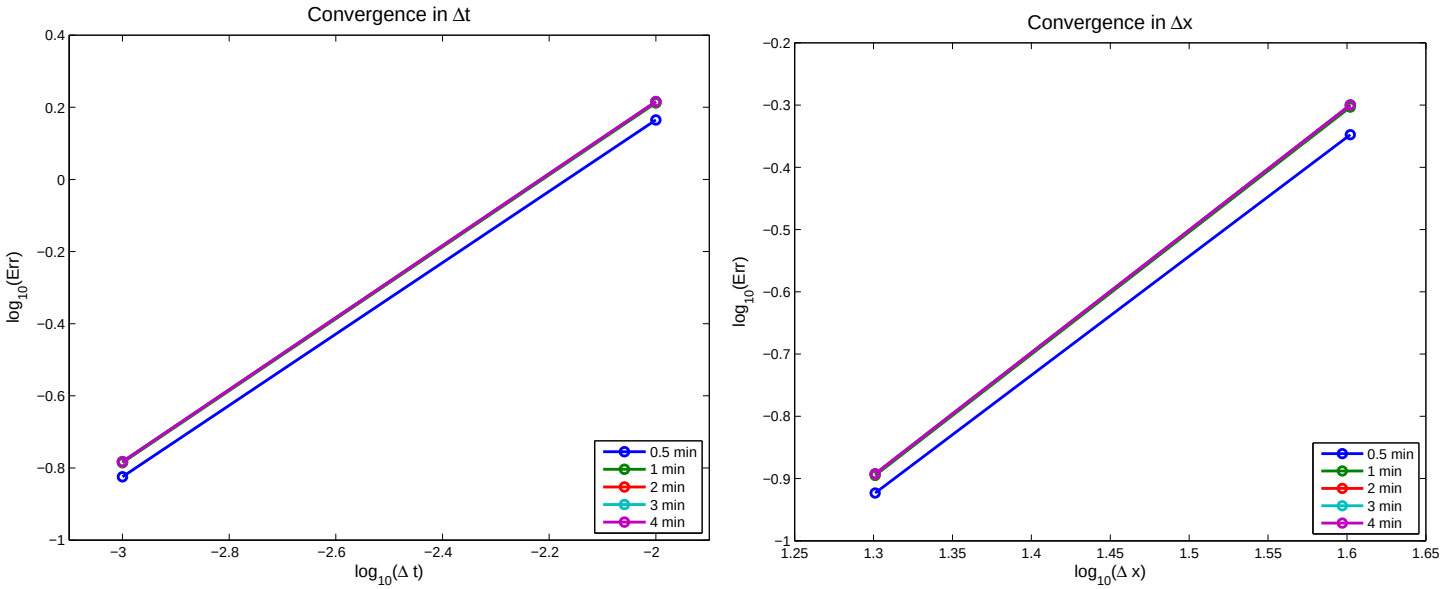

Figure 15) Left: Convergence in Δt for Example 5 (Section 4.5), with $\Delta x = 10\,\mu$m. Each curve gives the error across Δt at a different time (from top to bottom: t=0.5, 1, 2, 3, and 4 minutes). The convergence rates are computed as in Eqn. 43. See Table 11.
Right: Convergence in Δx for Example 4 (Section 4.5), with $\Delta t = 10^{-4}$ min. Each curve gives the error across Δx at a different time (from top to bottom: t=0.5, 1, 2, 3, and 4 minutes). The convergence rates are computed as in Eqn. 46. See Table 11.

Table 11: Convergence for Example 5 (Section 4.5).
Left: Convergence in Δt, with $\Delta x = 10\,\mu$m. For all times, we observe approximately first-order convergence.
Right: Convergence in Δx, with $\Delta t = 10^{-4}$ min. For all times, we observe approximately second-order convergence.

Time	Err(Δt, 10 μm)			Time	Err(10^{-4} min, Δx)		
(min)	$\Delta t = 10^{-2}$ min	$\Delta t = 10^{-3}$ min	order	(min)	$\Delta x = 40\,\mu$m	$\Delta x = 20\,\mu$m	order
0.5	1.46213217	0.149817435	0.989	0.5	4.4926e-01	1.1934e-01	1.91
1	1.628579376	0.163924374	0.997	1	4.9747e-01	1.2737e-01	1.97
2	1.643088104	0.164951938	0.998	2	5.0128e-01	1.2803e-01	1.97
3	1.643134677	0.164970988	0.998	3	5.0130e-01	1.2811e-01	1.97
4	1.643229553	0.164945039	0.998	4	5.0129e-01	1.2812e-01	1.97
mean			0.996303533	mean			1.96

4.5.2 Convergence in space

To test convergence in space, we set $\Delta t = 10^{-4}$ min and simulated 4 minutes of diffusion at spatial resolutions $\Delta x = 10\,\mu$m (high or fine resolution), 20 μm (medium resolution), and 40 μm (low resolution). As before, we will use $\rho_{10^{-4},10}$ as an estimate of the exact solution, with the errors and order of convergence defined as in Section 4.2.2. The convergence results are stated in Table 11 (right) and plotted in Fig. 15 (right). Approximately second-order convergence is observed at all computed times.

4.5.3 Selecting Δt for reasonable accuracy at $\Delta x = 20\,\mu$m resolution

As before, we next investigated the relative accuracy of solutions at $\Delta x = 20\,\mu$m at a variety of time step sizes. In Fig. 16 and Table 12, we see for the solutions are stable for all tested time steps, and that for $\Delta t = 0.01$ min, the relative error never exceeds $\sim 5\%$ at any of the tested times. This means that for diffusion-decay problems with parameters of this order of magnitude, we can use step sizes that are approximately 15 times larger than the explicit CFL condition (3D: $(20\,\mu\text{m})^2/(2 \times 3 \times 10^5\,\mu\text{m}^2/\text{min}) \sim 6.7 \times 10^{-4}$ min) and still obtain reasonable accuracy.

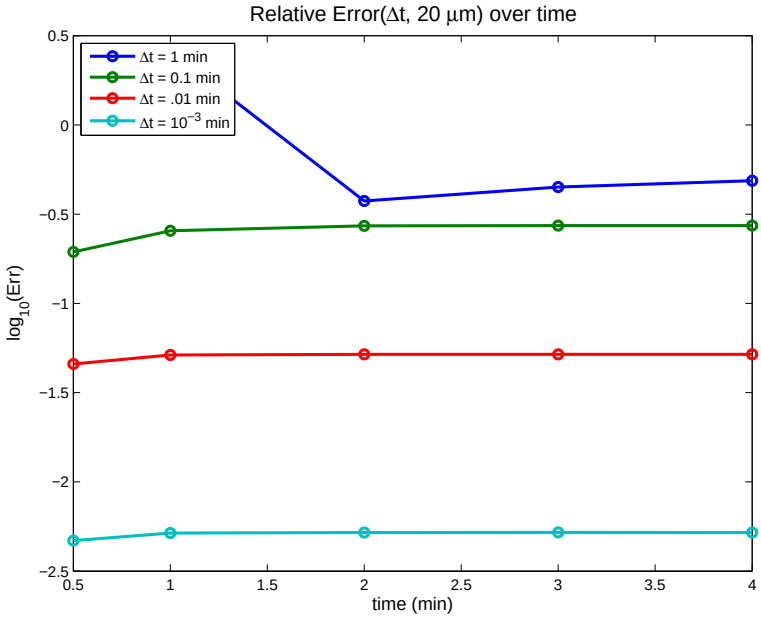

Figure 16) Relative accuracy versus Δt for $\Delta x = 20\,\mu$m in Example 5 (Section 4.5). At this spatial resolution, $\Delta t = 0.01$ min may be sufficient for many common problems.

Table 12: Relative accuracy versus Δt for $\Delta x = 20\,\mu$m in Example 4 (Section 4.4). At this spatial resolution, $\Delta t = 0.01$ min may be sufficient for many common problems.

time	Relative Err($\Delta t, 20\,\mu$m), where Δt is:			
(min)	1 min	0.1 min	10^{-2} min	10^{-3} min
0.5	n.a.	0.194368401	0.045734272	0.004687355
1	2.59024396	0.255074619	0.051374281	0.00517237
2	0.374705709	0.271920769	0.051850793	0.005206683
3	0.448872584	0.272663193	0.05185227	0.005207285
4	0.486527376	0.272744359	0.051855273	0.005206467
mean	0.975087407	0.253354268	0.050533378	0.005096032

5 Performance testing

We tested the performance and scalability of BioFVM with respect to the number of simulated substrates (Sec. 5.1), the number of voxels (scales with the simulated domain size and/or the spatial resolutions; Sec. 5.2), and the number of cells releasing and/or consuming substrates (Sec. 5.3). The computational cost (measured as total wall time for a fixed problem) was found to scale linearly with each of these. All tests were performed on a desktop workstation with a quad-core Intel i7-4790 processor (3.60 GHz) and 16 GB of memory, using MinGW-W64 (gcc version 4.9.1) on Windows 7. We used the compiler flags:

```
-march=core-avx2 -O3 -s -fomit-frame-pointer -mfpmath=both -fopenmp -m64 -std=c++11
```

5.1 Performance scaling with number of substrates

We first tested the computational impact of increasing the number of simulated substrates, by solving

$$\frac{\partial \boldsymbol{\rho}}{\partial t} = \boldsymbol{D} \circ \nabla^2 \boldsymbol{\rho} - \boldsymbol{\lambda} \circ \boldsymbol{\rho} + \mathbf{f} \text{ in } \Omega \tag{57}$$

$$\frac{\partial \boldsymbol{\rho}}{\partial \mathbf{n}} = 0 \text{ on } \partial\Omega, \tag{58}$$

for 2 simulated minutes ($\Delta t = 0.01$ min) on the 1 mm^3 domain ($\Delta x = 10\mu$m, giving 1 million computational voxel) from Example 2 (Section 4.2), with $N \in \{1, 2, 4, 8, 16, 32, 64, 128\}$ simulated substrates. We set $\lambda_i = 0.1$ min^{-1} for each i, we used the same source $\mathbf{f}$ as in Section 4.2, and we randomly chose $D_i \sim 10^5$ mm^2/min for each i.

See Fig. 17. The computational cost–the wall time for the 2 minute diffusion simulation–scaled linearly with the number of simulated substrates (N), with linear least squares fit given by

$$\text{Time (sec)} \approx 5.4704 + 1.1620 N_{\text{substrates}}. \tag{59}$$

Increasing from 1 substrate to 10 substrates increases computational time a factor of approximately 2.6. The source code for this problem can be found in the `examples` directory of any BioFVM download (file: `performance_test_substrates.cpp`).

Notice that this says that approximately 8-9 minutes would be required to simulate 1 hour of diffusion by 10 substrates on a 1-million voxel mesh, and 3-4 minutes to simulate a single substrate.

Number of substrates	Wall time (sec)
1	7.251
2	9.496
4	10.971
8	15.615
16	25.103
32	40.85
64	72.912
128	157.888

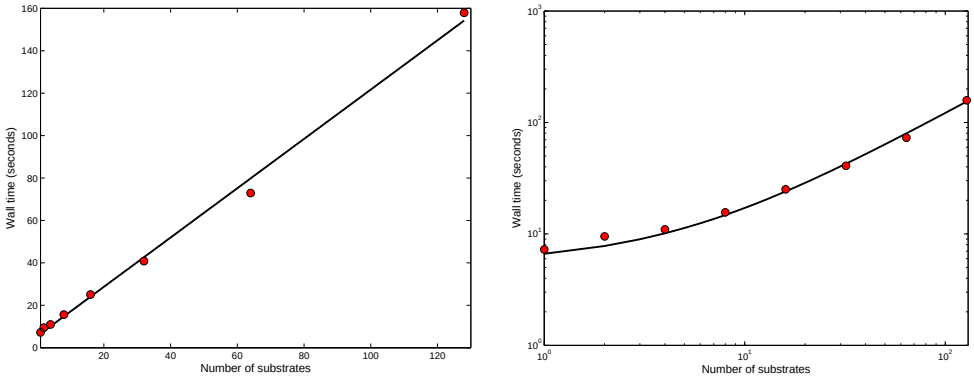

Figure 17) Computational cost scales linearly with the number of simulated substrates (Section 5.1). **Left:** Time to simulate 2 minutes of diffusion for N substrates, with 1 million voxels and $\Delta t = 0.01$ min. **Middle:** Plot of the computation times (red circles) and the linear least squares fit (black line). **Right:** Plot of the computation times (red circles) and the linear least squares fit (black curve), logarithmic scale.

5.2 Performance scaling with number of voxels

We next tested the computational impact of increasing the number of voxels, by solving Eqn. 57 for 2 simulated minutes ($\Delta t = 0.01$ min) on increasingly large domains (from 8,000 to over 4 million voxels) with Δx fixed at 10 μm; D and λ were as in the previous test, and f was changed to a bulk source throughout the region

$$f = \begin{cases} 1 & \text{if } |x| < 5 \text{ or } |y| < 5 \text{ or } |z| < 5 \\ 0 & \text{otherwise.} \end{cases} \tag{60}$$

See Fig. 18. The computational cost scaled linearly with the number of voxels, with linear least squares fit

$$\text{Time (sec)} \approx 0.05727 + 6.2 \times 10^{-6} N_{\text{voxels}}. \tag{61}$$

Notice that using $\Delta t = 0.1$ min would decrease these computational times by a factor of 10. The source code for this problem can be found in the `examples` directory of any BioFVM download (file: `performance_test_voxels.cpp`).

5.3 Performance scaling with number of cells (uptake/source terms)

We next tested the computational impact of increasing the number of cells, by solving Eqn. 57 for 2 simulated minutes ($\Delta t = 0.01$ min) on the same domain as before, but at a resolution of $\Delta x = 20$ μm (125,000 voxels) tailored to the 20-μm diameter cell size. We simulated an increasing number of cells (1,000 to over 4 million); D, λ and f were as in the previous tests. See Fig. 19. The computational cost scaled linearly with the number of voxels, with linear least squares fit given by

$$\text{Time (sec)} \approx 1.15 + 4.1 \times 10^{-6} N_{\text{cells}}. \tag{62}$$

Number of voxels	Wall time (sec)
8000	0.148
27000	0.243
64000	0.415
125000	0.737
216000	1.37
343000	2.263
512000	3.254
729000	4.592
1000000	6.26
1331000	8.321
1728000	10.729
2197000	13.913
2744000	17.113
3375000	20.458
4096000	25.776

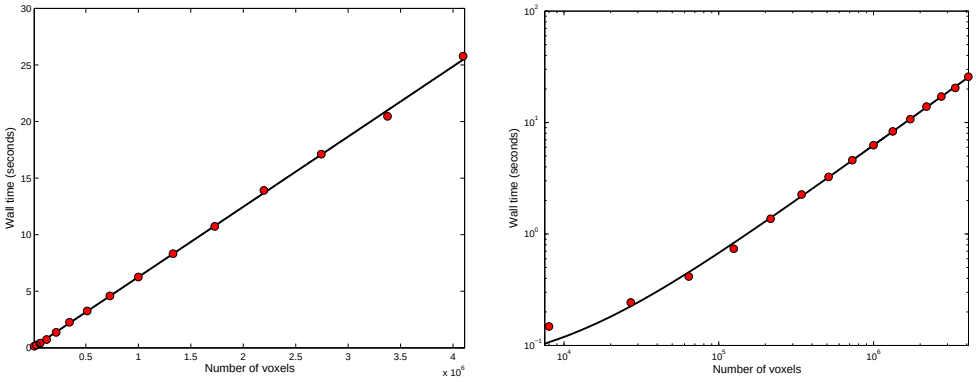

Figure 18) Computational cost scales linearly with the number of simulated voxels (Section 5.1). **Left:** Time to simulate 2 minutes of diffusion for N voxels, with 1 substrate and $\Delta t = 0.01$ min. **Middle:** Plot of the computation times (red circles) and the linear least squares fit (black line). **Right:** Plot of the computation times (red circles) and the linear least squares fit (black curve), logarithmic scale.

A closer examination of Fig. 19 shows that the cells' contribution to the overall computational cost was negligible through approximately 100,000 to 150,000 cells. Notice that using $\Delta t = 0.1$ min would decrease these computational times by a factor of 10. The source code for this problem can be found in the `examples` directory of any BioFVM download (file: `performance_test_numcells.cpp`).

Number of cells	Wall time (sec)
1024	1.015
2048	1.014
4096	1.058
8192	1.053
16384	1.06
32768	1.093
65536	1.407
131072	1.618
262144	2.287
524288	3.489
1048576	5.978
2097152	10.69
4194304	18.024

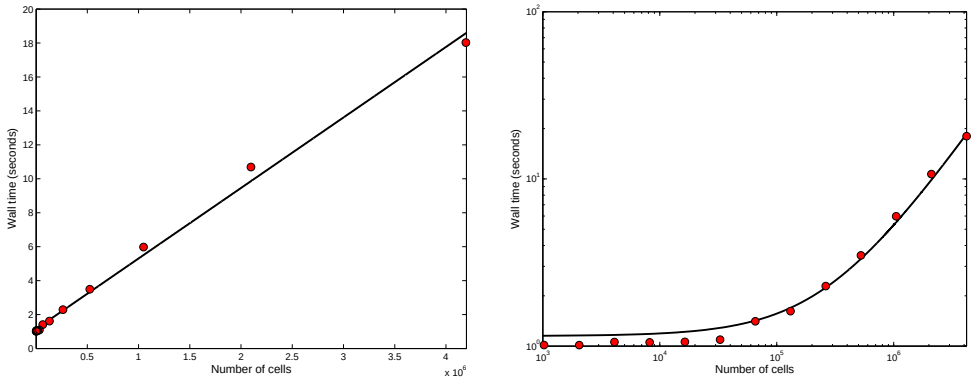

Figure 19) Computational cost scales linearly with the number of simulated cells (Section 5.3). **Left:** Time to simulate 2 minutes of diffusion for N cells, with 1 substrate, 1 million voxels, and $\Delta t = 0.01$ min. **Middle:** Plot of the computation times (red circles) and the linear least squares fit (black line). **Right:** Plot of the computation times (red circles) and the linear least squares fit (black curve), logarithmic scale.

6 References

[1] Aude Carreau et al. Why is the partial oxygen pressure of human tissues a crucial parameter? Small molecules and hypoxia. *Journal of cellular and molecular medicine*, 15(6):1239–1253, June 2011. ISSN 1582-4934. doi: 10.1111/j.1582-4934.2011.01258.x. URL http://dx.doi.org/10.1111/j.1582-4934.2011.01258.x.

[2] Robert Eymard et al. Finite volume methods. In *Solution of Equation in $\mathbb{R}^n$ (Part 3), Techniques of Scientific Computing (Part 3)*, volume 7 of *Handbook of Numerical Analysis*, pages 713–1018. Elsevier, 2000. doi: http://dx.doi.org/10.1016/S1570-8659(00)07005-8. URL http://www.sciencedirect.com/science/article/pii/S1570865900070058.

[3] Hermann B. Frieboes et al. Three-dimensional multispecies nonlinear tumor growthII: Tumor invasion and angiogenesis. *Journal of Theoretical Biology*, 264(4):1254–1278, June 2010. ISSN 00225193. doi: 10.1016/j.jtbi.2010.02.036. URL http://dx.doi.org/10.1016/j.jtbi.2010.02.036.

[4] Hermann B. Frieboes et al. An Integrated Computational/Experimental Model of Lymphoma Growth. *PLoS Comput Biol*, 9(3):e1003008+, March 2013. doi: 10.1371/journal.pcbi.1003008. URL http://dx.doi.org/10.1371/journal.pcbi.1003008.

[5] David R. Grimes et al. A method for estimating the oxygen consumption rate in multicellular tumour spheroids. *Journal of The Royal Society Interface*, 11(92):20131124+, March 2014. ISSN 1742-5662. doi: 10.1098/rsif.2013.1124. URL http://dx.doi.org/10.1098/rsif.2013.1124.

[6] David R. Grimes et al. Oxygen consumption dynamics in steady-state tumour models. *Royal Society Open Science*, 1(1):140080+, September 2014. ISSN 2054-5703. doi: 10.1098/rsos.140080. URL http://dx.doi.org/10.1098/rsos.140080.

[7] K. Groebe and P. Vaupel. Evaluation of oxygen diffusion distances in human breast cancer xenografts using tumor-specific in vivo data: Role of various mechanisms in the development of tumor hypoxia. *International Journal of Radiation Oncology*Biology*Physics*, 15(3):691–697, September 1988. ISSN 03603016. doi: 10.1016/0360-3016(88)90313-6. URL http://dx.doi.org/10.1016/0360-3016(88)90313-6.

[8] J. Grote et al. Oxygen diffusivity in tumor tissue (DS-Carcinosarcoma) under temperature conditions within the range of 20-40°C. *Pflügers Archiv*, 372(1):37–42, 1977. ISSN 0031-6768. doi: 10.1007/BF00582204. URL http://dx.doi.org/10.1007/BF00582204.

[9] Arseny Kapoulkine. pugixml: Light-weight, simple and fast XML parser for C++ with XPath support, 2015. URL http://pugixml.org/.

[10] Paul Macklin. Biological background. In Vittorio Cristini and John S Lowengrub, editors, *Multiscale Modeling of Cancer: An Integrated Experimental and Mathematical Modeling Approach*, chapter 2, pages 2–23. Cambridge University Press, Cambridge, UK, 2010.

[11] Paul Macklin and John S. Lowengrub. Evolving interfaces via gradients of geometry-dependent interior poisson problems: application to tumor growth. *J. Comput. Phys.*, 203(1):191–220, 2005. doi: 10.1016/j.jcp.2004.08.010. URL http://dx.doi.org/10.1016/j.jcp.2004.08.010.

[12] Paul Macklin and John S. Lowengrub. An improved geometry-aware curvature discretization for level set methods: application to tumor growth. *J. Comput. Phys.*, 215(2):392–401, 2006. doi: 10.1016/j.jcp.2005.11.016. URL http://dx.doi.org/10.1016/j.jcp.2005.11.016.

[13] Paul Macklin and John S. Lowengrub. A new ghost cell/level set method for moving boundary problems: Application to tumor growth. *J. Sci. Comput.*, 35(2-3):266–299, 2008. doi: 10.1007/s10915-008-9190-z. URL http://dx.doi.org/10.1007/s10915-008-9190-z. (invited author: J.S. Lowengrub).

[14] Paul Macklin et al. Patient-calibrated agent-based modelling of ductal carcinoma in situ (dcis): From microscopic measurements to macroscopic predictions of clinical progression. *J. Theor. Biol.*, 301:122–40, 2012. doi: 10.1016/j.jtbi.2012.02.002. URL http://dx.doi.org/10.1016/j.jtbi.2012.02.002.

[15] G.I. Marchuk. Splitting and alternating direction methods. volume 1 of *Handbook of Numerical Analysis*, pages 197–462. Elsevier, 1990. doi: http://dx.doi.org/10.1016/S1570-8659(05)80035-3. URL http://www.sciencedirect.com/science/article/pii/S1570865905800353.

[16] S R McKeown. Defining normoxia, physoxia and hypoxia in tumours–implications for treatment response. *Br. J. Radiol.*, 87(1035):20130676, 2014. doi: 10.1259/bjr.20130676.

[17] mingw-w64 Project. MinGW-w64: GCC for Windows 64 & 32 bits, 2014. URL http://mingw-w64.org/.

[18] Peggy L. Olive et al. Measurement of oxygen diffusion distance in tumor cubes using a fluorescent hypoxia probe. *International Journal of Radiation Oncology*Biology*Physics*, 22(3):397–402, January 1992. ISSN 03603016. doi: 10.1016/0360-3016(92)90840-e. URL http://dx.doi.org/10.1016/0360-3016(92)90840-e.

[19] Gibin G. Powathil et al. Modelling the effects of cell-cycle heterogeneity on the response of a solid tumour to chemotherapy: Biological insights from a hybrid multiscale cellular automaton model. *Journal of Theoretical Biology*, 308:1–19, September 2012. ISSN 00225193. doi: 10.1016/j.jtbi.2012.05.015. URL http://dx.doi.org/10.1016/j.jtbi.2012.05.015.

[20] Mark Robertson-Tessi et al. Impact of Metabolic Heterogeneity on Tumor Growth, Invasion, and Treatment Outcomes. *Cancer Research*, 75(8):1567–1579, April 2015. ISSN 1538-7445. doi: 10.1158/0008-5472.can-14-1428. URL http://dx.doi.org/10.1158/0008-5472.can-14-1428.

[21] Llewellyn H. Thomas. Elliptic Problems in Linear Difference Equations over a Network. In *Watson Sci. Comput. Lab Report*. Columbia University, New York, 1949.

[22] R. H. Thomlinson and L. H. Gray. The Histological Structure of Some Human Lung Cancers and the Possible Implications for Radiotherapy. *British Journal of Cancer*, 9(4):539–549, December 1955. ISSN 0007-0920. doi: 10.1038/bjc.1955.55. URL http://dx.doi.org/10.1038/bjc.1955.55.

[23] Peter Vaupel. Hypoxia and aggressive tumor phenotype: Implications for therapy and prognosis. *The Oncologist*, 13(suppl 3):21–26, 2008. doi: 10.1634/theoncologist.13-S3-21. URL http://theoncologist.alphamedpress.org/content/13/suppl_3/21.abstract.

[24] N.N. Yanenko. Simple schemes in fractional steps for the integration of parabolic equations. In Maurice Holt, editor, *The Method of Fractional Steps*, pages 17–41. Springer Berlin Heidelberg, 1971. ISBN 978-3-642-65110-6. doi: 10.1007/978-3-642-65108-3_2. URL http://dx.doi.org/10.1007/978-3-642-65108-3_2.