Biofilm.jl: A fast solver for one-dimensional biofilm chemistry and ecology

Mark Owkes, Kai Coblentz, Austen Eriksson, Takumi Kammerzell, Philip S. Stewart

Biofilm.jl: a fast solver for one-dimensional biofilm chemistry and ecology

Mark Owkes[a,*], Kai Coblentz[a], Austen Eriksson[a], Takumi Kammerzell[a], Philip S. Stewart[b]

[a]*Mechanical & Industrial Engineering, Montana State University*
[b]*Chemical & Biological Engineering, Montana State University*

Abstract

Biofilms are communities of microorganisms that grow on virtually all surfaces with sufficient nutrients including aquatic and industrial water systems and medical devices. Biofilms are complex, structured communities where the interplay of growth, metabolism, and competition between species interact with physical processes of diffusion, convection, attachment, and detachment. This work describes a model of a one-dimensional biofilm in a stirred tank reactor that incorporates these complexities. The model is implemented in the modern Julia programming language providing an efficient tool for studying a large variety of biofilms and the intricate communities the microorganisms create. Details of the new software, known as Biofilm.jl, including the mathematical model and organization and execution of the code, are provided. Examples of biofilms modeled using Biofilm.jl are presented such as a single heterotroph, a sulfide-oxidizing bacteria (SOB) and sulfate-reducing bacteria (SRB), and a phototroph. Postprocessing tools are described that allow a Biofilm.jl user to make plots and extract specific values from the solution and explore the simulated biofilm results.

Keywords: model, software, reaction-diffusion, microorganism, accumulation, Julia

1. Introduction

Biofilms are communities of microorganisms that are attached to a surface [1] and predominate in almost all wet environments with sufficient nutrients [2]. Biofilms can form on a variety of surfaces including, for example, natural

---

[*]Corresponding author.
E-mail address: mark.owkes@montana.edu

aquatic systems, teeth (dental plaque), and water system piping [3]. Furthermore, biofilms are not confined to solid-liquid surfaces and can be found at solid-air or liquid-liquid interfaces [3]. Biofilms have been shown to be associated with many human diseases and grow on a variety of medical devices and implanted biomaterials [4, 5, 6].

Biofilm microorganisms, known as particulates, grow and consume or produce substrates. For example, an aerobe is a particulate that will consume the substrate oxygen when it grows. A biofilm may consist of a single particulate or many particulates. When there are many particulates, interactions with consumed or produced substrates and other physical phenomena can lead to highly structured multispecies communities [7]. Interestingly, these multispecies biofilms have been demonstrated to be profoundly different from planktonic (drifting or floating) cells with increased growth rates, enhanced tolerance to antibacterial agents, and changes in gene expression [3, 2].

Mathematical models have played an important role in the biofilm field because the complexity of these systems - in which biological phenomena such as growth, metabolism, and competition between species interact with physical phenomena such as diffusion, convection, attachment and detachment - leads to non-intuitive emergent properties. These behaviors can be accessed rigorously through computational approaches. Many different model structures from one-dimensional flat slabs to fully three-dimensional particle-based simulations have been derived [8, 9, 10, 11, 12, 13, 14, 15] . Several comprehensive reviews describe and discuss the variety of models in detail [16, 17, 18].

One of the earliest model structures was a one-dimensional version in which the biofilm is treated as a uniformly thick flat slab [19, 20]. Solute transport into and out of the film is assumed to be governed by Fick's Law. The competition for space within the biofilm by different microbial species and abiotic particles is analyzed by differential material balances on each particulate. The dynamics of biofilm thickness are controlled by the net balance of integrated growth and detachment from the top layer of the biofilm.

Though relatively simplistic, these types of models successfully capture such phenomena as: coexistence of aerobes and anaerobes in close proximity within the same biofilm and stratification of these two types of microorganisms [19, 21], activity of biofilm-based wastewater treatment processes [20, 21, 22], slow growth and dormancy in some regions of the biofilm [23, 24, 25], reduced antimicrobial susceptibility in the biofilm mode of growth [23, 26, 24, 25], and plasmid transfer [27] .

Here we recapitulate the classic 1D biofilm model with an updated code and solver. This has the potential to be used as an educational and research tool, replacing outdated code developed at the Montana State University Center for Biofilm Engineering. This legacy code, now three decades old, is relatively slow and is not easily modified to include custom kinetic expressions or time-dependent boundary conditions. The updated code, known as Biofilm.jl, is written in the modern Julia programming language [28] which is designed to be fast while being a relatively easy language to write, run, and process results. Within Julia, Biofilm.jl leverages the state-of-the-art differential equation solvers [29] that help Biofilm.jl achieve fast and stable solutions to many biofilm problems.

The manuscript provides a detailed overview of the biofilm model and governing equations in Section 2 and the numerical methods and implementation are presented in Section 3. Specifics on how the code is organized and can be executed are in Section 4. Finally, results on applying the code to five different test cases are provided in Section 5.

2. Governing Equations

Biofilm.jl simulates a one-dimensional biofilm within a stirred tank reactor. More specifically, the concentrations of particulates X_t and substrates S_t within the tank are solved along with the particulate volume fractions P_b and substrate concentrations S_b within the biofilm. The thickness of the biofilm L_f is also computed. The code is written to solve for an arbitrary number of particulates and substrates, N_x and N_s, respectively.

2.1. Tank Equations

The temporal dynamics of particulate concentrations in the tank environment are described by

$$\frac{dX_{t:j}}{dt} = \mu_j(\boldsymbol{S}_t)X_{t:j} - \frac{QX_{t:j}}{V} + \frac{v_{\text{det}}AX_{b:j}(L_f)}{V} + \text{src}_{X:j} \tag{1}$$

for $j = 1, \ldots, N_x$. The terms on the right-hand-side (RHS) are 1) the growth of the particulate in the tank, 2) transport due to flow out of the tank, 3) transfer to the tank due to the detachment of particulates from the biofilm, and 4) a source term of particulates. The variables in the equation are as follows, t is time, $\mu_j(\boldsymbol{S}_t)$ is the growth rate of the jth

particulate and is dependent on the substrate concentrations within the tank S_t, Q is the flowrate into and out of the tank, V is the volume of the tank, A is the area of the biofilm, $X_{b:j}(L_f) = \rho_j P_{b:j}(L_f)$ is the jth particulate concentration at the top of the biofilm that can be computed from the particulate density ρ_j and volume fraction at the top of the biofilm $P_{b:j}(L_f)$, and $\text{src}_{X:j}$ is the source term for the jth particulate. The detachment velocity v_{det} is modeled using

$$v_{\text{det}} \equiv K_{\text{det}} L_f^2. \tag{2}$$

The substrate concentrations in the tank environment are described by

$$\frac{dS_{t:k}}{dt} = -\sum_{j=1}^{N_x} \frac{\mu_j(S_t)X_{t:j}}{Y_{j,k}} + \frac{QS_{\text{in}:k}}{V} - \frac{QS_{t:k}}{V} - \frac{AS_{t:k}^{\text{flux}}}{V} + \text{src}_{S:k} \tag{3}$$

for $k = 1, \ldots, N_s$, where $Y_{j,k}$ is the biomass yield coefficient on the substrate, $S_{\text{in}:k}$ is the influent substrate concentration, $S_{t:k}^{\text{flux}}$ is the flux of substrates into the biofilm from the tank due to diffusion, and $\text{src}_{S:k}$ is the source term for the kth substrate. The terms on the RHS are 1) consumption of substrates due to the growth of the particulate in the tank, 2) transport due to flow into the tank, 3) transport due to flow out of the tank, 4) transfer of substrates into the biofilm due to diffusion, and 5) a source term of substrates.

2.2. Biofilm Equations

The particulates within the biofilm are described using their volume fractions P_b and are described with

$$\frac{\partial P_{b:j}}{\partial t} = \mu_j(S_b)P_{b:j} - \frac{\partial v(z)P_{b:j}}{\partial z} + \frac{\text{src}_{X:j}}{\rho_j} \tag{4}$$

for $j = 1, \ldots, N_x$ and $P_{b:j}(t, z)$ is the jth particulate within the biofilm and varies with time and location within the biofilm. The terms on the RHS are 1) the growth of the particulate in the biofilm, 2) transport through the biofilm due to the growth velocity v_i, and 3) source term of particulate at i^{th} location in biofilm.

The growth velocity $v(z)$ is the vertical velocity of the biofilm due to the growth and sources of particulates deeper within the biofilm. In general, this velocity can be written as an integral that captures growth and sources from the bottom of the biofilm ($z = 0$) to some location z within the biofilm, i.e.,

$$v(z) = \int_{z'=0}^{z} \sum_{j=1}^{N_x} \frac{1}{P_{\text{tot}}} \left(\mu_j(S_b(z'))P_{b:j} + \frac{\text{src}_{X:j}}{\rho_j} \right) dz' \tag{5}$$

where $P_{\text{tot}} = \sum_{j=1}^{N_x} P_{b:j}$

The dynamics of substrates within the biofilm are described by

$$\frac{\partial S_{b:k}}{\partial t} = D_{b:k}\frac{\partial^2 S_{b:k}}{\partial z^2} - \sum_{j=1}^{N_x} \frac{\mu_j(S_b)X_{b:j}}{Y_{j,k}} + \text{src}_{S:k} \tag{6}$$

for $k = 1, \ldots, N_s$, where $D_{b:k}$ is the effective diffusion coefficient of the kth substrate within the biofilm. The terms on the RHS are 1) diffusion of substrates in the biofilm, 2) consumption of substrates due to the growth of the particulate in the biofilm, and 3) source term of substrates. Note that the top boundary condition of the diffusion term must match the flux of substrate into the biofilm $S_{t:k}^{\text{flux}}$ and provides a connection between the tank and biofilm dynamics.

2.2.1. Biofilm Thickness

The thickness of the biofilm L_f is described by

$$\frac{dL_f}{dt} = v(L_f) - v_{\text{det}} \tag{7}$$

where the first term on the RHS is the growth velocity at the top of the biofilm (Eq. 5) and the second term is the detachment velocity (Eq. 2).
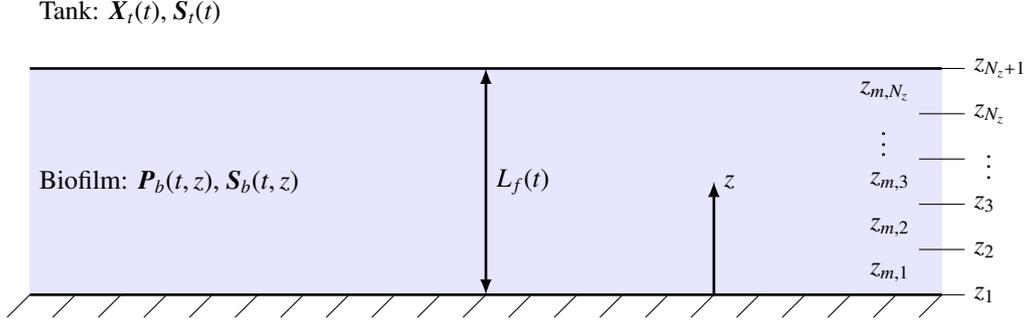
3

Tank: $X_t(t), S_t(t)$

Figure 1: Geometry used in the model. The tank is a stirred tank reactor with particulate $X_t(t)$ and substrate $S_t(t)$ concentrations. The one-dimensional biofilm contains particulate volume fractions $P_b(t, z)$ and substrate concentrations $S_b(t, z)$ that vary with z. The thickness of the biofilm is $L_f(t)$. The biofilm is discretized with N_z grid cells in the z-direction and the middle and faces of the grid cells are shown with z_m and z, respectively.

3. Discretization and Numerical Methods

The model solves for the particulate and substrate concentrations within the tank and the particulate volume fractions and substrate concentrations in the biofilm along with the biofilm thickness. The tank is assumed to be a stirred tank reactor and thus the tank concentrations are only a function of time. The variations within the biofilm are, however, important and a one-dimensional grid is used with N_z grid cells. The variables are stored at the cell centers denoted $z_{m,i}$ for $i = 1, \ldots, N_z$. Cell faces are denoted z_i for $i = 1, \ldots, N_z + 1$.

The governing equations consist of five time-dependent differential equations (Eqs. 1, 3, 4, 6, and 7) that are solved at the same time using built-in ODE solvers in Julia's DifferentialEquations.jl library [29]. To use the solvers the dependent variables are organized in a single vector such that the system of equations can be written as

$$\frac{d\mathbf{sol}}{dt} = \mathbf{rhs} \tag{8}$$

where

$$\mathbf{sol} = [X_t, S_t, P_b, S_b, Lf]^\mathsf{T}$$

where $[\cdot]^\mathsf{T}$ indicates transpose. The bold terms on the RHS of the previous equation contain the different particulates and substrates and can be written as

$$X = [X_{t:1}, X_{t:2}, \ldots, X_{t:N_x}]^\mathsf{T},$$
$$S = [S_{t:1}, S_{t:2}, \ldots, S_{t:N_s}]^\mathsf{T},$$
$$P_b = [P_{b:1}, P_{b:2}, \ldots, P_{b:N_x}]^\mathsf{T}, \text{ and}$$
$$S_b = [S_{b:1}, S_{b:2}, \ldots, S_{b:N_s}]^\mathsf{T}.$$

These last two expressions contain the particulate and substrates within the biofilm. The biofilm is discretized with N_z grid points and therefore each of these variables should be written as a vector of values at each grid location, i.e.,

$$P_{b:j} = [P_{b:j,1}, P_{b:j,2}, \ldots, P_{b:j,N_z}]^\mathsf{T}$$
$$S_{b:j} = [S_{b:j,1}, S_{b:j,2}, \ldots, S_{b:j,N_z}]^\mathsf{T}$$

All together **sol** is a vector with

$$N_{\text{var}} = N_x + N_s + N_x \cdot N_z + N_s \cdot N_z + 1 \tag{9}$$

variables.

Similarly, **rhs** is a vector of length N_{var} defined with

$$\mathbf{rhs} = \left[\frac{dX}{dt}, \frac{dS}{dt}, \frac{dP_b}{dt}, \frac{dS_b}{dt}, \frac{dLf}{dt} \right]^\mathsf{T}$$

4

where the right-hand sides are defined using the equations provided in Section 2. The RHS of the ordinary differential equations, Eq. 1, 3, and 7, can be used as written. However, the partial differential equations (Eq. 4 and 6) that describe the particulate volume fractions and substrate concentrations within the biofilm need to be discretized before they can be used with Julia's ODE solvers. Biofilm.jl uses a finite volume discretization to approximate the spatial derivatives.

To discretize Eq. 4 for the particulate volume fractions in the biofilm, the equation is written at each grid cell, i.e.,

$$\frac{\partial P_{b:j,i}}{\partial t} = \mu_j(S_{b:i})P_{b:j,i} - \frac{\partial v_i P_{b:j,i}}{\partial z} + \frac{\text{src}_{X:j,i}}{\rho_j} \tag{10}$$

for $j = 1, \ldots, N_x$ and $i = 1, \ldots, N_z$. The spatial derivative is discretized as

$$\frac{\partial v_i P_{b:j,i}}{\partial z} = \frac{P_{b:j,i+1}^{\text{flux}} - P_{b:j,i}^{\text{flux}}}{\Delta z}$$

where Δz is size of the grid cell and the flux on the ith face (z_i location) is defined as

$$P_{b:j,i}^{\text{flux}} = v(z_i)P_{b:j,i-1}.$$

for $j = 1, \ldots, N_x$ and $i = 2, \ldots, Nz + 1$. This expression approximates the volume fraction at the cell face using the volume fraction in the cell below the face introducing upwinding that stabilizes the solution. Note that a zero-flux condition is used at the bottom of the biofilm, $P_{b:j,1}^{\text{flux}} = 0$. In the previous equation $v(z_i)$ is computed using the discrete form of Eq. 5 which is

$$v(z_i) = \sum_{n=1}^{i-1} \sum_{j=1}^{N_x} \frac{1}{P_{\text{tot}}} \left(\mu_j(S_{b:i})P_{b:j,i} + \frac{\text{src}_{X:j,i}}{\rho_j} \right) \Delta z \tag{11}$$

and includes contributions from growth and sources in all cells below the ith face.

Similarly, to discretize Eq. 6 for the substrate concentrations in the biofilm, the equation is written at each grid cell, i.e.,

$$\frac{\partial S_{b:k,i}}{\partial t} = D_{b:k}\frac{\partial^2 S_{b:k,i}}{\partial z^2} - \sum_{j=1}^{N_x} \frac{\mu_j(S_{b:i})X_{b:j,i}}{Y_{j,k}} + \text{src}_{S:k,i} \tag{12}$$

for $k = 1, \ldots, N_s$ and $i = 1, \ldots, N_z$. The diffusion term with a second-order derivative is discretized using

$$D_{b:k}\frac{\partial^2 S_{b:k,i}}{\partial z^2} = \frac{S_{b:k,i+1}^{\text{flux}} - S_{b:k,i}^{\text{flux}}}{\Delta z}$$

and the flux on the ith face (z_i location) is defined using a second-order, central finite difference operator that can be written as

$$S_{b:k,i}^{\text{flux}} = D_{b:k}\frac{S_{b:k,i} - S_{b:k,i-1}}{\Delta z} \tag{13}$$

for $k = 1, \ldots, N_s$ and $i = 2, \ldots, N_z$ The flux requires boundary conditions at the top and bottom of the biofilm. A zero-flux condition is used at the bottom of the biofilm, i.e., $S_{b:k,1}^{\text{flux}} = 0$. For the boundary condition at the top of the biofilm, the diffusive flux from the tank is matched with the diffusive flux at the top of the biofilm, i.e.,

$$S_{t:k}^{\text{flux}} = S_{b:k,N_z+1}^{\text{flux}}$$
$$D_{t:k}\frac{dS_{t:k}}{dz} = D_{b:k}\frac{dS_{b:k,N_z}}{dz} \tag{14}$$

where D_t is the diffusion coefficient in the tank. A boundary layer within the tank at the top of the biofilm of thickness L_L is introduced to evaluate the derivative in the tank. Discretizing these derivatives requires introducing the substrate concentration at the top of the biofilm S_{top}, which when combined with finite differences, leads to

$$D_{t:k}\frac{S_{t:k} - S_{\text{top}}}{L_L} = D_{b:k}\frac{S_{\text{top}} - S_{b:k,N_z}}{\Delta z/2}.$$

Rearranging provides an expression for S_top, i.e.,

$$S_\text{top} = \frac{D_{t:k}(\Delta z/2)S_{t:k} + D_{b:k}L_L S_{b:k,N_z}}{D_{t:k}(\Delta z/2) + D_{b:k}L_L}. \tag{15}$$

With S_top defined, either the LHS or RHS of Eq. 14 can be used to set the flux at the top of the biofilm. Note that the previous expression is well-posed if the tank boundary layer is set to zero, i.e., $L_L = 0$ and the expression reduces to $S_\text{top} = S_{t:k}$.

In summary, the model consists of solving Eq. 8, which contains N_var (Eq. 9) ODEs. The RHSs of these ODEs are defined with Eqs. 1, 3, 10, 12, and 7. The next section describes the code organization that solves these equations including details on the inputs and outputs of the model.

4. Organization and Execution of the Code

The model is written entirely Julia. It was developed in Julia v1.8 and tested on v1.7. To use the model, a user will, typically, create a case file that defines the input parameters, calls the solver, and does any post-processing such as making plots or animations. Each of these is described in more detail below. Unit tests that compare simulated results to analytic solutions and a series of simulations compared to precomputed solutions are included and described in Appendix A. These unit tests ensure the accuracy of the code base.

4.1. Installation

Installation instructions are provided in the README file with additional detailed instructions provided at https://markowkes.github.io/Biofilm.jl/installation.

4.2. Input Parameters

The input parameters describe the biofilm to be simulated and provide code options. All the parameters are added to a dictionary (key:value structure). The dictionary is initially created with

```
# Create empty dictionary to hold parameters
d = createDict()
```

Parameters are then added by calling `addParam!`. For example, adding the simulation title and time to the dictionary can be done with, e.g.,

```
addParam!(d, "Title",    "Single Substrate and Particulate Case")
addParam!(d, "tFinal",   1.0)
```

This process of calling `addParam!` is repeated until all the parameters are added to the dictionary. The dictionary is now checked to make sure all the parameters are provided and packaged into a structure `p` that the solver is expecting by calling

```
# Package and check parameters
p = packageCheckParam(d)
```

The structure `p` fully describes the case and is the only variable passed to the solver. It might seem redundant to have the parameters entered into the dictionary `d` and then packaged into the struct `p`. While you can directly create the struct `p`, creating the dictionary first and then the struct is recommended because when building the dictionary much more helpful error messages can be produced to help the user identify and fix issues with the parameters, then packaging the parameters into the struct allows the solver to run more efficiently.

Below is a complete list of each parameter. Additional examples of setting the parameters can be found in the examples (https://markowkes.github.io/Biofilm.jl/examples/).

Simulation Parameters
- `Title` : Description of the case, used, e.g., on the title of plots.
- `tFinal` : Simulation is performed from $t = 0$ to $t =$`tFinal` days.
- `tol` : Tolerance used for differential equation solver. The solution will have an error less than `tol`.
- `outPeriod` : Period in days between text outputs during simulation. Note that the solver will take smaller timesteps than `outPeriod` to achieve the specified tolerance `tol` and the entire solution will be available when the solver completes.

Particulate Parameters
- `XNames` : Vector of strings with the name of each particulate used on text output and plots.
- `Xto` : Vector of initial particulate concentration(s) in the tank in g/m^3.
- `Pbo` : Vector of initial particulate volume fractions(s) in the biofilm. This value is used to set the volume fraction at all the grid cells.
- `rho` : Vector of particulate densities in g/m^3.
- `Kdet` : Detachment coefficient of particulates from the top of the biofilm in 1/m-d.
- `mu` : Vector of functions that provide the particulate growth rates in 1/d. For example, in the example in Section 5.3 there are two particulates, "Live" and "Dead", and one substrate. The growth rates are defined as

```
addParam!(d, "mu", [(S,X,Lf,t,z,p) -> (mumax * S[1]) / (KM + S[1])
                    (S,X,Lf,t,z,p) -> 0.0 ] )
```

which sets the growth rate of the first particulate to the Monod equation [30]

$$\mu_1 = \mu_{\text{Live}} = \mu_{\max} \frac{S_1}{K_M + S_1} \tag{16}$$

and the second growth rate to $\mu_2 = \mu_{\text{Dead}} = 0$. The growth rates can be other functions such as a double Monod equation, inhibition model, or any other function of the substrate concentrations `S`, particulate concentrations `X`, biofilm thickness `Lf`, time `t`, height in biofilm `z`, or other parameters within the struct `p`.
- `srcX` : Vector of functions that provide the sources of particulate concentrations in g/m^3-d.

Substrate Parameters
- `SNames` : Vector of strings with the name of each substrate used on text output and plots.
- `Sto` : Vector of initial substrate concentration(s) in the tank in g/m^3.
- `Sbo` : Vector of initial substrate concentration(s) in the biofilm in g/m^3. This concentration is used to set the concentration at all the grid cells.
- `Sin` : Vector of functions that provide the influent substrate concentrations in g/m^3.
- `srcS` : Vector of functions that provide the sources of substrates concentrations in g/m^3-d.
- `Yxs` : Matrix of biomass yield coefficients on the substrate. The matrix has size $N_x \times N_s$. Within the matrix, the value in the jth row and k^{th} column, i.e., `Yxs[j,k]`, provides the number of grams of the jth particulate grown for each gram of the kth substrate consumed. Positive values of `Yxs` indicate substrate consumption and negative values indicate substrate generation by particulate growth. If the particulate does not depend on the substrate the yield coefficient should be infinite. However, to make this input more user-friendly, a value of `0` or `Inf` can be entered and the code automatically converts a `0` to `Inf`.
- `Db` : Vector of effective substrate diffusion coefficients through the biofilm in m^2/day.
- `Dt` : Vector of aqueous substrate diffusion coefficients in the fluid environment within the tank in m^2/day.

7

Tank Parameters

- `V` : Volume of tank in m^3.

- `A` : Area of biofilm within the tank in m^2.

- `Q` : Flowrate into and out of tank in m^3/day.

Biofilm Parameters

- `Nz` : Number of grid points used to discretize the biofilm.

- `LL` : Thickness of boundary layer within the tank on the surface of biofilm in m.

- `Lfo` : Initial thickness of the biofilm in m.

Optional Parameters

- `plotPeriod` [default=`outPeriod`] : Period in days between plot renderings during the simulation. For long simulations, you can increase `plotPeriod` to reduce the number of plots that are created and speed up the simulation. Note: `plotPeriod` is required to be a multiple of `outPeriod`.

- `plotSize` [default=`(1600,1000)`] : Size of plots in pixels.

- `makePlots` [default=**`true`**] : Boolean that controls if code should produce plots while running.

- `optionalPlot` [default=`"growthrate"`] : While running, the code displays six plots. The sixth plot defaults to plotting the particulate growth rate versus location in biofilm. Setting `optionalPlot = "source"` will change this plot to the particulate source term versus location in biofilm.

When defining the parameters it is often useful to reference other variables or functions. For example, when defining the growth rate in Eq. 16 the variables $\mu_{\max}$ and K_m are used. In the code, these variables can be defined before `growthrate` is added to the dictionary. Similarly, functions can be defined and used when defining parameters, see Example 5.5.

4.3. Overview of Solver

The solver consists of a collection of Julia (.jl) files. Each file is described in additional detail below.

Biofilm.jl :   is the main file of the code base and defines the Biofilm module, exported functions and objects, and includes all the other files. Biofilm.jl is used by first including the module

```julia
julia> using Biofilm
```

and then calling one of the functions described below.

solver.jl :   contains the function `BiofilmSolver(p::param)`, which runs a simulation of a biofilm using the parameters provided in the struct `p` that includes all the inputs described in Section 4.2. This function prepares the solver, runs the solver, and formats and returns the solution to be easily interpreted by the user.

The workhorse within `BiofilmSolver` is a call to the function `solve()`, which is part of the DifferentialEquations.jl package [29] This package contains many state-of-the-art methods for solving differential equations and is key to the speed of Biofilm.jl. The ODE solver computes the solution of Eq. 8.

rhs.jl :   contains `biofilmRHS!()`, which computes the RHS of the differential equations in Eq. 8. The first part of this function splits the vector of dependent variables **sol** into X_t, S_t, $\boldsymbol{P}_b$, $\boldsymbol{S}_b$, and L_f. Next, a number of intermediate variables are computed which are terms that appear multiple times in the RHSs by calling functions that are defined in computes.jl. Finally the RHS of Eqs. 1, 3, 10, 12, and 7 are computed and organized in a single vector.

computes.jl :   contains functions that compute intermediate variables that appear multiple times in the evaluation of the differential equation RHSs. By computing these variables once the results can be reused to improve the speed of the code.

outputs.jl : produces a text output to the terminal and plots the solution.

structs.jl : defines the composite types or structs used by the code. These structs hold a number of parameters or variables together to reduce the number of variables that need to be passed into functions. At the top of each function, any variables from the structures are accessed with the `@unpack` macro.

parameters.jl : defines the functions `createDict()`, `addParam!(d,name,value)`, and `packageCheckParam(d)` that are used to create a dictionary of parameters and package and check the parameters. Additionally, this file provides the function `printDict(d)` that can be used to print all the values in the a dictionary.

tools.jl : provides a variety of helper functions.

postprocess.jl : contains functions to help a user analyze the results of a simulation. Additional details are provided in the next section.

4.4. Postprocessing Output

A simulation is run by calling

```julia
julia> t,zm,Xt,St,Pb,Sb,Lf,sol = BiofilmSolver(p) # Run solver
```

The results are returned in the following output variables. `t` is a vector of the solution times. `zm` is a vector of grid cell centers used to discretize the biofilm. Since the biofilm thickness (and thus the grid) changes with time, only the grid at the final time is provided as an output. `Xt` and `St` are arrays of the particulate and substrate concentrations in the tank as a function of times `t`. The variables `Pb` and `Sb` are arrays of the biofilm particulate volume fractions and substrate concentrations at the final simulation time as a function of the grid point locations `zm`. Finally, `sol` is an object that is created by DifferentialEquations.jl that contains the complete solution of the ODE solver. Although it contains the entire solution, it is difficult to process since it contains N_{var} (Eq. 9) dependent variables. These outputs can be analyzed to provide additional information on the biofilm that has been simulated using the following functions.

`biofilm_plot(sol,p)` : can be used to produce the standard plots of the simulation as shown in, e.g., Fig. 2. This is useful to make a plot of the solution at the end of the simulation, especially if plots are turned off during the simulation using the optional parameter `makePlots=`**`false`**.

This function calls a plotting recipe and allows for considerable customization of the plot using the inputs of the Plots.jl library and solution processing form DifferentialEquations.jl. The examples in Section 5 demonstrate some plots that can be created with this function.

`biofilm_analyze(sol,p,times [,makePlot=`**`true`**`])` : will display the dependent variables at the specified times `t`. For example, after running the example described in Section 5.1, the solution could be analyzed to check the solution at $t = [0, 0.25, 0.5, 0.75, 1.0]$, i.e.,

```julia
julia> times = 0:0.25:1.0
julia> biofilm_analyze( sol, p, times)
Analyzing Single Substrate and Particulate Case
  Time   |     Bug |   Oxygen | min,max(     Bug) | min,max(   Oxygen) |      Lf
   0.000 |      10 |       10 |    0.08,    0.08 |       0,       0 |      10
   0.250 |     102 |     51.7 |    0.08,    0.08 |    29.1,    51.3 |     545
   0.500 |     256 |     2.94 |    0.08,    0.08 |   0.568,    2.87 |     348
   0.750 |     257 |     2.93 |    0.08,    0.08 |   0.745,    2.87 |     312
   1.000 |     257 |     2.93 |    0.08,    0.08 |   0.761,    2.87 |     309
```

where the first column is time, the second and third columns are the particulate and substrate concentrations in the tank, the fourth and fifth columns are the minimum and maximum of the particulate volume fractions and substrate concentration within the biofilm, and the last column is the biofilm thickness. This function provides the outputs `Xt`, `St`, and `Lf` at the specified times for further postprocessing. For example, this is used to process the example in Section 5.5 to create Fig. 9.

Including the optional parameter `makePlots=`**`true`** will create a plot of the solution within the biofilm at each specified time. This is used in the processing of the example in Section 5.5.

`biofilm_movie(sol, p, times [, filename="anim.gif", fps=20)]`: can be used to make an animation of the solution at the specified times. The optional parameters `filename` and `fps` can be included to change the filename or frame rate of the movie. This function uses the `@animate` macro and `gif()` function from the Plots.jl package.

`biofilm_sol2csv( sol, p [, filename="biofilm.csv"])`: takes the output of a simulation and writes the results as a CSV file. Each row in the file corresponds to a different output time and the column contains the time, the tank particulate and substrate concentrations, biofilm particulate volume fractions and substrate concentrations, and finally the biofilm thickness. This function can be useful for users that prefer looking at results in a spreadsheet.

5. Examples

This section provides a series of example cases that highlight the capabilities of Biofilm.jl. These examples are the same as those provided with the code, however, additional postprocessing has been done for some of the cases. The code used to create the results in this manuscript is available in a branch called CodePaper accessible at `https://github.com/markowkes/Biofilm.jl/tree/CodePaper/examples`

5.1. Case 1: Single Substrate and Particulate

This is a simple example of a single particulate named `"Heterotroph"` that consumes the substrate `"Nutrient"` to grow using the Monod equation [30]. The case is run by starting Julia, installing Biofilm.jl, and executing

```
julia> include("examples/Case1.jl")
```

The output from the program is shown in Fig. 2. Note that the size of the plot was adjusted by creating the plot with the command

```
julia> biofilm_plot(sol,p,size=(900,600))
```

which is run after the simulation is complete. The top row shows how the particulate concentration (left), substrate concentration (center), and biofilm thickness (right) vary with time. The bottom row shows the particulate volume fraction (left), substrate concentration (center), and particulate growth rate (right) versus location within the biofilm at the end of the simulation (1.00 days for this case).

The heterotroph concentration in the tank starts at the initial condition of `Xto`=10 g/m^3 and increases to a steady-state value of 257 g/m^3 as seen in the top-left plot. The nutrient concentration in the tank starts at an initial concentration of `Sto`=10 g/m^3, increases quickly due to the flow into the tank with a concentration of `Sin`=100 g/m^3, then starts decreasing as the heterotroph concentration increases and more of the nutrient is consumed until a steady-state of 2.92 g/m^3 is reached as shown in the top-center plot. The biofilm thickness increases from a value of `Lfo`=1.0E-5 μm quickly due to growth in the initial system with an abundance of the nutrient. Eventually, the thickness decreases to a steady thickness of 309 μm as observed in the top-right plot.

Within the biofilm at the final time of `t`=1.0 days, the bottom-left plot shows that the heterotroph volume fraction is 0.08 everywhere. This value comes from the initial condition of `Pbo`=0.08 and the fact that there is only one particulate in this simple system. The nutrient concentration, in the bottom-center plot, is highest at the top of the biofilm and decreases within the biofilm as diffusion of the nutrient into the biofilm is balanced by the consumption by the heterotroph. The concentration of the nutrient at the top of the biofilm has a value of `Sb[Nz]`=2.87 g/m^3, which is close to the steady-state tank concentration of 2.92 g/m^3. The difference is due to the diffusion rate through the boundary layer and very top of biofilm (Eq. 14) The heterotroph's growth rate, shown in the bottom-right plot, is highest near the top of the biofilm where the nutrient concentration is highest and decreases further into the biofilm.

10

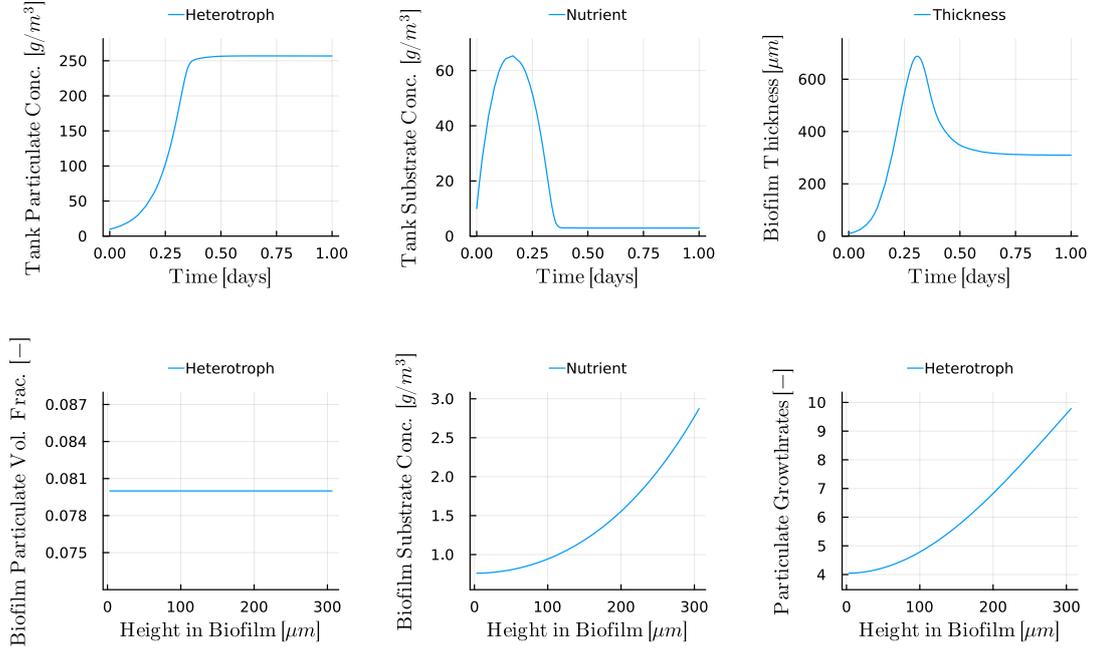## Single Substrate and Particulate Case : t = 1.00

Figure 2: Standard output at the end of running Case1.jl.

5.2. Case 2: Multiple Substrates

This example is based on the acid stress response in Zhang et al. [31] and features biomass X that consumes glucose s and produces lactate p. Here the variables s and p stand for substrate and product, respectively. Note that both s and p are modeled with Eqns. 3 and 6 with $S = [s, p]^\mathsf{T}$.

The presence of lactate can inhibit the growth of biomass. This appears as a reduction in growth rate, as p increases, until a maximum value of $p_{\max}$ above which the growth ceases, i.e.,

$$\mu = \begin{cases} \mu_{\max} s \left(1 - \dfrac{p}{p_{\max}}\right) & p \le p_{\max} \\ 0.0 & p > p_{\max} \end{cases} \tag{17}$$

where the term in parentheses captures inhibition. In the code, this is done with two steps. The first is to define a function that computes the growth rate for a single combination of s and p, i.e.,

```
# Define mu function
function mu(s,p)
    if p < p_max
        mu = mu_max*s*(1-p/p_max)
    else
        mu = 0.0
    end
    return mu
end
```

Next, this function is provided to the growth rate variable in the parameter dictionary (see Section 4.2), i.e.,

```
# Growthrate: call mu(s,p) for S = [s,p]
addParam!(d, "mu", [(S,X,Lf,t,z,p) -> mu(S[1],S[2])])
```

11

which calls `mu(s,p)` function using the notation `S = [S[1],S[2]] = [s,p]`.
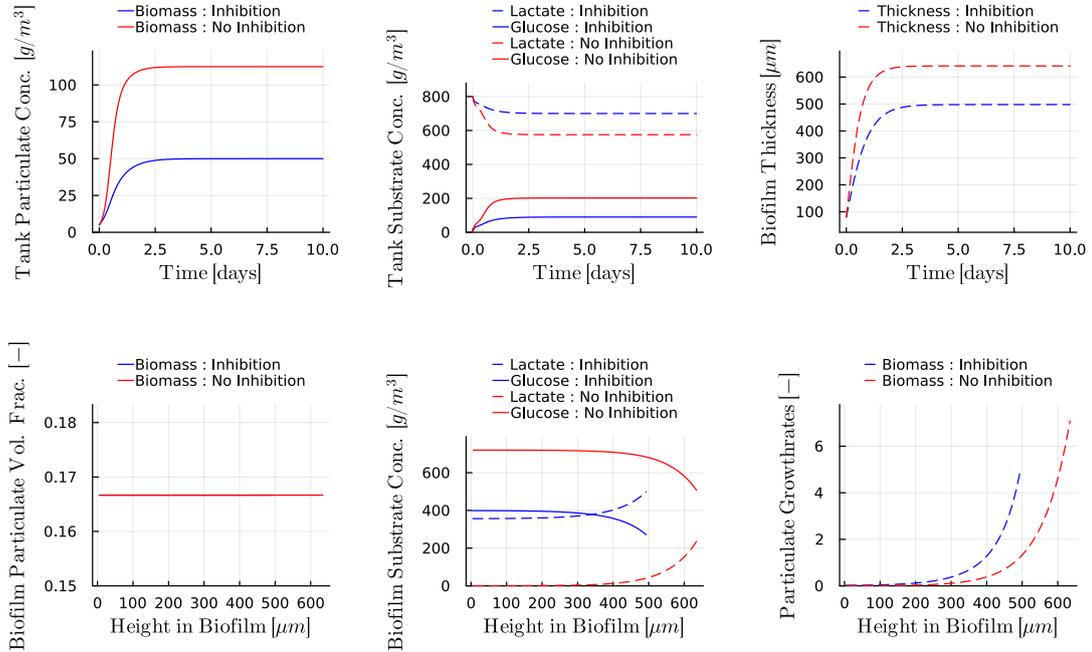

Figure 3: Output of Case2.jl run with (blue) and without (red) growth inhibition.

To see the impact of the inhibition, two simulations are run with the inhibition turned on and off, respectively. The results from both simulations are shown in Fig. 3 which was generated with the code

```
# Run with inhibition
p_max = 400
t_in,zm_in,Xt_in,St_in,Pb_in,Sb_in,Lf_in,sol_in = BiofilmSolver(p)
plt = biofilm_plot(sol_in,p,"Inhibition",
                   size=(900,600), line=(:blue,[:solid :dash]))

# Run without inhibition & add to plot
p_max  = Inf
t_no,zm_no,Xt_no,St_no,Pb_no,Sb_no,Lf_no,sol_no = BiofilmSolver(p)
plt = biofilm_plot!(plt,sol_no,p,"No Inhibition",
                    size=(900,600), line=(:red,[:solid :dash]))
```

This code is run after running a standard Case 3 simulation which sets the case parameters. The code above runs two simulations, the first with the inhibition turned on (`p_max = 400`) and plots the results in blue. Next a second simulation is run without inhibition (`p_max = Inf` and the result is added to the previous plot and colored in red. Note that the second plot command uses `biofilm_plot!` (note the exclamation point), which adds to the plot instead of creating a new plot.

The results in Fig. 3 show the output of both simulations. In the top row, the biomass (left) and substrate (center) concentrations along with the biofilm thickness (right) are shown versus time. After roughly 2.5 days all the quantities reach a steady state. The bottom-left plot shows the biomass volume fraction, which since there is only one particulate remains at the input value of 1/6 throughout the biofilm. The bottom-center plot shows the glucose and lactate concentrations throughout the biofilm. The glucose (solid lines) is diffusing from the tank into the biofilm and is consumed by the biomass, which lowers the concentration at lower heights. The lactate (dashed lines) is produced by

biomass and has high concentrations in the lower part of the biofilm. Diffusion moves the lactate toward the surface of the biofilm where it enters the tank. The biomass growth rate is shown in the bottom-right figure. Growth is highest at the top of the biofilm where high glucose concentrations exists and decreases at lower heights in the biofilm.

Comparing the inhibited (blue) results to the case without inhibition (red) shows that without inhibition the biomass concentrations in the tank (top-left) is over 2 times larger and the biofilm is roughly 25% thicker (top-right). For both cases the growth of biomass occurs at the top of the biofilm and ceases at lower heights within the biofilm, however the limit to growth is different in the two cases. Without inhibition, the biomass consumes all of the glucose that is diffusing through the biofilm and the glucose concentration goes to zero, limiting growth, at lower heights in the biofilm (bottom-center). With inhibition, growth is limited by the inhibition due to the higher concentrations of lactate lower in the biofilm (bottom-center) and some glucose exists even at the bottom of the biofilm (bottom-center).

5.3. Case 3: Multiple Particulates

This example demonstrates a simulation with multiple particulates, namely `"Living Bug"` and `"Dead Bug"`. The living bug consumes the substrate and grows using the Monod equation. The living bugs die and turn into dead bugs, this is captured in the model with the source terms

$$\text{src}_{\text{Living}} = -bX_{\text{Living}}, \text{ and}$$
$$\text{src}_{\text{Dead}} = bX_{\text{Living}}.$$

The results at the end of the 100-day simulation are shown in Fig. 4. Of note for this case are the variations within the biofilm. Near the surface of the biofilm, the substrate diffuses from the tank (bottom-center plot) and is consumed by the living bugs causing the living bug volume fraction to be larger in roughly the top half of the biofilm. Deeper in the biofilm, the living bugs do not receive substrate and die causing the dead bug volume fraction to increase. For this case, the parameter `optionalPlot="source"` is used by adding it to the parameter dictionary with

```
addParam!(d, "optionalPlot", "source")
```

With this parameter, the bottom-right figure shows the source term that transfers living bugs to dead bugs. The source has the largest magnitude near the top of the biofilm where the living bug concentration is the highest.

5.4. Case 4: Multiple Substrates and Particulates

This case demonstrates how to model a biofilm with three substrates and three particulates. The substrates are oxygen, sulfate, and hydrogen sulfide and the particulates are sulfide-oxidizing bacteria (SOB), sulfate-reducing bacteria (SRB), and dead bacteria.

The system requires interactions between the bacteria and substrates to function. Oxygen and sulfate enter the tank with the influent. The SRB uses the sulfate and produces sulfide, but this growth is inhibited by oxygen. The SOB can then use the sulfide, produced by the SRB, and oxygen to grow. The SRB and SOB slowly die creating the dead bacteria. These dynamics appear in the growth rates which can be written as

$$\mu_{\text{SOB}} = \mu_{\text{max:B}} \left(\frac{S_1}{K_{m:B_1} + S_1} \cdot \frac{S_3}{K_{m:B_3} + S_3} \right)$$
$$\mu_{\text{SRB}} = \mu_{\text{max:C}} \left(\frac{S_2}{K_{m:C_2} + S_2} \cdot \frac{1.0}{1.0 + S_1/K_I} \right)$$
$$\mu_{\text{Dead}} = 0.0$$

where S_1, S_2, and S_3 correspond to oxygen, sulfate, and hydrogen sulfide, respectively. The parameters $\mu_{\text{max:B}}$, $\mu_{\text{max:C}}$, $K_{m:B_1}$, $K_{m:B_3}$, $K_{m:C_2}$, and K_I are constants and set in the case file. The source terms can be written as

$$\text{src}_{\text{SOB}} = -D_{\text{SOB}}X_1$$
$$\text{src}_{\text{SRB}} = \phantom{+D_{\text{SOB}}X_1} -D_{\text{SRB}}X_2 - D_{\text{O:SRB}}S_1$$
$$\text{src}_{\text{Dead}} = +D_{\text{SOB}}X_1 + D_{\text{SRB}}X_2 + D_{\text{O:SRB}}S_1$$

where X_1, X_2, and X_3 correspond to SOB, SRB, and dead biomass, respectively. The constants D_{SOB} and D_{SRB} describe the death rate of SOB and SRB, respectively. $D_{O:\text{SOB}}$ is the death rate of SRB due to the precense of oxygen.
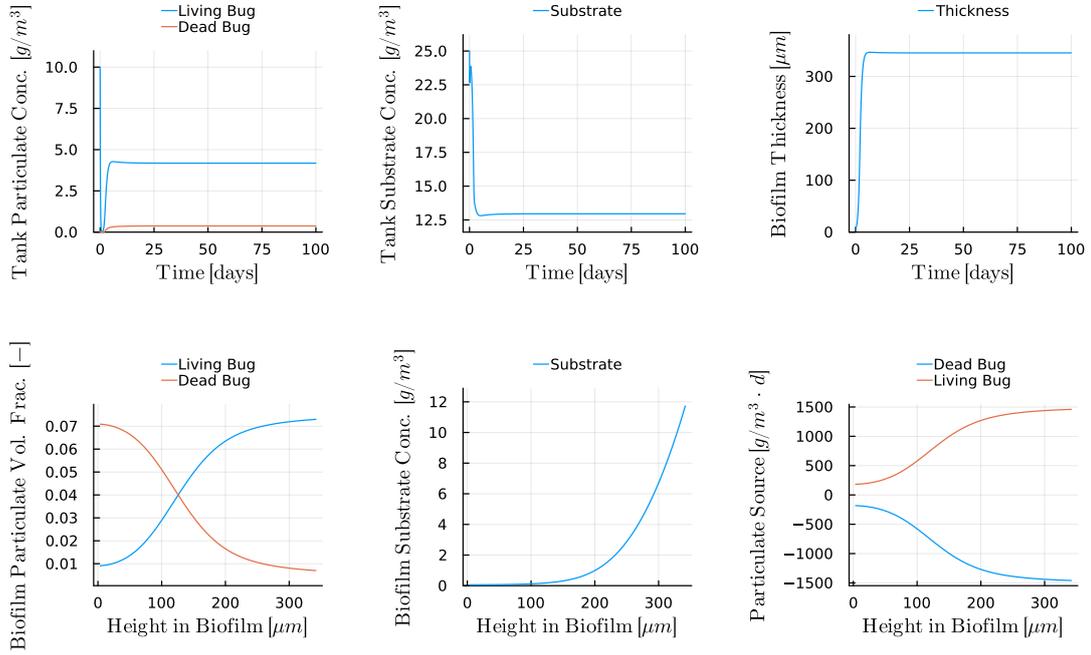
These expressions are coded as

Figure 4: Standard output at the end of running Case3.jl.

```
addParam!(d, "mu", [
    (S,X,Lf,t,z,p) -> mumaxB*(S[1]./(KmB1.+S[1])).*(S[3]./(KmB3.+S[3])),   # SOB
    (S,X,Lf,t,z,p) -> mumaxC*(S[2]./(KmC2.+S[2])).*(1.0./(1.0.+S[1]/KI)) , # SRB
    (S,X,Lf,t,z,p) -> 0.0 ] )                                              # Dead
```

and

```
addParam!(d, "srcX",  [
    (S,X,Lf,t,z,p) -> - D_SOB*X[1]                                ,   # SOB
    (S,X,Lf,t,z,p) ->             - D_SRB*X[2] - D_O_SRB*S[1] ,   # SRB
    (S,X,Lf,t,z,p) -> + D_SRB*X[2] + D_SOB*X[1] + D_O_SRB*S[1] ]) # Dead
```

The results after 100 days are shown in Fig. 5. The top row shows the SOB, SRB, and dead bacteria concentrations within the tank (left), substrate concentrations (center), and biofilm thickness (right). After roughly 50 days the quantities reach a steady state. The bottom row shows the particulate volume fractions (left), substrate concentrations (center), and particulate growth rates (right). In the bottom of the biofilm, a layer of dead cells collects as little sulfate (needed by SRB) or oxygen (needed by SOB) reaches this layer. In the middle region, the SOB lives and has access to the sulfide from the SRB and oxygen diffusing from the tank. In the top layer, the SRB thrives as the sulfate concentration is very high.

5.5. Case 5: Phototroph

This example demonstrates how a particulate that grows in light with a diurnal cycle can be simulated. The particulate is a phototroph with a growth rate that depends on light intensity $I(t, z)$, i.e.,

$$\mu_{\text{phototroph}} = \mu_{\text{max}} I(t, z).$$

As the phototroph grows it generates the substrate, oxygen, through the biomass yield coefficient `Yxs=[-0.52]`.
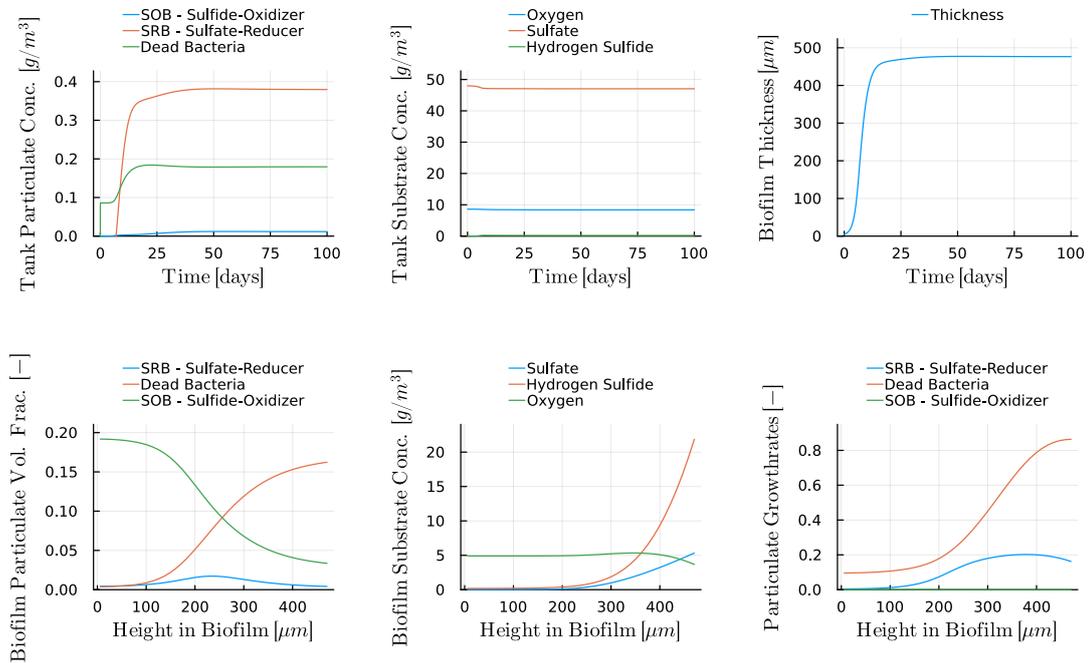
14

Figure 5: Standard output at the end of running Case4.jl.

The light within the tank turns on and off throughout each day. Mathematically, the intensity of light within the tank is 1.0 on each day after 0.25 days and then 0.0 at 0.75 days creating the on/off cycle. In the code, this is achieved by defining the intensity function as

```
# Define smoothed Heaviside function
smoothHeaviside(t,t0)=0.5*tanh.(100*(t.-t0).-0.5).+0.5
# Define light intensity as function of time
intensity(t) = 1.0 - (smoothHeaviside(mod(t,1),0.25)
                      -smoothHeaviside(mod(t,1),0.75))
```

where the `smoothHeaviside` function spreads out the discontinuity in intensity over a finite time. Furthermore, the light dissipates as it travels through the biofilm and the dissipation is defined with

```
diss=2000; # Dissipation rate [1/m]
dissipation(z,Lf) = max.(0.0,1.0.-(Lf.-z)*diss)
```

Finally, the light intensity at a specified time and position in the biofilm can be computed with

```
# Define light intensity as a function of
# time and location within biofilm
light(t,z,Lf) = intensity(t)*dissipation(z,Lf)
```

These functions are defined before the input parameters and are called when the growth rate is defined as

```
mumax = 0.4;
addParam!(d, "mu", [(S,X,Lf,t,z,p) -> mumax*light(t,z,Lf)])
```

To check the light intensity function, we can make a plot of light intensity over a few days within the tank, e.g.,

```
# Define constants
Lf = 600e-6; z = Lf;
```

15

```
# Define times to plot
t = range(0.0,3.0,1000)
# Make plot
plot(t,map(t -> light(t,z,Lf),t))
```

which produces the plot shown in Fig. 6a. Figure 6b shows how the light dissipates as it travels through the biofilm and it is largest at the top of the biofilm (600 μm) and decreases linearly until reaching a zero intensity.

(a) Variation with time
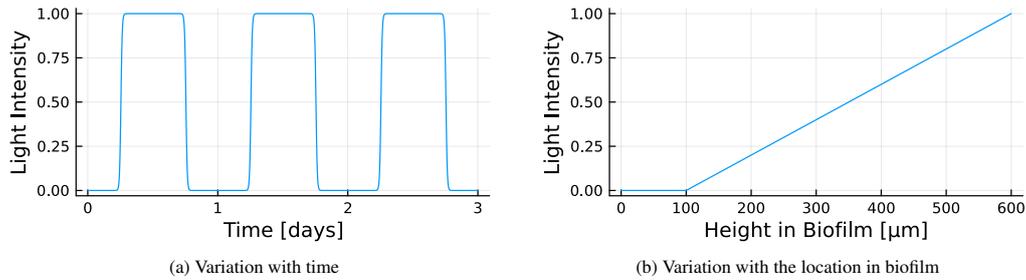

(b) Variation with the location in biofilm

Figure 6: Light intensity versus time and location within biofilm for Case 5.

Due to the diurnal discontinuities in the light intensity, it is important to make sure the solver reduces the timestep near the discontinuity. This is achieved by setting `discontinuityPeriod=0.25` using

```
# Let the solver know when discontinuities (changes in light) occur
addParam!(d, "discontinuityPeriod",0.25)
```

which makes the solver look for discontinuities every 0.25 days. Without this parameter, periods when the light is turned on or off are missed since the solver uses large timesteps.
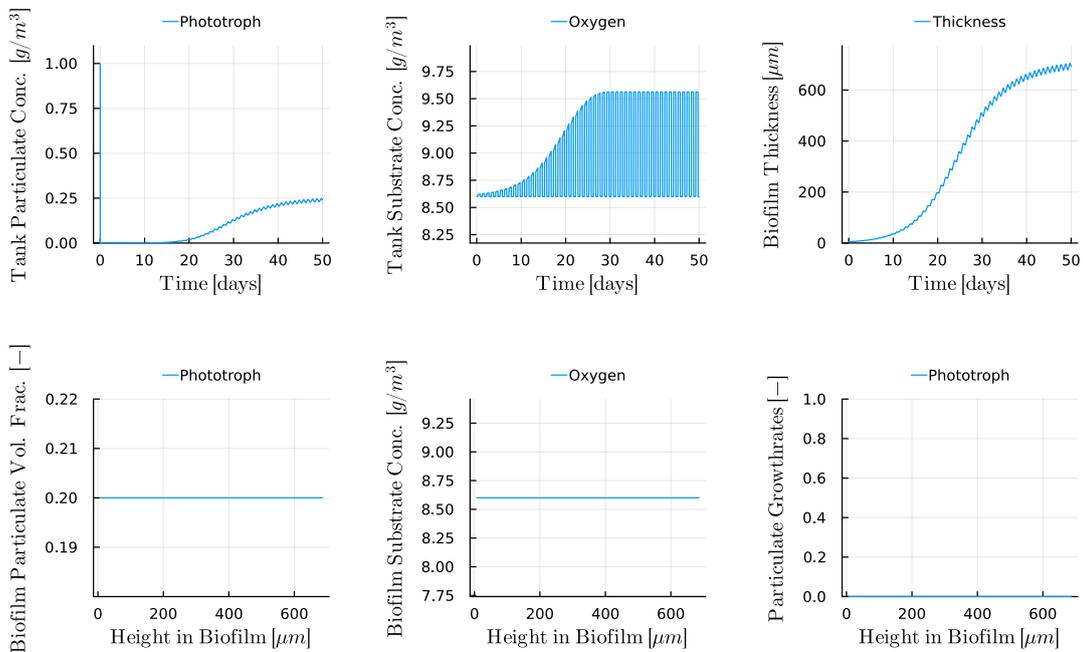

Figure 7: Standard output at the end of running Case5.jl.

16

The output of this example is shown in Fig. 7. The top row shows the phototroph concentration (top-left), oxygen concentration (top-center), and biofilm thickness (top-right) all as a function of time. The results vary during each day as the light turns on and off. The bottom row displays the variations within the biofilm at the final time of 50 days when the light is off. With the light off, no growth is occurring and the quantities do not vary throughout the biofilm.
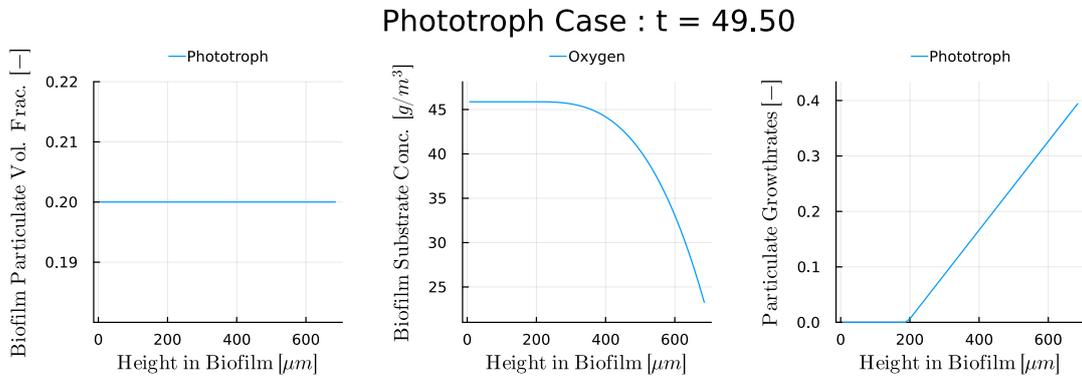

Figure 8: Analysis of Case 5 solution at t=49.5 days.

The quantities within the biofilm can be analyzed at different times. In particular, it will be useful to analyze the results when the light is on. This can be done by running the `biofilm_analyze()` postprocessing function with the solution of Case 5 when the light is on (e.g., 49.5 days) which is done by running

```
tout=49.5 # Time when the light is on
biofilm_analyze(sol,p,tout,makePlot=true)
```

This code analyzes the results and produces the plots shown in Fig. 8. When the light is on, the phototroph is growing (right) and the oxygen concentration is high within the biofilm as it is produced by the phototroph and decreases towards the top of the biofilm due to diffusion (center). Since there is only one particulate, the figure of the volume fraction (left) remains the same throughout the entire simulation.
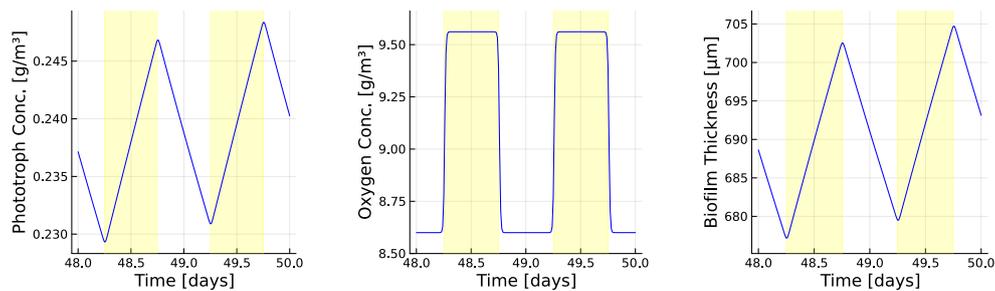

Figure 9: Phototroph and oxygen concentrations and biofilm thickness versus last two days of Case 5 simulation. Yellow regions indicate times when the light is on.

Due to the high-frequency changes in the results, this example can be post-processed to observe additional features in the solution. To plot just the last two days of the simulation, the `biofilm_analyze()` function can be used again, but now with multiple times, i.e.,

```
# Times to analyze solution
tout = 48.0:0.01:50.0;
# Get solution at certain times
Xtout,Stout,Lfout = biofilm_analyze(sol,p,tout)
# Plot tank particulate concentration versus time
```

17

```
plot(tout,Xtout')
```

which produces a simpler version of the left plot in Fig. 9. The complete code to produce the three plots in Fig. 9 is available in Appendix B.

With these results, it is clear how the quantities vary with time. When the light is on the phototroph (particulate) concentration increases and then decreases when the light is off. The oxygen concentration is highest in the tank when the light is on and oxygen is generated by the growing phototroph. When the light is off, the oxygen concentration decreases to 8.6 which matches the specified influent substrate concentration `Sin`. The biofilm thickness increases when the light is on due to the phototroph growth in the biofilm and then decreases when the light is off.

6. Conclusions

A modern implementation of a one-dimensional biofilm model is presented. The software, known as Biofilm.jl, solves for the temporal dynamics of particulates (biomass) and substrates which are effected by biological phenomena including growth, metabolism, and competition as well as physical processes such as diffusion, convection, attachment and detachment. The model is implemented in Julia [28] and leverages the state-of-the-art differential equations solvers provided by DiffereentialEquations.jl library [29].

Example biofilms are provided to demonstrate the capabilities of Biofilm.jl. The first example is of a single heterotroph growing in an environment with a nutrient. The second example demonstrates an acid stress response and has a biomass that consumes glucose and produces lactate which inhibits the biomass growth. The third example demonstratess a single biomass species that grows and then dies. The fourth example demonstrates a complex interplay of sufide-oxidizing bacteria (SOB), sulfate-reducing bacteria (SRB), and dead bacteria with the substrates oxygen, sulfate, and hydrogen sulfide. Finally, a fifth example is of a phototroph in a diurnal cycle. The examples highlight the variety of biofilms that Biofilm.jl can simulate as well as provide templates for postprocessing simulation results.

Biofilm.jl is provided with an open-source license and is hosted on github. The repository contains detailed user manual with installation, execution, and postprocessing instructions as well as an overview of the governing equations. Coding and Julia best practices, including documentation, unit tests, and plot recipes, are implemented.

Appendix A. Code Unit Tests

Tests have been implemented to ensure the code is running correctly. The tests consist of 1) a series of unit tests that check specific parts of the code and compare the results to analytic solutions and 2) examples that are compared with previously computed answers to ensure a code modification does not introduce a bug. Below are details on the unit tests and the analytic solutions.

The tests can be run by cloning the GitHub repository, activating the package, entering the package manager by typing "]" at the Julia prompt, and then running `test`. In addition to manual tests, all the tests are executed automatically each time a new commit is added to the GitHub repository.

Appendix A.1. Zero L_L

Having the code behave robustly when the boundary layer within the tank is set to zero is a nice code feature. This is achieved through Eq. 15 which avoids dividing by zero. The unit test `test_zeroLL` runs a simulation with `LL = 0.0` and checks that the substrate concentration at the top of the biofilm `S_top` equals the substrate concentration at the topmost grid cell within the biofilm. The test shows the condition is met to machine precision.

Appendix A.2. Flux matching

At the top of the biofilm the flux of substrate through the tank boundary layer matches the flux through the top of the biofilm to the first grid point, i.e., Eq. 14. This test ensures that the calculation of `S_top` is correct and the fluxes do match computing `S_top` then computing the two fluxes and ensuring they are equal. The test shows the condition is met to machine precision.

Appendix A.3. Diffusion test

The distribution of substrates within the biofilm is mainly controlled by diffusion and consumption due to the growth of particulates. In this test, the steady-state distribution of substrate within the biofilm is compared with the analytic solution for a biofilm with one substrate and one particulate. The particulate has a growth rate of $\mu = \mu_{\max} S / K_M$. This biofilm, at steady-state, has an analytic solution of [32]

$$S_{b:1}(z) = S_{t:1} \frac{\cosh \phi z / Lf}{\cosh \phi} \tag{A.1}$$

where

$$\phi = \sqrt{\frac{\mu_{\max} \rho P_{b:1} L_f^2}{D_b K_m Y_{1,1}}}$$

The unit test consists of running a series of simulations until the biofilm reaches a steady state with different numbers of grid cells within the biofilm. For each simulation, the error, which is the maximum difference between the analytic and computed solution, is computed. Finally, the order of the method is computed by fitting a line to the error versus the number of grid points in log-log space. It is expected the methods are second-order accurate and that the test shows an order of 1.9 for the grids tested.

Appendix A.4. Steady-State with Large Diffusivity

If the diffusion coefficients D_b and D_t are considered to be very large, a steady-state solution to the biofilm model can be found since the substrate concentration can be assumed to be uniform within the biofilm and also equal to the tank concentration. Note that in the following derivation of the analytic solution, subscripts on the substrate and particulate have been dropped as there is only one of each.

The steady-state solution can be found by setting the time derivatives equal to zero in the differential equations. For example, the thickness of the biofilm is described by Eq. 7 and at steady-state this reduces to

$$v(z_{N_z+1}) = v_{\det},$$

which means that the vertical velocity due to growth in the biofilm (LHS) matches the detachment velocity (RHS). The LHS can be computed using Eq. 11 which when simplified, by setting $\mathrm{src}_X = 0$ and letting $S_b = S$ where S is the uniform substrate concentration in the biofilm and tank, becomes $v(z_{N_z}) = \mu(S) L_f$. The RHS can be evaluated with the definition of the detachment velocity in Eq. 2. Combining these simplifications and solving for L_f leads to

$$L_f = \frac{\mu(S)}{K_{\det}} \tag{A.2}$$

which provides the steady-state biofilm thickness.

The particulate concentration in the tank is described by Eq. 1, which can be simplified by assuming steady-state, no source term, and using $v_{\det} = K_{\det} L_f^2 = \mu(S) L_f$ (Eq. A.2) leads to

$$0 = \mu(S) X_t - \frac{Q X_t}{V} + \frac{\mu(S) L_f A X_{b:N_z}}{V}. \tag{A.3}$$

The steady-state substrate concentration can be simplified by taking Eq. 3 and letting the time derivative equal zero and setting the source term to zero. Furthermore, the flux of substrate into the biofilm can be found by assuming $S_b = S$ and then integrating Eq. 6 over the biofilm and using the divergence theorem which leads to

$$S_t^{\mathrm{flux}} = \frac{\mu(S) X_b}{Y_{1,1}} L_f$$

and states that the flux of substrate into the biofilm is equal to the consumption used by the growth of particulates within the biofilm. All together these simplifications change Eq. 3 into

$$0 = -\frac{\mu(S) X_t}{Y_{1,1}} + \frac{Q S_{\mathrm{in}}}{V} - \frac{Q S}{V} - \frac{\mu(S) X_b L_f A}{Y_{1,1} V}$$

which can be rearranged to provide an expression for $X_t(S)$, i.e.,

$$X_t = \frac{Y_{1,1}Q}{\mu(S)V}(S_{\text{in}} - S) - \frac{X_b L_f A}{V} \tag{A.4}$$

To solve for the steady-state solution, an initial guess for the substrate concentration S was made. With this guess, the biofilm thickness L_f can be computed with Eq. A.2, tank particulate concentration with Eq. A.4, and then the residual of Eq. A.3 is evaluated. Based on this residual, the guess for S is updated and the procedure is repeated until the residual becomes less than a tolerance. Running this test shows that the simulated value of the substrate concentration is found to be within 0.004% of the analytic value.

Appendix A.5. Time Integration

This test ensures the time integration is implemented correctly by checking the substrate concentration in the tank with an inflow and outflow. There is a particulate in the simulation, but the growth rate is set to zero so that it does not consume (or produce) substrate. The diffusion coefficients are also set to very small numbers to limit the impacts of substrate entering or leaving the biofilm. With these assumptions, Eq. 3 simplifies to

$$\frac{dS_t}{dt} = \frac{Q}{V}(S_{\text{in}} - S_t) \tag{A.5}$$

which has a solution

$$S_t(t) = S_{\text{in}} + (S_t^0 - S_{\text{in}})e^{-\frac{Q}{V}t}$$

where S_t^0 is the initial condition. Comparing the computed solution to the analytic solution shows that the maximum relative error in the solution is roughly 1e-6.

Appendix B. Case 5 Postprocessing Code

This code was used to produce the plots in Figs. 8 and 9 in Section 5.5. The code can be executed after the simulation for Case 5 is run.

```
# Biofilm quantities
tout = 49.5 # Time when light is on
biofilm_analyze(sol,p,tout,makePlot=true,plotSize=(900,325))
# Could also make plot by directly calling recipe
# biofilm_plot_film(sol([0,tout]),p,size=(900,325))
savefig("Case5_lighton.pdf")

# Times to analyze solution
tout = 48.0:0.01:50.0;
# Get solution at certain times
Xtout,Stout,Lfout = biofilm_analyze(sol,p,tout)

# Function to plot when the light is on
rectangle(w, h, x, y) = Shape(x .+ [0,w,w,0], y .+ [0,0,h,h])
function plot_light()
    plot(rectangle(0.5,1000.0,48.25,0.0),
        opacity=.2,
        fillcolor=:yellow,
        linecolor=:yellow,
        )
    plot!(rectangle(0.5,1000.0,49.25,0.0),
        opacity=.2,
        fillcolor=:yellow,
        linecolor=:yellow,
        )
```

```julia
    end

# Plot tank particulate concentration versus time
plot_light()
plot!(tout,Xtout',
    linecolor=:blue,
    xlabel=("Time [days]"),
    ylabel=("Phototroph Conc. [g/m3]"),
    legend=false,
    size=(300,300),
    ylims=(minimum(Xtout)-0.001,maximum(Xtout)+0.001),
    )
savefig("Case5_Xt.pdf")

# Plot tank substrate concentration versus time
plot_light()
plot!(tout,Stout',
    linecolor=:blue,
    xlabel=("Time [days]"),
    ylabel=("Oxygen Conc. [g/m3]"),
    legend=false,
    size=(300,300),
    ylims=(minimum(Stout)-0.1,maximum(Stout)+0.1),
    )
savefig("Case5_St.pdf")

# Plot biofilm thickness versus time
plot_light()
plot!(tout,1e6.*Lfout,
    linecolor=:blue,
    xlabel=("Time [days]"),
    ylabel=("Biofilm Thickness [um]"),
    legend=false,
    size=(300,300),
    ylims=(minimum(1e6*Lfout)-2,maximum(1e6*Lfout)+2),
    )
savefig("Case5_Lf.pdf")
```

References

[1] G. O'Toole, H. B. Kaplan, R. Kolter, Biofilm Formation as Microbial Development, Annual Review of Microbiology 54 (1) (2000) 49–79, publisher: Annual Reviews. doi:10.1146/annurev.micro.54.1.49.

[2] J. W. Costerton, Z. Lewandowski, D. E. Caldwell, D. R. Korber, H. M. Lappin-Scott, MICROBIAL BIOFILMS, Annual Review of Microbiology 49 (1) (1995) 711–745, publisher: Annual Reviews. doi:10.1146/annurev.mi.49.100195.003431.

[3] S. L. Percival, S. Malic, H. Cruz, D. W. Williams, Introduction to Biofilms, in: S. Percival, D. Knottenbelt, C. Cochrane (Eds.), Biofilms and Veterinary Medicine, Springer Series on Biofilms, Springer, Berlin, Heidelberg, 2011, pp. 41–68. doi:10.1007/978-3-642-21289-5_2.

[4] R. M. Donlan, J. W. Costerton, Biofilms: Survival Mechanisms of Clinically Relevant Microorganisms, Clinical Microbiology Reviews 15 (2) (2002) 167–193, publisher: American Society for Microbiology. doi:10.1128/CMR.15.2.167-193.2002.

[5] L. J. Douglas, Candida biofilms and their role in infection, Trends in Microbiology 11 (1) (2003) 30–36. doi:10.1016/s0966-842x(02)00002-1.

[6] G. Ramage, J. P. Martínez, J. L. López-Ribot, Candida biofilms on implanted biomaterials: a clinically significant problem, FEMS yeast research 6 (7) (2006) 979–986. doi:10.1111/j.1567-1364.2006.00117.x.

[7] P. Stoodley, K. Sauer, D. G. Davies, J. W. Costerton, Biofilms as Complex Differentiated Communities, Annual Review of Microbiology 56 (1) (2002) 187–209, publisher: Annual Reviews. doi:10.1146/annurev.micro.56.012302.160705.

[8] J. W. Wimpenny, R. Colasanti, A unifying hypothesis for the structure of microbial biofilms based on cellular automaton models, FEMS Microbiology Ecology 22 (1) (1997) 1–16, _eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1111/j.1574-6941.1997.tb00351.x. doi:10.1111/j.1574-6941.1997.tb00351.x.

[9] L. A. Lardon, B. V. Merkey, S. Martins, A. Dötsch, C. Picioreanu, J.-U. Kreft, B. F. Smets, iDynoMiCS: next-generation individual-based modelling of biofilms, Environmental Microbiology 13 (9) (2011) 2416–2434. `doi:10.1111/j.1462-2920.2011.02414.x`.

[10] J. U. Kreft, C. Picioreanu, J. W. Wimpenny, M. C. van Loosdrecht, Individual-based modelling of biofilms, Microbiology (Reading, England) 147 (Pt 11) (2001) 2897–2912. `doi:10.1099/00221287-147-11-2897`.

[11] C. Picioreanu, J.-U. Kreft, M. C. M. Van Loosdrecht, Particle-based multidimensional multispecies biofilm model, Applied and Environmental Microbiology 70 (5) (2004) 3024–3040. `doi:10.1128/AEM.70.5.3024-3040.2004`.

[12] J. d. B. Xavier, C. Picioreanu, M. C. M. van Loosdrecht, A general description of detachment for multidimensional modelling of biofilms, Biotechnology and Bioengineering 91 (6) (2005) 651–669. `doi:10.1002/bit.20544`.

[13] E. Alpkvist, I. Klapper, A multidimensional multispecies continuum model for heterogeneous biofilm development, Bulletin of Mathematical Biology 69 (2) (2007) 765–789. `doi:10.1007/s11538-006-9168-7`.

[14] J. D. Chambless, S. M. Hunt, P. S. Stewart, A three-dimensional computer model of four hypothetical mechanisms protecting biofilms from antimicrobials, Applied and Environmental Microbiology 72 (3) (2006) 2005–2013. `doi:10.1128/AEM.72.3.2005-2013.2006`.

[15] J. D. Chambless, P. S. Stewart, A three-dimensional computer model analysis of three hypothetical biofilm detachment mechanisms, Biotechnology and Bioengineering 97 (6) (2007) 1573–1584. `doi:10.1002/bit.21363`.

[16] M. A. S. Chaudhry, S. A. Beg, A Review on the Mathematical Modeling of Biofilm Processes: Advances in Fundamentals of Biofilm Modeling, Chemical Engineering & Technology 21 (9) (1998) 701–710, _eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1002/%28SICI%291521-4125%28199809%2921%3A9%3C701%3A%3AAID-CEAT701%3E3.0.CO%3B2-L. `doi:10.1002/(SICI)1521-4125(199809)21:9<701::AID-CEAT701>3.0.CO;2-L`.

[17] Q. Wang, T. Zhang, Review of mathematical models for biofilms, Solid State Communications 150 (21) (2010) 1009–1022. `doi:10.1016/j.ssc.2010.01.021`.

[18] H. Horn, S. Lackner, Modeling of biofilm systems: a review, Advances in Biochemical Engineering/Biotechnology 146 (2014) 53–76. `doi:10.1007/10_2014_275`.

[19] O. Wanner, W. Gujer, A multispecies biofilm model, Biotechnology and Bioengineering 28 (3) (1986) 314–328. `doi:10.1002/bit.260280304`.

[20] O. Wanner, P. Reichert, Mathematical modeling of mixed-culture biofilms, Biotechnology and Bioengineering 49 (2) (1996) 172–184. `doi:10.1002/(SICI)1097-0290(19960120)49:2<172::AID-BIT6>3.0.CO;2-N`.

[21] J. P. Arcangeli, E. Arvin, Modelling the growth of a methanotrophic biofilm: estimation of parameters and variability, Biodegradation 10 (3) (1999) 177–191. `doi:10.1023/a:1008317906069`.

[22] O. Wanner, E. Morgenroth, Biofilm modeling with AQUASIM, Water Science and Technology: A Journal of the International Association on Water Pollution Research 49 (11-12) (2004) 137–144.

[23] P. S. Stewart, Biofilm accumulation model that predicts antibiotic resistance of Pseudomonas aeruginosa biofilms., Antimicrobial Agents and Chemotherapy 38 (5) (1994) 1052–1058.

[24] M. E. Roberts, P. S. Stewart, Modeling Antibiotic Tolerance in Biofilms by Accounting for Nutrient Limitation, Antimicrobial Agents and Chemotherapy 48 (1) (2004) 48–52. `doi:10.1128/AAC.48.1.48-52.2004`.

[25] M. E. Roberts, P. S. Stewart, Modelling protection from antimicrobial agents in biofilms through the formation of persister cells, Microbiology (Reading, England) 151 (Pt 1) (2005) 75–80. `doi:10.1099/mic.0.27385-0`.

[26] P. S. Stewart, M. A. Hamilton, B. R. Goldstein, B. T. Schneider, Modeling biocide action against biofilms, Biotechnology and Bioengineering 49 (4) (1996) 445–455. `doi:10.1002/(SICI)1097-0290(19960220)49:4<445::AID-BIT12>3.0.CO;2-9`.

[27] D. L. Beaudoin, J. D. Bryers, A. B. Cunningham, S. W. Peretti, Mobilization of broad host range plasmid from Pseudomonas putida to established biofilm of Bacillus azotoformans. II. Modeling, Biotechnology and Bioengineering 57 (3) (1998) 280–286.

[28] J. Bezanson, A. Edelman, S. Karpinski, V. B. Shah, Julia: A Fresh Approach to Numerical Computing, SIAM Review 59 (1) (2017) 65–98, publisher: Society for Industrial and Applied Mathematics. `doi:10.1137/141000671`.

[29] C. Rackauckas, Q. NIE, DifferentialEquations.jl – A Performant and Feature-Rich Ecosystem for Solving Differential Equations in Julia, Journal of Open Research Software 5 (1) (2017) 15, number: 1 Publisher: Ubiquity Press. `doi:10.5334/jors.151`.

[30] J. Monod, The Growth of Bacterial Cultures, Annual Review of Microbiology 3 (1) (1949) 371–394. `doi:10.1146/annurev.mi.03.100149.002103`.

[31] T. Zhang, B. Pabst, I. Klapper, P. S. Stewart, General Theory for Integrated Analysis of Growth, Gene, and Protein Expression in Biofilms, PLOS ONE 8 (12) (2013) e83626, publisher: Public Library of Science. `doi:10.1371/journal.pone.0083626`.

[32] C. G. Hill, T. W. Root, Introduction to Chemical Engineering Kinetics and Reactor Design, 2nd Edition, Wiley, Hoboken, New Jersey, 2014, pp. 381–382.