Biofilms: United They Stand, Divided They Colonize

By Angela B. Shiflet and George W. Shiflet
Wofford College, Spartanburg, South Carolina

1. Scientific Question

Introduction

What do stones in streams, teeth, water and sewer pipes, and the breathing passages of cystic fibrosis patients have in common? At first, these may seem rather unrelated, but they are all linked by at least one commonality—all are covered or lined with biofilms. These assemblages may not be very familiar to you, but they prove absolutely critical to your life.

Scientists have been aware of biofilms for some time. Anton van Leeuwenhoek, who invented and handcrafted microscopes during the late 17th and early 18th centuries, saw remnants of a biofilm when he observed scrapings he made from his teeth (Donlan and Costerton 2002). We just haven't appreciated their importance until recently.

What is a biofilm, exactly? Simply, **biofilms** are communities of very small organisms that adhere to a surface (**substratum**) in an aqueous environment (Donlan and Costerton 2002). These organisms are usually bacteria, but algae or fungi may also form biofilms. Sometimes, these groups may even be mixed. The organisms are not only attached to a substratum, but are linked with each other within a matrix of biopolymers (polysaccharides, proteins, lipids, and nucleic acids). It is now believed that the vast majority of microbes are not solitary, **planktonic**, as was once assumed. Most microbial life seems to be part of one of these communities, and the planktonic forms may be simply ways to colonize other surfaces. We also know now that the free-floating members of each species are phenotypically quite different from their socially connected counterparts (Boles et al. 2004).

When van Leeuwenhoek looked at the "animalicules" from his dental scrapings, what he was actually seeing is what we now call "plaque." Dental plaque is a fitting example of a biofilm (Overman 2000). Forming on the surfaces of the teeth and soft tissues of the oral cavity, it is linked to dental caries (cavities), periodontal diseases, and even cardiovascular disease (Genco et al. 2002). Examination of plaque reveals a complex architecture with a heterogeneous array and dispersal of cells within a matrix associated with fluid-filled spaces. The cells are mostly bacteria belonging to as many as 500 distinct species, but there are usually white blood cells and some epithelial cells, as well.

Immediately after you have your teeth cleaned, a glycoprotein coating, derived from saliva, coats your teeth. This coating is called the **pellicle** and includes some protective measures like lysozyme and antibodies (immunoglobulin A). Despite these measures, the pellicle is colonized quickly by a variety of bacteria, including *Streptococcus mutans*. These pioneers will proliferate, and other species will also adhere. If plaque is allowed to accumulate, a diverse community of bacteria develops, where the intra- and interspecies interactions become quite elaborate. Nutrients are procured and processed from the oral

cavity. Resulting metabolic products and secretions from one bacterium may serve as growth factors, nutrients, or signals for others in the community.

Bacteria associated with plaque are busy producing toxins and various metabolites. Some of these metabolites are organic acids, like lactic acid, which initiate the formation of caries by demineralization of the enamel. *Streptococcus mutans* is often implicated in the process, but there are a number of other acid-producing bacteria that contribute. After the weakening and penetration of the enamel, other bacteria producing extracellular enzymes extend the damage to the now-vulnerable dentine and cementum.

The presence of microbial biofilms can also give rise to inflammation and swelling of the margins between the teeth and gums. The production of enzymes, toxins, and other metabolites can cause a deeper deterioration of the support structures of the teeth. Severe periodontal disease is the leading cause of tooth loss in adults.

Biofilms are ubiquitous, and dental plaque is only one example. Given their prevalence, there must be some advantages for microbes to band together into such communities. In fact, there is quite a list of advantages. For a human pathogen like *Pseudomonas aeruginosa*, a common cause of respiratory diseases, living in biofilms greatly increases the success of infection (Boles et al. 2004). Biofilms afford greatly increased protection from antibiotics, the host's immune system, and physical injury. Scientists have discovered that biofilm organisms show great genetic diversity, which also stabilizes the community and promotes survival in the hosts (Boles et al. 2004). This is in conformance with the well-known ecological principle of the "insurance hypothesis"—diverse subpopulations increase the chances of survival of the community over a wider range of environmental conditions.

With the close association of biofilm constituents, there is the additional opportunity to share metabolites. Furthermore, such organisms are better able to communicate, coordinate behavior, and transfer genetic information (Harrison et al. 2005).

According to Costerton, 65-80% of all bacterial diseases in human beings are from chronic biofilm infections (Costerton 1999; Costerton 2004). For years, most physicians and scientists conceived of bacterial diseases derived from the single-celled, planktonic form of the microbe. Medical treatment was gauged to combat such forms, not biofilms. So, it is little wonder that we are finding increasingly difficulties in combating pathogens.

For the most part, biofilms seem to be a threat to our species, although we are beginning to utilize them in positive ways for bioremediation and wastewater treatment for "green" buildings. Because these communities have such important impacts on our lives, it behooves us to better understand the structure and function of biofilms (Stewart 2003).

The Problem

In this module, we simulate the formation of the structure of a biofilm without regard to its function. Starting with an initial development in two dimensions (2D), we later extend the simulation to three dimensions (3D). As the simulation time proceeds in a sequence of discrete steps, we consider the following phases at each time step:

1. Diffusion of nutrients
2. Growth and death of microbes
3. Consumption of nutrients by microbes

The projects consider additional phases, such as diffusion and release of microbial products, attachment to the biofilm of a microbe that is wandering in free space, and detachment of microbes from the biofilm. For simplicity, we consider the biofilm to be composed of only one type of bacteria, while projects at the end of this module consider more complex arrangements.

2. Computational Models

Cellular Automaton Simulation

One way to look at the world is to study a process as a group of smaller pieces (or cells or sites) that are somehow related. Each piece corresponds to an area (or volume) in the world. Each piece can be associated with one of several possible states at any given time. One convenient way to lay out the world is as a rectangular grid of cells. Rules specify how a cell changes state over time based on the states of the cells around it.

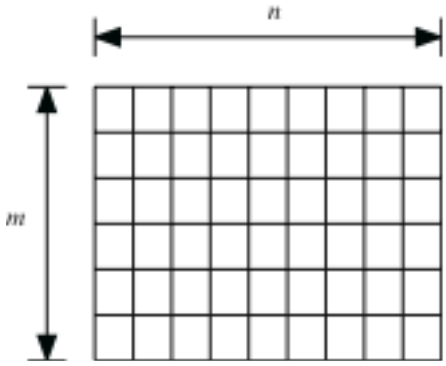
A computer simulation involving such a system is a cellular automaton. **Cellular automata** are dynamic computational models that are discrete in space, state, and time. We picture space as a one-, two-, or three-dimensional grid (also sometimes called an **array** or **lattice**). A **site** (or **cell**) of the grid has a state, and the number of states is finite. **Rules** (or **transition rules**), specifying local relationships and indicating how cells are to change state, regulate the behavior of the system. An advantage of such grid-based models is that we can visualize through informative animations the progress of events. For example, we can view a simulation of the movement of ants toward a food source, the propagation of infectious diseases, heat diffusion, distribution of pollution, the motion of gas molecules in a container, or the growth of a biofilm.

Diffusion Model

In modeling the nutrients in the system, we assume that we have a homogeneous nutrient that is completely mixed at a constant temperature. Although one of the projects considers another alternative, for now we assume that the nutrients diffuse at the same rate throughout the system.

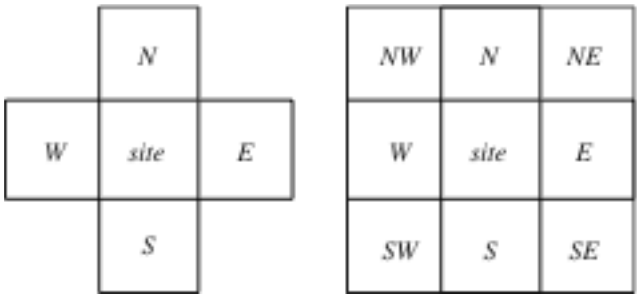
As with many simulations, we model the dynamic area under consideration with an m-by-n grid (or $m \times n$ grid), or lattice or a two-dimensional array, of numbers (see Figure 1). Each cell in the lattice contains a value representing a characteristic of a corresponding location.

Figure 1 Cells to model area



The state of a cell often depends on the states of its four or eight nearest neighbors, as in Figure 2. For diffusion of a nutrient through the grid, we use eight neighbors.

Figure 2 Cells that determine a site's next value



We base our model of diffusion on **Newton's Law of Heating and Cooling**, which states that the rate of change of the temperature with respect to time of an object is proportional to the difference between the temperature of the object and the temperature of its surroundings. Similarly, we can say that the change in a cell's nutrient value, $\Delta site$, from time t to time $t + \Delta t$ is a **diffusion rate parameter** (r) times the sum of each difference in the nutrient value of a neighbor ($neighbor_i$) and the cell's nutrient value ($site$), as follows:

$$\Delta site = r \sum_{i=1}^{8} (neighbor_i - site), \text{ where } 0 < r < 1/8 = 0.125$$

Thus, the nutrient value at time $t + \Delta t$ is the following:

$$site + \Delta site = site + r \sum_{i=1}^{8} (neighbor_i - site),$$

where $0 < r < 0.125$ and the sum is over the eight neighbors. With subtraction of $r(site)$ occurring 8 times, the formula simplifies to the following weighted sum of nutrient values of the cell and its neighbors:

$$(1 - 8r)site + r \sum_{i=1}^{8} neighbor_i, \quad \text{where } 0 < r < 0.125$$

Similar diffusion formulas can have smaller coefficients for the corners than for the north, east, south, and west neighbors.

Quick Review Question 1 Suppose the diffusion rate parameter is 0.1 and the nutrient values in the cells are as in Figure 3. Calculate the nutrient value in the center cell at the next time step.

Figure 3 Nutrition values for a grid section for Quick Review Question 1

0.2	0.3	0.4
0.0	0.5	0.6
0.3	0.3	0.7

Boundary Conditions

We must be able to apply the diffusion formula to every grid point, such as in Figure 1, including those on the boundaries of the first and last rows and the first and last columns. However, the diffusion formula has parameters for the grid point (*site*) and its eight nearest neighbors. Thus, we extend the boundaries by one cell, creating what we call **ghost cells**. Several choices exist for values in those cells:

- Give every extended boundary cell a constant value, as indicated in gray in Figure 4. For a value of 0, the boundary insulates. We call this situation **fixed boundary conditions**. In the case of the spread of nutrient, the boundary is similar to a surface or an area with no nutrient. For an infinite reservoir of a nutrient, we might make every boundary cell have a nutrient value, say of 0.2.

- Give every extended boundary cell the value of its immediate neighbor, which we call **reflecting boundary conditions**. Thus, the values on the original first row occur again on the new first row, which serves as a boundary. Similar situations occur on the last row and the first and last columns. (See Figure 5.) In the case of the spread of nutrient, the boundary tends to propagate the current local situation.

- Wrap around the north-south values and the east-west values in a fashion similar to a donut, or a torus. Extend the north boundary row with a copy of the original south boundary row, and extend the south boundary with a copy of the original north boundary row. Similarly, expand the column boundaries on the east and west sides. Thus, for a cell on the north boundary, its neighbor to the north is the corresponding cell to the south. (See Figure 6.) Such conditions are called **periodic boundary conditions**, and the area is a closed environment with the situation at one boundary effecting its opposite boundary cells. Periodic boundary conditions tend to minimize the effects that the finite size of the grid causes.

Figure 4 Grid with extended boundaries with each cell on an extended boundary having a constant value

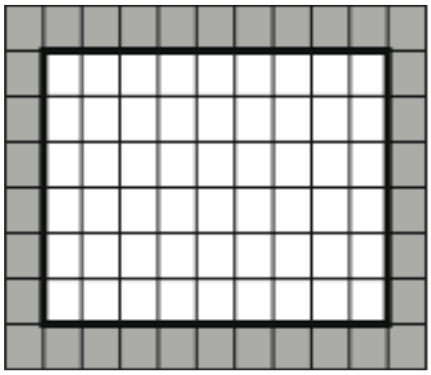


Figure 5 Grid with extended boundaries with each cell on an extended boundary having the value of its immediate neighbor in the original grid. Extension shown in two steps.

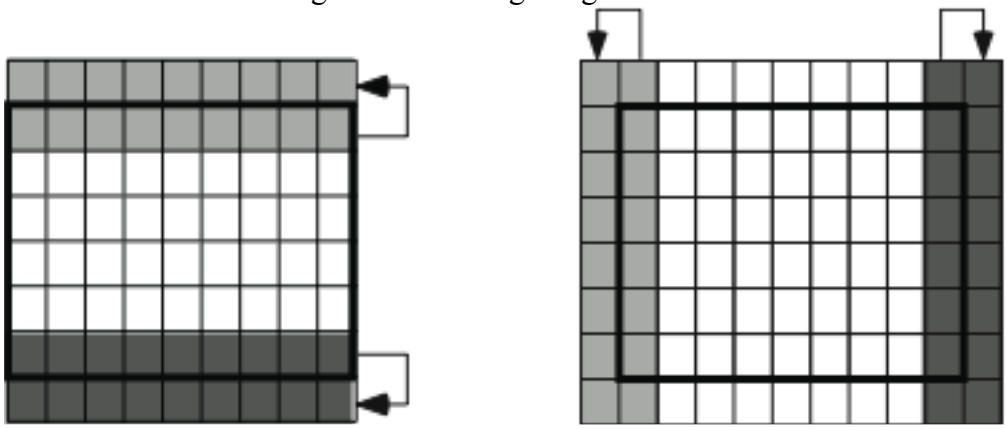
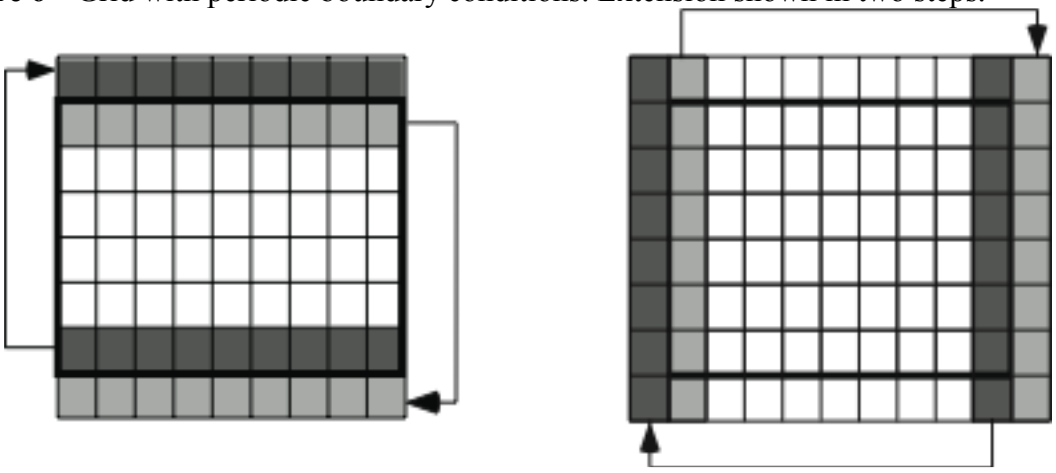


Figure 6 Grid with periodic boundary conditions. Extension shown in two steps.



Quick Review Question 2 Answer the following questions about Figure 3 as an extremely small entire nutrient grid.

a. Give the size of the grid extended to accommodate boundary conditions.

b. Give the values in the first row of the extended matrix assuming fixed boundary conditions with fixed value 0.
c. Give the values in the first row of the extended matrix assuming reflecting boundary conditions, depending on whether we copy rows or columns first.
d. Give the values in the first row of the extended matrix assuming periodic boundary conditions, depending on whether we copy rows or columns first.

In the biofilms model, we employ a combination of boundary conditions. Suppose that the surface to which the biofilm adheres, or substratum, is on the left and an infinite supply of nutrients occurs on the right. For this infinite supply, the expanded nutrient grid has an east-most (right) column with each cell having constant nutrient value. In this same grid with no nutrients present on the surface, we have a west-most column of all zeros. We use periodic boundary conditions in the north and south directions so that part of the nutrient in the north diffuses to the south and vice versa.

Biofilm Growth

For modeling the bacteria in biofilms, we employ an identically shaped grid to that of the nutrient grid, and cells in the same position in the two grids represent the same location. For example, the cell in row 3 and column 7 of the bacteria grid indicates the bacterial state (empty, bacterium, dead bacterium), while the corresponding cell in the expanded nutrition grid indicates the amount of nutrients there.

A cell of the bacteria grid can be in one of three states: contain a live bacterium, contain a dead bacterium, or be empty. As with the nutrient grid, we have periodic boundary conditions in the north-south direction. In an extended bacteria grid, the far west (left) direction has an edge of border cells indicating the substratum, and the far east (right) direction also has an edge of border cells that do not accommodate growth from the interior.

If a location with a bacterium has no nutrition, the bacterium dies of starvation. Cells with dead bacteria remain in that state from one time step to the next. In the projects, we consider other possibilities, such as decay of a dead bacterium to nutrients.

With a certain probability a live bacterium divides at random into a neighboring empty cell. Researchers have considered several calculations of the probability of such growth, usually related to the amount of available nutrition. One method is to multiply a positive constant $p \le 1$ by the nutritional value of the bacterium's cell divided by the sum of the nutritional values of all cells with bacteria:

$$p\left(\frac{cell's\ nutrition}{\sum_i nutrition_i}\right)$$, where the sum is over all cells with live bacteria and $0 < p \le 1$

For example, suppose $p = 0.4$ and three cells have bacteria with corresponding nutrition values of 0.7, 0.5, and 0.8. The probability of the first bacterium dividing is

$$0.4\left(\frac{0.7}{0.7+0.5+0.8}\right) = 0.14 = 14\%$$

Quick Review Question 3 Consider Figure 3 as a complete nutrition grid. Suppose the corresponding bacteria grid has bacteria in the following row-column cells: (1, 1), (1, 2),

(2, 1), (2, 2), (3, 3). Locations (1, 3) and (3, 2) have dead bacterium, while the two other cells are empty.

- **a.** For a *p* value of 0.8, using the above formula, calculate the probability that the bacterium at location (2,2) will divide.
- **b.** If division does occur into one of the four nearest neighbors, give the candidate locations for its daughter bacterium.
- **c.** In the bacteria grid, give the state of the cell at (2, 1) at the next time step.

Consumption of Nutrients

At each time step, a bacterium consumes a constant amount (***CONSUMED***) of nutrient. The amount of nutrient in a cell cannot fall below 0.

Quick Review Question 4 Suppose *CONSUMED* = 0.2 for the nutrition grid in Figure 3, and bacteria are in the following row-column locations: (1, 1), (1, 2), (2, 2), (3, 1). Give the nutrition values after consumption in one time step.

3. Algorithms

Diffusion Algorithm

Diffusion occurs on the nutrient grid at each time step. We initialize this grid to be an *m*-by-*n* matrix with each element having a dimensionless constant value, ***MAXNUTRIENT***. For ease of visualization, we have $0 < MAXNUTRIENT \leq 1$.

***initNutrientGrid*(*m*, *n*)**
Function to return an *m*-by-*n* matrix with each element having the value *MAXNUTRIENT*

The function ***diffusion*** takes the diffusion rate (*diffusionRate*) and the nutrient values of a cell (*site*) and its eight neighbors (*N*, *NE*, *E*, *SE*, *S*, *SW*, *W*, *NW*) using the computation indicated in the "Diffusion Model" section.

***diffusion*(*diffusionRate*, *site*, *N*, *NE*, *E*, *SE*, *S*, *SW*, *W*, *NW*)**
Function to return the new nutrition value of a cell
Algorithm:
return $(1 - 8diffusionRate)site +$
$diffusionRate(N + NE + E + SE + S + SW + W + NW)$

For calculation of new values along the edges, we must extend the boundaries of a grid. As indicated in the "Boundary Conditions" section, we have periodic boundary conditions in the north-south directions, constant 0 in the west direction containing the substratum, and constant *MAXNUTRIENT* in the east direction with its endless nutrient supply. The function ***extendNutrientGrid*** takes an *m*-by-*n* matrix, *mat*, and returns such an extended $(m + 2)$-by-$(n + 2)$ matrix.

***extendNutrientGrid*(*mat*)**
Function to take an *m*-by-*n* matrix parameter and return an (*m* + 2)-by-(*n* + 2) matrix with periodic boundary conditions in the north-south directions, a first column of zeros, and a last column with constant value *MAXNUTRIENT*
Algorithm:
matNS ← concatenation of last row of *mat*, *mat*, and first row of *mat*
return concatenation of column of zeros, *matNS*, and column of *MAXNUTRIENT*'s

After extending the grid by one cell in each direction using these boundary conditions, we apply the function *diffusion* to each internal cell and then discard the boundary cells. To do so, we define a function ***applyDiffusionExtended*** that takes an extended square lattice (*matExt*) and returns the internal lattice with *diffusion* applied to each site. Figure 7 depicts an extended grid with the internal grid, which is a copy of the original lattice, as white. With row and column numbering starting at 1, the number of rows (*m*) and columns (*n*) of the returned lattice are two less than the number of rows and columns of *matExt*, respectively. We apply the function *diffusion*, which has parameters *diffusionRate*, *site*, *N*, *NE*, *E*, *SE*, *S*, *SW*, *W*, and *NW*, to each internal cell in lattice *matExt*. These internal cells are in rows 2 through *m* + 1 and columns 2 through *n* + 1. We added the boundary rows and columns to eliminate the different cases for cells with missing neighbors. Thus, for *i* going from 2 through *m* + 1 and for *j* going from 2 through *n* + 1, *applyDiffusionExtended* obtains a cell value for a new *m* × *n* lattice as the application of *diffusion* to a site with coordinates *i* and *j* and its neighbors with corresponding coordinates as in Figure 8.

***applyDiffusionExtended*(*matExt*, *diffusionRate*)**
Function to accept an extended matrix and diffusion rate and to return an internal matrix with the value of each element being the value returned by *diffusion*
Algorithm:
m ← (number of rows of *matExt*) - 2
n ← (number of columns of *matExt*) - 2
for *i* going from 2 through *m* + 1
    for *j* going from 2 through *n* + 1
        assign to *site*, *N*, *NE*, *E*, *SE*, *S*, *SW*, *W*, *NW* the values from *matExt*
            as indicated in Figure 8
        mat(*i*, *j*) ← *diffusion*(*diffusionRate*, *site*, *N*, *NE*, *E*, *SE*, *S*, *SW*, *W*, *NW*)
return *mat*

Figure 7 Internal grid in color that is a copy of the original grid (see Figure 1) embedded in an extended grid

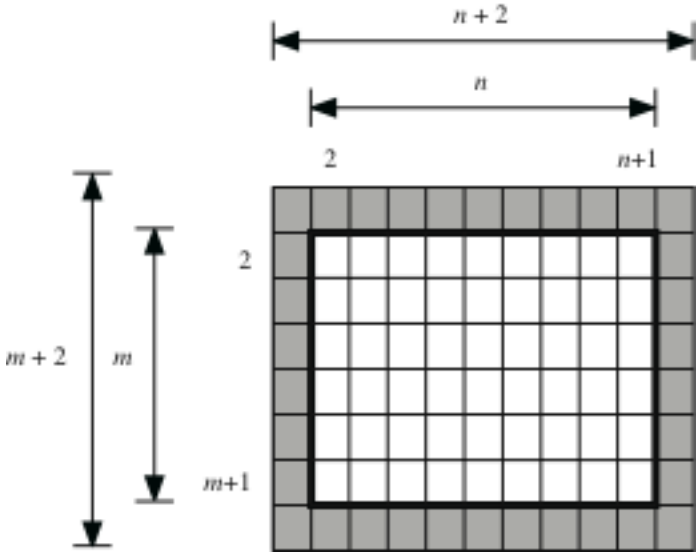


Figure 8 Indices for a lattice site and its neighbors

NW $(i-1, j-1)$	*N* $(i-1, j)$	*NE* $(i-1, j+1)$
W $(i, j-1)$	*site* (i, j)	*E* $(i, j+1)$
SW $(i+1, j-1)$	*S* $(i+1, j)$	*SE* $(i+1, j+1)$

Quick Review Question 5 Suppose *extMat* is an extended matrix of size 97-by-62.

- **a.** Give the size of the matrix *applyDiffusionExtended* returns.
- **b.** In the algorithm for *applyDiffusionExtended*, give the range of rows that i goes through for this matrix.
- **c.** When $i = 33$ and $j = 25$, give the indices of the site's neighbor to the north.
- **d.** For this site, give the indices of its neighbor to the southwest.

Growth Algorithm

To represent the four states of a cell in the extended bacteria grid, we employ the following values: 0 to indicate an empty cell with no bacterium, 1 for a cell with a bacterium, 2 to represent a cell with a dead bacterium, and 3 for a border cell. Table 1 lists these values and meanings along with associated names ***EMPTY***, ***BACTERIUM***, ***DEAD***, or ***BORDER*** that have values of 0, 1, 2, and 3, respectively. We initialize these constants at beginning of the program and employ the descriptive names throughout the program. Thus, the code is easier to understand and to change.

Table 1 Cell values with associated constants and their meanings

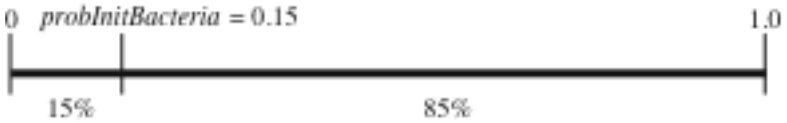
Value	Constant	Meaning
0	*EMPTY*	The cell does not contain a live or dead bacterium or border.
1	*BACTERIUM*	The cell contains a live bacterium.
2	*DEAD*	The cell contains a dead bacterium.
3	*BORDER*	The cell is on the border and not under active consideration.

To initialize a grid for a simulation, for each cell we must designate if the location contains a bacterium or not. We can form this initial configuration in a variety of ways to study various situations. For this simulation, the initial bacteria grid is an *m*-by-*n* matrix with bacteria (value *BACTERIUM*) occurring at random in the first column and all other cells being empty (value *EMPTY*).

The initialization algorithm employs random numbers and probability in determining the first column's values. Suppose only 15% of these cells contain bacteria. Thus, a *probInitBacteria* = 0.15, or 15%, chance exists for a cell to contain a bacterium. If the location is to contain a bacterium, we make the cell's value equal to *BACTERIUM* (value = 1); and otherwise, the cell's value becomes *EMPTY* (value = 0). For each cell, we generate a uniformly distributed random floating point number from 0.0 up to 1.0. On the average, 15% of the time this random number is less than 0.15, while 85% of the time the number is greater than or equal to 0.15 (Figure 9). Thus, to initialize the cell, if the random number is less than 0.15, we make the cell's value *BACTERIUM*; otherwise, we assign *EMPTY* to the cell's value. Thus, using the probability and cell values above, we employ the following logic to initialize each cell in the grid:

***initBacteriaGrid*(*m*, *n*, *probInitBacteria*)**
Function to return an initial *m*-by-*n* bacteria grid of all *EMPTY* values except for a first column, where the probability of a bacterium in a cell is *probInitBacteria*
Algorithm:
emptyMat ← *m*-by-(*n* - 1) matrix with each cell being *EMPTY*
onSurface ← *m*-by-1 matrix (column vector) with each element calculated as follows:
if a random floating point number is less than *probInitBacteria*
set the cell's value to *BACTERIUM*
else
set the cell's value to *EMPTY*
return *m*-by-*n* matrix with *onSurface* as first column and *emptyMat* as rest of matrix

Figure 9 15% of floating point values between 0 and 1 are less than *probInitBacteria* = 0.15



Quick Review Question 6 Consider the following pseudocode that returns the direction (*N*, *E*, *S*, or *W*) a simulated animal moves:

```
if a random number, rand, is < 0.12
    return N
else if rand < 0.26
    return E
else if rand < 0.69
    return S
else
    return W
```

Give the probability that the animal moves in each of the following directions:
a. *N* **b.** *E* **c.** *S* **d.** *W*

For growth of a bacterium, we must examine its neighbors. Thus, as indicated above and designed below, we use periodic boundary conditions in the north-south directions, a first column with each cell having the value *BORDER* indicating a surface to the west, and a last column with each cell having the value *BORDER* so that bacteria do not grow to the east.

***extendBacteriaGrid*(*mat*)**
Function to take an *m*-by-*n* matrix parameter and return an (*m* + 2)-by-(*n* + 2) matrix with periodic boundary conditions in the north-south directions and with fixed boundary conditions in the east-west directions using constant value *BORDER*
Algorithm:
matNS ← concatenation of last row of *mat*, *mat*, and first row of *mat*
return concatenation of column of *BORDER*'s, *matNS*, and column of *BORDER*'s

As stated in the section on "Biofilm Growth," for some constant, $0 < p \le 1$, our simulation calculates the probability that a cell will divide as follows:

$$p\left(\frac{cell's\ nutrition}{\sum_i nutrition_i}\right) = (cell's\ nutrition)\left(\frac{p}{\sum_i nutrition_i}\right),$$

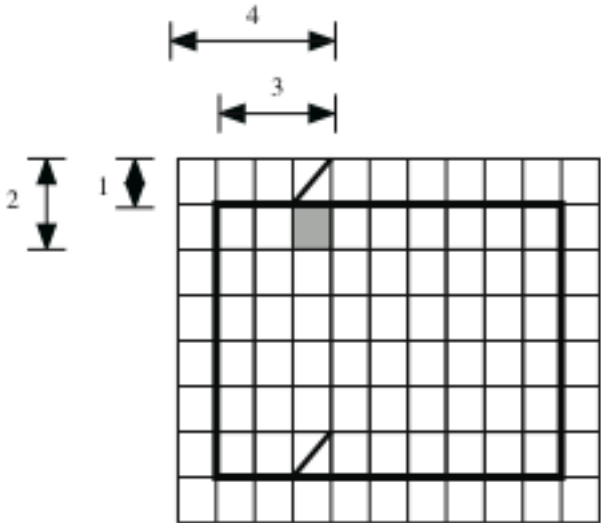
where the sum is over all cells with live bacteria. Because the last term, $p \Big/ \sum_i nutrition_i$, is the same for all cells, we define a function to return its value. However, if the grid does not contain any bacteria, we are careful not to divide by zero and return zero instead.

***probGrow*(*bacteriaGrid*, *nutritionGrid*, *p*)**
Function to return $p \Big/ \sum_i nutrition_i$, where the sum is over all cells with live bacteria, or 0 if the grid does not contain live bacteria

For a *BACTERIUM* cell that is to divide, we must pick an empty neighbor to accept the daughter bacterium. While projects consider other alternatives, in this simulation if no empty neighbor exists, division does not occur. However, when possible, we select one of the empty neighbors at random. The function *pickNeighbor* has parameters of a cell's

row (*i*) and column (*j*) in an extended matrix, the number of rows of the corresponding un-extended matrix (*m*), and the values of the (*i*, *j*) cell's four nearest neighbors (*N*, *E*, *S*, *W*). The function returns indices in the corresponding un-extended bacteria grid. Thus, we first define *newi* and *newj* to be the indices in the un-extended grid corresponding to the indices, *i* and *j*, in the extended grid. That is, *newi* and *newj* are one less than *i* and *j*, respectively. If no neighbor (*N*, *E*, *S*, *W*) is empty, we return the pair (*newi*, *newj*) so that division does not occur. Otherwise, we return the row and column in the un-extended grid of the selected empty neighbor. We must be careful to consider north-south period boundary conditions. Thus, an empty cell north of a first row bacteria grid cell is really on grid's last row. For example, suppose $i = 2$, $j = 4$, $m = 6$, and $N = EMTPY$ as in Figure 10. With coordinates (2, 4) in the extended bacteria grid, the corresponding coordinates in the un-extended grid are (*newi*, *newj*) = (1, 3). Suppose the cell to the north is picked to accept the daughter bacteria in division. Wrapping around with periodic boundary conditions, this cell is really far south at location (6, 3) in the un-extended grid. Similarly, we must consider periodic boundary conditions when a cell in the last row of the un-extended grid has a selected empty neighbor to the south. Because the extended grid has a first and a last column of all *BORDER* values, we cannot pick a cell off the un-extended bacteria grid to the west or east, which simplifies the code.

Figure 10 Extended and un-extended bacteria grid with empty cell to north of site next to boundary



***pickNeighbor*(*i*, *j*, *m*, *N*, *E*, *S*, *W*)**

Function to return the row and column in the un-extended bacteria grid of a randomly selected empty neighbor of a given cell. If an empty neighbor does not exist, the function returns the indices of the site in the un-extended bacteria grid corresponding to the given cell.

Pre:

i, *j* - indices of site in extended bacteria grid
m - number of rows of un-extended bacteria grid
N, *E*, *S*, *W* - values of nearest four neighbors of site in extended bacteria grid

Post:

The function has returned the indices in the un-extended bacteria grid of an empty neighbor or, if no such neighbor exists, the indices of the given site in the un-extended bacteria grid

Algorithm:

```
lst ← list of N, E, S, W
pos ← list indices (1 through 4) where EMPTY occurs in lst
newi ← i - 1                    // indices in un-extended grid
newj ← j - 1
if pos has no elements
    return (newi, newj)
else
    rand ← random integer from 1 through the length of pos
    if pos(rand) is 1
        if newi > 1
            return (newi - 1, newj)        // north
        else
            return (m, newj)               // wrap around because of periodic
                                           // boundary conditions
    else if pos(rand) is 2
        return (newi, newj + 1)            // east
    else if pos(rand) is 3
        if newi < m
            return (newi + 1, newj)        // south
        else
            return (1, newj)               // wrap around because of periodic
                                           // boundary conditions
    else
        return (newi, newj - 1)            // west
```

Quick Review Question 7

- **a.** Suppose one call to *pickNeighbor* has arguments 18, 23, 17, *DEAD*, *BORDER*, *BACTERIUM*, *BACTERIUM*. Give the possible coordinate pair(s) it can return.
- **b.** Give the number of rows and columns in the extended grid.
- **c.** Give the number of rows and columns in the un-extended grid.
- **d.** Suppose one call to *pickNeighbor* has arguments 18, 23, 17, *EMPTY*, *BORDER*, *EMPTY*, *EMPTY*. Give the values of *newi* and *newj*.
- **e.** Give the possible coordinate pair(s) it can return.

Preliminary to the main iteration of the growth algorithm, we make a copy (*bacGrid*) of the bacteria grid for updating, determine its number of rows (m) and columns (n), calculate the partial probability $prob = p \Big/ \sum_i nutrition_i$, and expand the bacteria and nutrition grids to account for boundary conditions. Then, looking for bacteria, we iterate through every internal position of the extended bacteria grid (*extBacGrid*) by having an index (i) going from row 2 through row (m + 2) and an index

(*j*) changing from row 2 through row (*n* + 2). If a bacterium has no nutrition (nutrition value of 0), we change the corresponding element of *bacGrid* to be *DEAD*. Because *extBacGrid* is an expanded matrix of size (*m* + 2)-by-(*n* + 2) while *bacGrid* has size *m*-by-*n*, an element of *extBacGrid* with indices *i* and *j* corresponds to an element of *bacGrid* with indices (*i* - 1) and (*j* - 1), respectively. For a position with a bacterium that is to live, we calculate the probability (*prob* times its nutrition value) that the bacterium will divide. A random floating point number between 0 and 1 is less than that probability the corresponding percentage of the time. When this occurs, we call *picNeighbor* to obtain the indices (*newi* and *newj*) of the new daughter bacterium and then change the corresponding element of *bacGrid* from *EMPTY* to *BACTERIUM*.

***grow*(*bacteriaGrid*, *nutritionGrid*, *p*)**

Function to take a bacteria grid, a nutrition grid, and a partial probability *p* and return a bacteria grid for the next time step as follows: If a site with a bacterium has no nutrient, the bacterium dies; otherwise, if possible, the bacterium divides and its daughter bacterium inhabits a randomly selected empty neighboring site.

Algorithm:

```
bacGrid ← bacteriaGrid
m ← number of rows in nutritionGrid
n ← number of columns in nutritionGrid
prob ← probGrow(bacteriaGrid, nutritionGrid, p)
extBacGrid ← extendBacteriaGrid(bacteriaGrid)
extNutGrid ← extendNutrientGrid(nutritionGrid)
for i going from 2 through m + 1, do the following:
    for j going from 2 through n + 1, do the following:
        if extBacGrid(i, j) is BACTERIUM,
            if extNutGrid(i, j) <= 0
                bacGrid(i - 1, j - 1) ← DEAD
            else if a random number is less than prob*extNutGrid(i, j)
                (newi, newj) = pickNeighbor(i, j, m, extBacGrid(i - 1, j),
                                            extBacGrid(i, j + 1), extBacGrid(i + 1, j),
                                            extBacGrid(i, j - 1))
                bacGrid(newi, newj) ← BACTERIUM
return bacGrid
```

Quick Review Question 8 With execution of *grow* on a bacteria grid, give the possible change of state for a cell with value

a. *BACTERIUM*

b. *EMPTY*

Consumption Algorithm

For this simulation, nutrition is only consumed in the cells containing bacteria. In each such cell, a bacterium eats a constant amount (*CONSUMED*) of nutrients, so that the new value for the cell's nutrient is the old value minus *CONSUMED*. However, a bacterium cannot consume more than is there; so that if the difference is negative, we use 0.0

instead. We can employ an *if* statement or can take the maximum of 0.0 and the old nutrient value minus *CONSUMED* to insure that each result is non-negative.

***consumption*(*bacteriaGrid*, *nutritionGrid*)**
Function to return a new nutrition grid after bacteria have consumed nutrients in one time step
Algorithm:
m ← number of rows of *nutritionGrid*
n ← number of columns of *nutritionGrid*
nutGrid ← *nutritionGrid*
for *i* going from 1 through *m*
for *j* going from 1 through *n*
if *bac*(*i*, *j*) is *BACTERIUM*
nutGrid(*i*, *j*) ← maximum of 0.0 and (*nutGrid*(*i*, *j*) - *CONSUMED*)
return *nutGrid*

Quick Review Question 9 Write pseudocode using an *if* statement instead of "maximum" in the nested loops to obtain a new value for *nutGrid*(*i*, *j*).

Simulation Program

To perform the simulation of a biofilm's structural formation, we define a function *biofilm* with parameters *m* and *n*, the number of grid rows and columns, respectively; *probInitBacteria*, the probability of a bacterium in an initial bacteria grid's first column element; *diffusionRate*, the rate of diffusion of nutrients in the nutrient grid; *p*, the constant ($0 < p \leq 1$) used in the calculation of the probability that a bacterium divides; and *t*, the number of time steps. The function *biofilm* returns two lists, a list of the initial bacteria grid and the next *t* bacteria grids in the simulation and a corresponding list of nutrient grids. Pseudocode for *biofilm* is as follows:

***biofilm* (*m*, *n*, *probInitBacteria*, *diffusionRate*, *p*, *t*)**
Function to return a list of bacteria grids and a list of nutrition grids in a simulation of the formation of the structure of a biofilm with one type of bacteria. In a bacteria grid, a cell value of *EMPTY* indicates the cell is empty; *BACERIUM*, the cell contains a live bacterium; and *DEAD*, a dead bacterium. In a nutrition grid, cell values range from 0 (no nutrient) to 1.
Pre:
m, *n* are the number of rows and columns, respectively, of the bacteria and nutrient grids.
probInitBacteria is the probability of a bacterium in an element of the initial bacteria grid's first column.
diffusionRate is the rate of diffusion of nutrients in the nutrient grid.
p is the constant ($0 < p \leq 1$) used in the calculation of the probability that a bacterium divides, (cell's nutrient value) $p \Big/ \sum_i nutrition_i$.
t is the number of time steps.

Post:

Two lists were returned: a list of the initial bacteria grid and the grid at each time step of the simulation and a corresponding list of nutrient grids.

Algorithm:

call *initBacteriaGrid* to initialize *bacteriaGrid* to be an *m*-by-*n* grid of values, *EMPTY* (no bacterium), *BACTERIUM* (live bacterium), and *DEAD* (dead bacterium), where *probInitBacteria* is the probability of a bacterium in an element of the first column
call *initNutrientGrid* to initialize *nutrientGrid* to be an *m*-by-*n* grid of values *MAXNUTRIENT*
bacGrids ← list containing *bacteriaGrid*
nutGrids ← list containing *nutrientGrid*
do the following *t* times:
extNutrientGrid ← call *extendNutrientGrid*(*nutrientGrid*)
nutrientGrid ← call *applyDiffusionExtended* to return new *m*-by-*n* nutrient grid with *diffusion* applied to each internal cell of *extNutrientGrid* with diffusion rate *diffusionRate*
bacteriaGrid ← call *grow* to return new *m*-by-*n* bacteria grid after growth/death of bacteria
nutrientGrid ← call *consumption* to return new *m*-by-*n* nutrition grid after bacteria consume nutrients
bacGrids ← the list with *bacteriaGrid* appended onto the end of *bacGrids*
nutGrids ← the list with *nutrientGrid* appended onto the end of *nutGrids*
return *bacGrids* and *nutGrids*

Quick Review Question 10 For the *biofilm* algorithm, use the text's notation for the function calls to make assignments to the following variables.

a. *bacteriaGrid = initBacteriaGrid*
b. *nutrientGrid = initNutrientGrid*
c. *extNutrientGrid = extendNutrientGrid*(*nutrientGrid*)
d. *nutrientGrid = applyDiffusionExtended*
e. *bacteriaGrid = grow*
f. *nutrientGrid = consumption*

Display Simulation

Visualization helps us understand the meaning of the grids. For each bacteria grid in the first list returned by *biofilm*, we generate a graphic for a rectangular grid with yellow representing an empty site; green, a bacterium; and dark gray, a dead bacterium. The function ***showBacteriaGraphs*** with parameter *graphList* containing the list of lattices from the simulation produces these figures. We animate the sequence of graphics to view the changing biofilm scene.

Because nutrient values are on a continuum from 0 to 1, we employ a grayscale for the animation of nutrient diffusion. On such a scale, 0 is black and 1 is white. So that the

higher nutrient values appear darker, we subtract each nutrient value from 1 to obtain its degree of gray. For example, a low nutrient value of 0.2 converts to a grayscale value of 1 - 0.2 = 0.8, which displays as light gray. In contrast, a high nutrient value of 0.8 has a grayscale value of 0.2 and appears as dark gray in the animation.

4. Software Implementation

Click here to download the simulation (*Biofilm.nb*) in *Mathematica*. This tutorial provides instructions on working in the *Mathematica* notebook interface and the file *Biofilm_Description.pdf* documents the *Mathematica* implementation of this model.

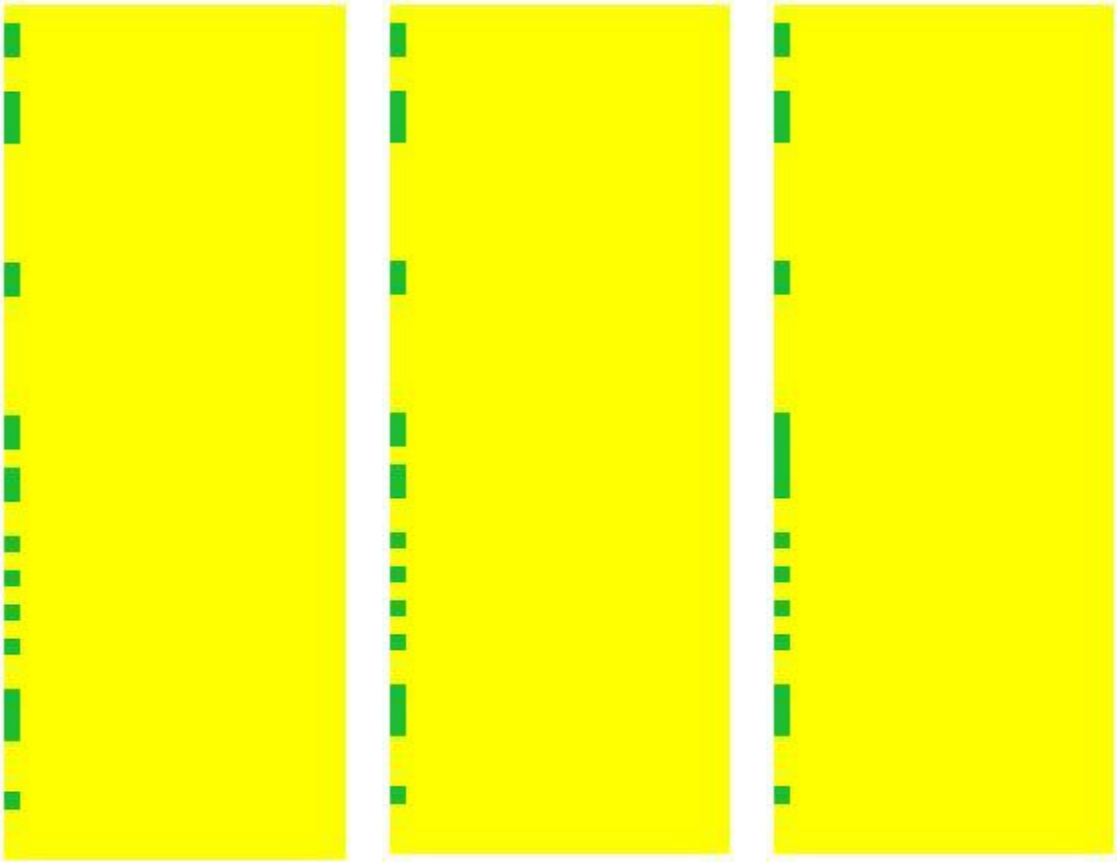
5. Example Problem

Figure 11 displays several frames of a biofilm simulation with bacteria grids on one row and the corresponding nutrient grids on the next. Clearly, different initial seeds result in different sequences. This simulation employs the parameters $m = 50$, $n = 20$, *probInitBacteria* = 0.5, *diffusionRate* = 0.1, $p = 1$, and $t = 125$ and an arbitrary seed of 48 for the random number generator. In the simulation, the biofilm does not grow from $t = 0$ to $t = 1$, but subsequent frames show the biofilm spreading to neighboring cells. The nutrient grids illustrate the bacteria's gradual consumption of food as well as the diffusion of nutrients. Grids for times starting at $t = 55$ reveal the influence of north-south periodic boundary conditions as the biofilm at the top spreads to the bottom of the grid. The frame at $t = 100$ shows how some bacteria have consumed all their resources and died (in dark gray). As time advances, parts of the biofilm coalesce, and bacteria fill holes in the biofilm (see frame at $t = 125$). We must of course be careful not to allow the simulation to run so long that the biofilm reaches the east edge and starts filling in that end.

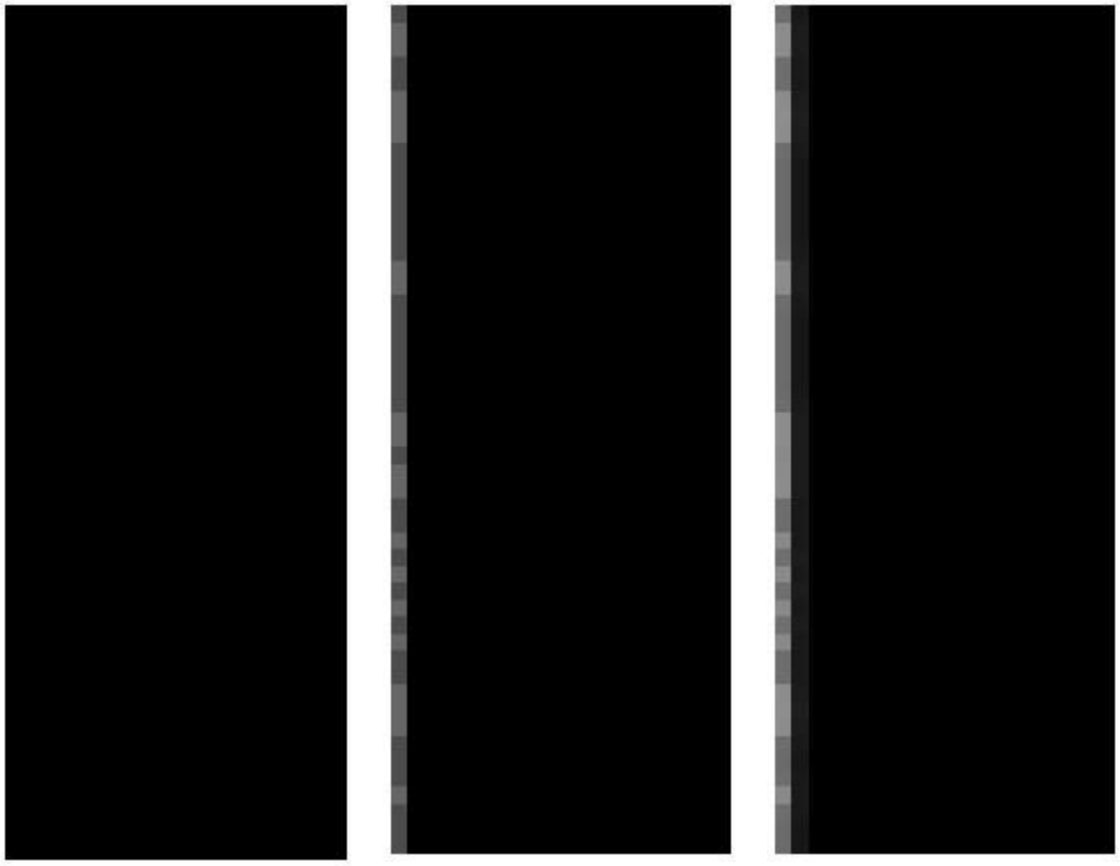
Figure 11 Several frames in an animation of the spreading of a biofilm

$t = 0$ **$t = 1$** **$t = 2$**

Bacteria Grids

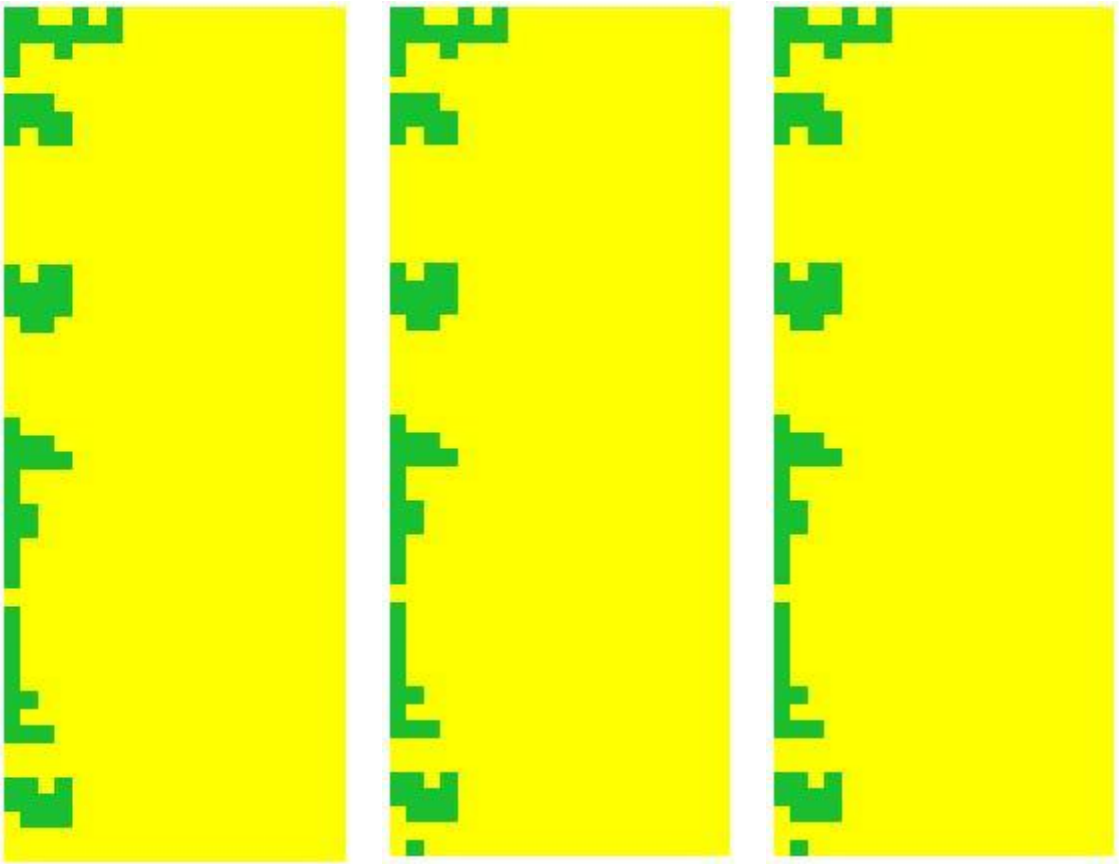


Nutrient Grids

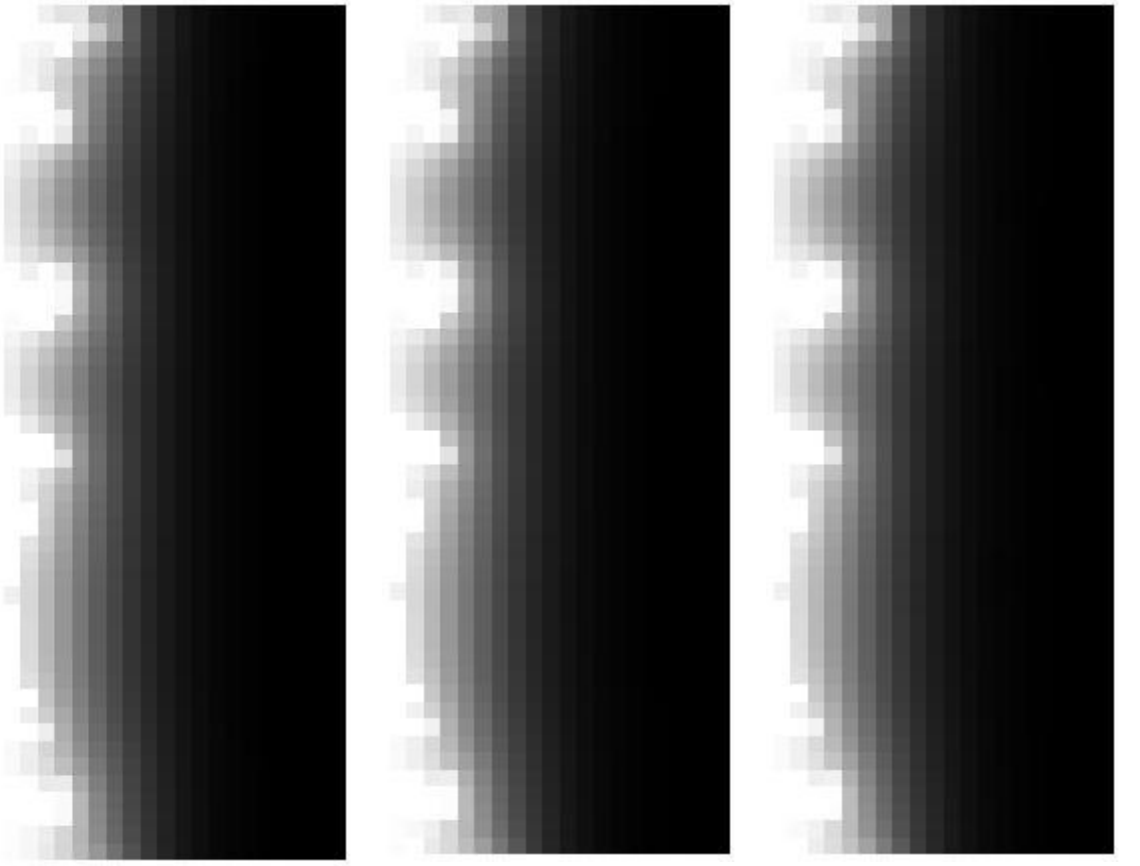


***t* = 54** ***t* = 55** ***t* = 56**

Bacteria Grids

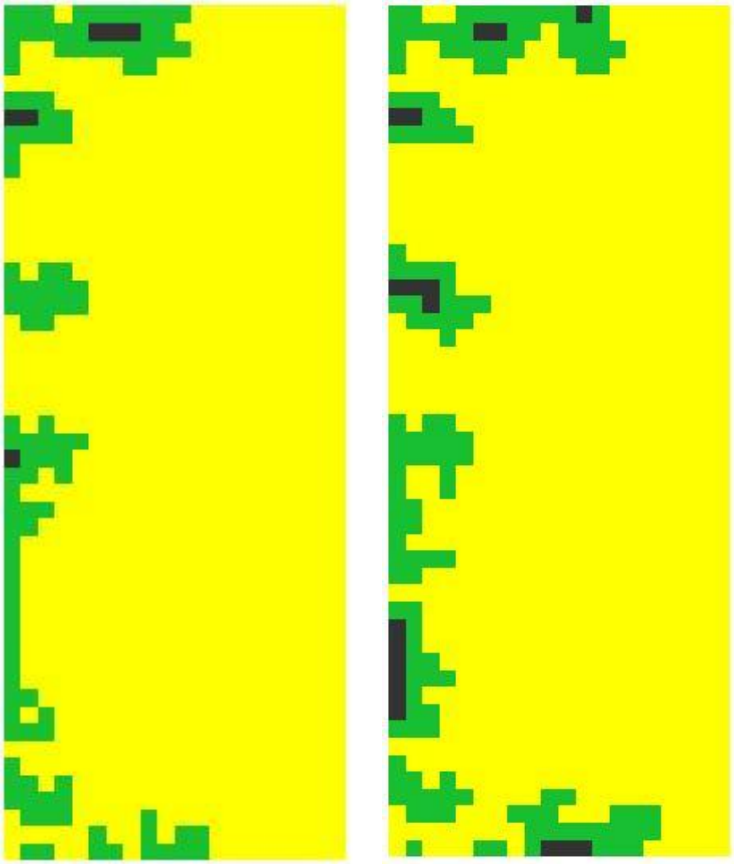


Nutrient Grids

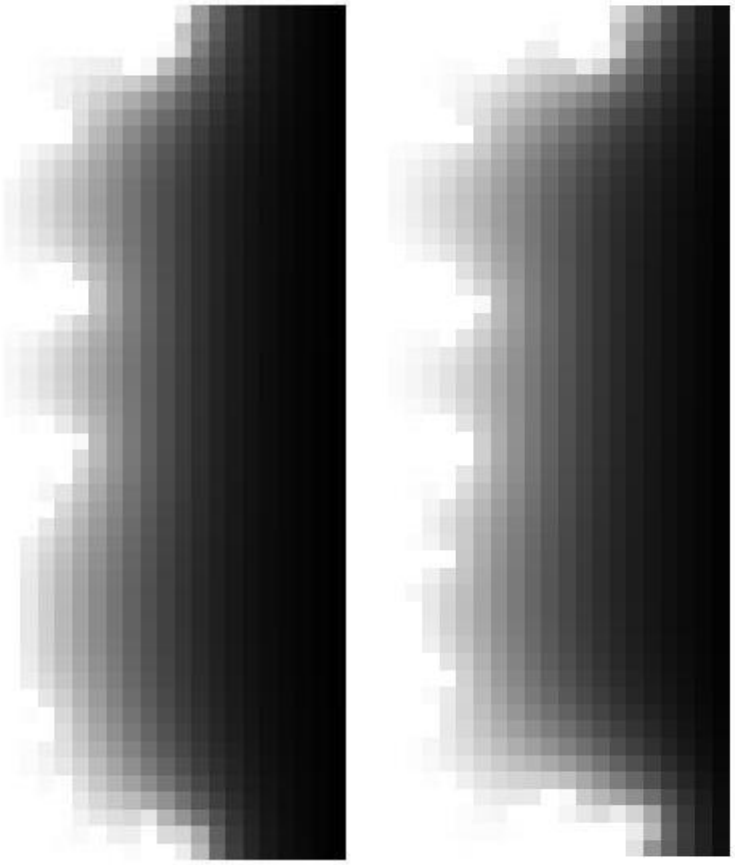


***t* = 100** ***t* = 125**

Bacteria Grids



Nutrient Grids



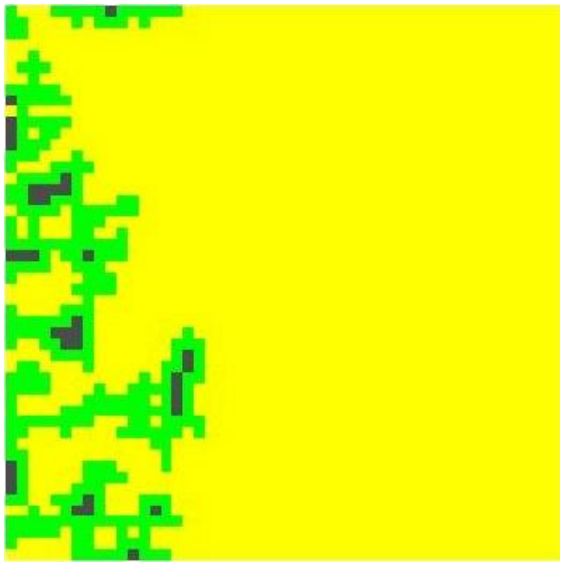
6. Rubric for Assessment

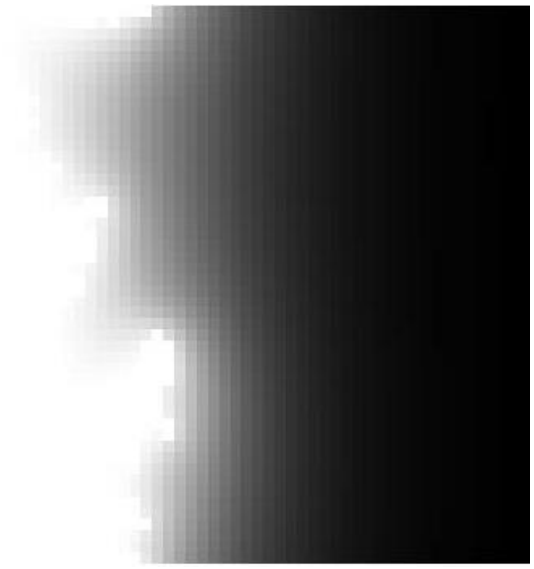
As (IWA 2006) points out, "Most biofilm models today capture only a small fraction of the total complexity of a biofilm system, but they are highly useful." We have chosen in the *biofilm* simulation only to model structural formation, not function. Simulation results agree with various features of biological biofilms. For example, as Figure 11 shows, with time, the overall thickness increases, and inert (dead) areas are greater near the substratum (Laspidou and Rittmann 2004).

As (Schaudinn et al. 2007) so eloquently states, "magnified views reveal microcolonies in an English garden to topiary delights, taking shapes that resemble mushrooms, towers, and arboreal structures…." Researchers have also observed other interesting features, such as pores, in biofilms (Harrison et al. 2005). Frames of simulation results in Figures 11 and 12 display these phenomena. The nutrient grid and both simulation rules based on reality provide explanations for some of these shapes. Bacteria have consumed much of the nutrients towards the substratum in the west; nutrients continually come from the east; and bacteria divide at higher rates in nutrient-rich environments.

Figure 12 Mushroom shapes and pores in simulated biofilm along with corresponding nutrient grid

Bacteria Grid **Nutrient Grid**





However, allowing our simulation to run for many time steps can reveal some anomalies not generally present, such as very long dendritic structures. Refining the model to account for erosion of surface bacteria should ameliorate this situation (see Projects).

Also, the current model does not show water channels present in so many biofilms and does not indicate the biofilm's density, which is significantly greater near the substratum. Moreover, in our simplifying assumptions, we ignored important aspects, such as hydrodynamics, extracellular polymeric substances (EPS), chemical oxygen demand, and heterogeneity. Accounting for such features can enlighten our understanding of biofilms but can result in significantly more complex models that require much greater computing resources. Other types of models, such as continuum or discrete particle-based models, are advantageous for showing such features and for taking into account biofilm function.

7. Computing Power

Download high performance computing biofilm simulations for two, *BiofilmParallelDual.c*, and for multiple processors, *BiofilmParallelMulti.c*, and an accompanying walkthrough, *MPI_Biofilm_Description.doc*, by Shay M. Ellison.

Biofilms are highly complex with numerous features. We have chosen to consider form, not function, in 2D and have employed a number of simplifying assumptions. The simulations were run on a personal computer. However, with larger grids, conversion to 3D, and refinement to more complex models, simulations can stretch computing resources significantly. For example, (IWA 2006) in referring to biofilm models involving hydrodynamics states, "Although such 2d models are now accessible for ordinary personal computers of nowadays (even for time-dependent problems a few minutes may be sufficient), the 3d problems of such type are at limit and better require parallel computing power." Thus, it is advantageous for the modeler to be able to use high performance computing (HPC) when needed. The accompanying tutorial on C and

MPI provide the necessary background to understand the HPC version of the simulation model of this module.

8. Projects

1. Adjust the biofilm simulation to show attachment to the biofilm of bacteria floating in the nutrients. To do so, have *biofilm* execute a fourth phase (after consumption) at each time step. We can use the technique of **diffusion-limited aggregation** (**DLA**) for the attachment. One at a time, "bacteria" are released from random positions on the east boundary (or at least at a random position east of the biofilm) to go on random walks. For each time step of such a walk, a bacteria moves at random to a neighboring position. If the walker comes in contact with another particle (i.e., a neighbor to its north, east, south, or west), with a designated sticking probability, the walker adheres to the particle, resulting in a larger biofilm. If the walker travels too close to the east boundary of the grid, the simulation deletes that walker and releases another random walker. To speed attachment, we can have such a bacterium move eastward with a smaller probability than it moves in the other directions. Notice that such free-floating bacteria only adhere to the surface of the biofilm. For simplicity, define another state and constant for an unattached bacterium, and do not allow a floating bacterium to divide. However, such a bacterium does consume nutrients. Discuss the impact on the structure by allowing attachment. Discuss the effect of consumption of nutrients by a wandering bacterium.

2. One problem with the module's simulation is the formation of long dendritic structures, which do not occur so frequently in biofilms (see Figure 11, $t = 125$). We have not considered the loss of pieces from the biofilm due to erosion, abrasion, grazing, or sloughing (Picioreanu et al. 1996). Adjust the biofilm simulation so that it shows the erosion of surface bacteria. Similar to Project 1, release an inert particle one at a time from a random location east of the biofilm and have the particle go on a random walk. If the particle touches a bacterium (i.e., a neighbor to its north, east, south, or west), remove the bacterium, making its cell empty. Have *biofilm* execute one step of this random walk (after consumption) at each time step. Notice that erosion only occurs on the surface of the biofilm. Discuss the impact of allowing erosion on the structure. Which bacteria are more likely to erode?

3. Develop an alternative to the biofilm detachment model of Project 2 by eliminating any bacterium above a designated height from the surface. An example of such situation is a constant-depth film fermentor, a device that periodically removes the surface growth to maintain a biofilm with a constant geometry. Researchers use this system to grow and study oral biofilms (dental plaque) in the laboratory (Picioreanu et al. 2004; UCL Eastman 2008). Run the simulation long enough to observe the pores gradually filling and formation of a compact biofilm.

4. Expand the biofilms simulation to have both an attachment phase (see Project 1) and an erosion phase (see Project 2 or 3) at each time step. Discuss the results.

5. Revise the *diffusion* algorithm so that diffusion into a site is less likely to occur from its corner neighbors. Using the values in Table 2 for the returned value of *diffusion*, calculate the sum of the products of nutrition grid cells and corresponding table values. Thus, the new nutrition value of the site is 25% of its old nutrition value; plus 12.5% of each of the cells to the north, east, south, and west; plus 6.25% of the each of the corner cells. Notice that the sum of these percentages is 100%. This sum is called a **weighted sum** with each nutrition value carrying a particular weight as indicated by the table. Discuss any differences in diffusion and biofilm growth between this model and that of the module.

Table 2 Table for Project 5

0.0625	0.125	0.0625
0.125	0.25	0.125
0.0625	0.125	0.0625

6. One classical growth model has a bacterium that is to divide dying if its cell has no empty neighbors. Develop this model and compare the results to *biofilm* of this module.

7. The model in this module has a bacterium only consuming nutrition from its own site. However, it is reasonable to consider that the bacterium might consume some nutrition nearby. Adjust *consumption* so that a bacterium consumes a proportion of the nutrition from its own site and smaller proportions of nutrition from its four nearest neighbor sites. Adjust *grow* so that a bacterium dies if its available nutrition falls below a given threshold.

8. Create a variation of Project 1 where initially no bacteria are attached to the surface. Free-floating bacteria can attach to the surface of the biofilm. Run the simulation several times and discuss the variety of initial patterns of colonization.

9. Revise the biofilm simulation so that a dead bacterium decays with time, forming additional nutrients. To do so, we can have degrees of dead, such as *DEAD1*, *DEAD2*, and *DEAD3* for a bacterium that decays in three time steps.

10. Revise the biofilm simulation so that if an empty neighbor does not exist for a dividing bacterium, a random walk of a given maximum number of steps occurs to search for an empty location. In a walk, a north, east, south, or west direction is repeatedly selected at random until success or the maximum number of steps is achieved. Such a random walk in search of an available location is comparable to a daughter bacterium adhering to the mother bacterium and pushing other bacteria out of the way. Discuss the impact of this revision on the biofilm structure.

11. Division requires energy. Thus, revise the simulation so that a dividing cell consumes nutrition from its own and, to a lesser extent, its neighboring cells. Discuss the impact of this revision on the biofilm structure.

12. Develop a simulation where the biofilm has two types of bacteria, Types 1 and 2, which are competing for resources. Have the bacteria grow at different rates. That is, have a Type 1 bacterium divide with a certain probability and a Type 2 bacterium divide with another probability. Also, have Type 1 bacteria consume nutrients at a different rate than Type 2 bacteria. Examine different initial situations, such as having the number of Type 1 bacteria being greater than the number of Type 2 bacteria or vice versa or having diverse initial configurations. Discuss the results concerning competition and the developing structure. The visualization should display the two bacteria with different colors.

13. Model a biofilm composed of two organisms, Type 1 that grows fast in an oxygen-rich environment, which occurs at the surface, and Type 2 that thrives in a low-oxygen setting deeper within the biofilm. Have detachment (see Projects 2 and 3) as a phase of the simulation. Discuss the results (Picioreanu et al. 2004).

14. According to [Stewart, 2003], "a biofilm that is 10 cells thick will exhibit a diffusion time 100 times longer than that of a lone cell." Adjust the diffusion algorithm to model slower diffusion deeper within the biofilm. Compare the results of this model to *biofilm* of this module.

15. Revise Project 1 or 3 to account for a phenomenon observed in some biofilms of the necessity of a critical neighborhood density for growth. To do so, we could adjust the rule for growth so that division cannot occur unless a bacterium has at least one neighbor. Discuss the results. (Picioreanu et al. 1996)

16. Model the formation of filamentous bacteria, which grow in long thread-like strands, by having preferred growth in the direction away from the surface of the biofilm. Filamentous bacteria occur in wastewater treatment activated sludge flocs, which are large aggregates of adherent bacteria. These flocs can be filtered out for drinking water purification and sewage treatment (Picioreanu et al. 2004).

17. The module's model does not account for the bacterial products, such as chemical signals, metabolites, and antibiotic chemicals. In the same phase as consumption, model such product release.

18. Using a computational tool, develop a 3D version of the *biofilm* model.

19. Using a computational tool, develop a 3D version of any of the projects.

20. Using C with MPI, develop a HPC version of any of the projects.

21. Using C with MPI, develop a 3D version of the *biofilm* model.

22. Using C with MPI, develop a 3D version of any of the projects.

23. Develop sequential and parallel versions of any of the above projects or the biofilms model in the module.

a. Time both versions, running them with increasingly larger datasets and with a fixed number of processes on a parallel machine. Graph the **speedup**, or the time for the sequential version over the time for the parallel version, versus dataset size. Discuss the results.

b. For a large dataset, time the sequential version. Then, for the same dataset, repeatedly time the parallel version for an increasing number of processes. Graph the speedup versus number of processes. Discuss the results.

9. Answers to Quick Review Questions

1. $0.36 = (1 - 8 \cdot 0.1)(0.5) + 0.1(0.2 + 0.3 + 0.4 + 0.0 + 0.6 + 0.1 + 0.3 + 0.7)$

2.
a. 5-by-5
b. 0, 0, 0, 0, 0
c. 0.2, 0.2, 0.3, 0.4, 0.4
d. 0.7, 0.1, 0.3, 0.7, 0.1

3.
a. $0.24 = 0.8(0.5/(0.2 + 0.3 + 0.0 + 0.5 + 0.7))$
b. (2, 3) is the only empty neighbor
c. dead because the bacterium has no nutrition

4. First row: 0.0, 0.1, 0.4; Second row: 0.0, 0.3, 0.6; Third row: 0.0, 0.3, 0.7
Subtraction only occurs at locations with bacteria, and a nutrition value cannot fall below 0.0. Calculations are First row: 0.2 - 0.2, 0.3 - 0.2, 0.4; Second row: 0.0, 0.5 - 0.2, 0.6; Third row: maximum of 0.1 - 0.2 and 0.0, 0.3, 0.7

5.
a. 95-by-60
b. 2 through 96
c. (32, 25)
d. (34, 24)

6.
a. 0.12, or 12%
b. 0.14, or 14%, = 0.26 - 0.12
c. 0.43, or 43%, = 0.69 - 0.26
d. 0.31, or 31%, = 1.0 - 0.69

7.
a. (17, 22) in the un-extended grid because site (18, 23) in the extended grid has no empty neighbors
b. 19 rows and 24 columns because $m = 17$ is the number of rows in the un-extended grid and site (18, 23) in the extended grid has a border cell to its east in column 24.
c. 17 rows and 22 columns
d. (17, 22)
e. (16, 22), (1, 22) because of periodic boundary conditions, and (17, 21)

8. *BACTERIUM* → *DEAD*, *EMPTY* → *BACTERIUM*

9. *nutGrid*(*i*, *j*) ← (*nutGrid*(*i*, *j*) - *CONSUMED*)
if *nutGrid*(*i*, *j*) < 0.0
nutGrid(*i*, *j*) = 0.0

10. **a.** *bacteriaGrid* = *initBacteriaGrid*(*m*, *n*, *probInitBacteria*)
b. *nutrientGrid* = *initNutrientGrid*(*m*, *n*)
c. *extNutrientGrid* = *extendNutrientGrid*(*nutrientGrid*)
d. *nutrientGrid* = *applyDiffusionExtended*(*extNutrientGrid*, *diffusionRate*)
e. *bacteriaGrid* = *grow*(*bacteriaGrid*, *nutritionGrid*, *p*)
f. *nutrientGrid* = *consumption*(*bacteriaGrid*, *nutritionGrid*)

9. References

Boles, B. R., M. Thoendel, and Singh, P. K. 2004. "Self-generated diversity produces "insurance effects" in biofilm communities." Proceedings of the National Academy of Science 101 (47): 16630-16635.

Chicurel, M. 2000. "Slimebusters." Nature 408:284-286.

Costerton, B. 2004. "Microbial ecology comes of age and joins the general ecology community." Proceedings of the National Academy of Science 101 (49):16983-16984.

Costerton, J. W., P. S. Stewart and E. P. Greenberg. 1999. "Bacterial biofilms: a common cause of persistent infections." Science 284: 1318-1322.

Donlan, R. M. and J. W. Costerton. 2002. "Biofilms: survival mechanisms of clinically relevant microorganisms." Clin. Microbiol. Rev. 15(2) 167-193.

Fux, C. A., J. W. Costerton, P. S. Stewart and P. Stoodley (2005). "Survival strategies of infectious biofilms." Trends Microbiol. 13(1): 34-40.

Genco, R., S. Offenbacher & Beck, J. 2002. Periodontal disease and cardiovascular disease: epidemiology and possible mechanisms. *J. Am. Dent. Assoc.*: 133: 14S-22S.

Harrison, J. J., R. J. Turner, L. L. R. Marques and H. Ceri. (2005). "Biofilms." American Scientist 93: 508-515.

IWA (International Water Association). 2006. "Mathematical Modeling of Biofilms." IWA Task Group on Biofilm Modeling. IWA Publishing. 208 p. 2006_Book_IWA-STR18_Wanner-et-al.pdf

Laspidou, Chrysi S. and Bruce E. Rittmann. 2004. "Evaluating trends in biofilm density using the UMCCA model." Water Research 38: 3362–3372

_____. 2004. "Modeling the development of biofilm density including active bacteria, inert biomass, and extracellular polymeric substances." Water Research 38: 3349–3361.

Nadell, C. D., J. B Xavier, S. A. Levin and K. R. Foster 2008. "The evolution of quorum sensing in bacterial biofilms." PLoS Biology 6(1):171-179.

Overman, Pamela R. 2000. Biofilms: a new view of plaque. Journal of Contemporary Dental Practice. 1(3): 1-7.

Picioreanu, Cristian, Jan-Ulrich Kreft, and Mark C. M. van Loosdrecht. 2004. "Particle-Based Multidimensional Multispecies Biofilm Model." Applied and Environmental Microbiology 70(5): 3024–3040.

_____, M.C.M. van Loosdrecht and J.J. Heijnen. 1996. “Cellular Automata Models for Biofilm Growth.” Presented at the “Bioprocess Engineering Course”, 14-18 June, Stockholm.

Sauer, K., A. H. Rickard and D. G. Davies. 2007. “Biofilms and biocomplexity.” Microbe 2(7): 347-353.

Schaudinn, Christoph, et al. 2007. "Bacterial Biofilms, Other Structures Seen as Mainstream Concepts" v. 2, No. 5, Microbe, pp. 231-237.

Stewart, Philip S. 2003. "Guest Commentaries, Diffusion in Biofilms" J. Bacteriol. 185(5): 1485–1491.

UCL Eastman Dental Institute, Microbial Diseases. 2008. "In Vitro Models, Biofilms and Ecology: Constant Depth Film Fermentor." http://www.eastman.ucl.ac.uk/research/MD/biofilms_ecology_models/index.html.

Watnick, P. and Kolter, R. 2000. “Biofilm, city of microbes.” J. Bacteriol. 182(10): 2675-2679.