## 10.5 Biofilms: United They Stand, Divided They Colonize

## *MATLAB* Quick Review Questions

*Introduction to Computational Science:*

*Modeling and Simulation for the Sciences, 2nd Edition*

Angela B. Shiflet and George W. Shiflet

Wofford College

© 2014 by Princeton University Press

This file contains system-dependent Quick Review Questions and answers in *MATLAB* for Module 10.5, "Biofilms: United They Stand, Divided They Colonize." Complete all code development in *MATLAB*.

**Quick Review Question 1** Suppose constant *MAXNUTRIENT* is initialized to be 1.0. Write a function, *initNutrientGrid*, with parameters *m* and *n* for the number of rows and columns to return an *m*× *n* grid (i.e., table) of *MAXNUTRIENT* values.

**Quick Review Question 2** Suppose the constant nutrient value to the east is 1.0 and *mat* is the *MATLAB* matrix [[0.1, 0.1, 0.1]; [0.2, 0.2, 0.2]; [0.3, 0.4, 0.5]]. Give the second row of the table that *extendNutrientGrid*(*mat*) returns.

**Quick Review Question 3** This question develops *initBacteriaGrid*, which has parameters for *m*, *n*, and *probInitBacteria*.

**a.** Assign to *bacteriaGrid* an *m* × *n* matrix of *EMPTY* values.

**b.** For the first row of *bacteriaGrid* make a cell's value is *BACTERIUM* with a probability of *probInitBacteria*.

**Quick Review Question 4** The function *extendBacteriaGrid* with parameter *mat* returns an extended bacteria grid.

**a.** What values will appear in the first row of the extended matrix?

**b.** What values will appear in the first column of the extended matrix?

**Quick Review Question 5** Suppose *i* and *j* are a cell's row and column in an extended matrix and *m* = 12 is the number of rows in the corresponding un-extended matrix. For each cell picked for the daughter bacterium in division, give the returned coordinates, (*newi*, *newj*), in the un-extended grid.

**a.** *i* = 5, *j* = 7, cell to west picked

**b.** *i* = 5, *j* = 7, cell to east picked

**c.** *i* = 1, *j* = 7, cell to north picked

**d.** *i* = 12, *j* = 7, cell to south picked

**Quick Review Question 6** Suppose *pickNeighbor* begins as follows:

function [ a,b ] = pickNeighbor(i, j, m, N, E, S, W)

with parameters of a cell's row (*i*) and column (*j*) in an extended matrix, the number of rows (*m*) of the corresponding un-extended matrix, and the values of the (*i*, *j*) cell's four nearest neighbors (*N*, *E*, *S*, *W*) in the bacteria grid. We define *lst* as [N, E, S, W].

**a.** Write the statement to assign to *pos* a list of the indices of all *EMPTY* values in *lst*.

**b.** Define *newi* and *newj* to be the indices in the un-extended grid corresponding to the indices, *i* and *j*, in the extended grid.

**c.** Start the *if* statement that tests if no neighbor (*N*, *E*, *S*, *W*) is empty to return the pair (*newi*, *newj*) through *a* and *b*. The next question parts refer to the *else* clause of the *if* statement.

**d.** Assign a random index from *pos* to *r*.

*For Parts e-j, give* a *and* b *that are returned in each situation.*

**e.** *pos*(*r*) is 1 (i.e., north) and *newi* is greater than 1

**f.** *pos*(*r*) is 1 and *newi* is 1

**g.** *pos*(*r*) is 3 (i.e., south) and *newi* is less than *m*

**h.** *pos*(*r*) is 3 and *newi* is *m*

**i.** *pos*(*r*) is 2 (i.e., east)

**j.** *pos*(*r*) is 4 (i.e., west)

**Quick Review Question 7**

**a.** Write a statement to assign to *n* the number of columns in *nutritionGrid*.

**b.** Following the pseudocode in the text, define the function *grow*.

**Quick Review Question 8** If a location has a bacterium and 0.05 amount of nutrient and *CONSUMED* is 0.1, give the new value of the nutrient in that cell of the nutrition grid after consumption.

**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*).

**Quick Review Question 10** The function *extendNutrientGrid* takes a parameter grid, *mat*, and returns an extended matrix for periodic boundary conditions in the north-south directions and absorbing boundary conditions in the east (constant value 1.0) and west (constant value 0.0) directions.

**a.** Write a statement to start implementing the periodic boundary conditions in the north-south directions by assigning the extended matrix to *extendRows*.

**b.** Write a statement to assign to *m* the number of rows of *mat*.

**c.** Write a statement to assign to *substrate* a column vector of *m* + 2 zeros.

**d.** Write a statement to assign to *constNutrient* a column vector of *m* + 2 *MAXNUTRIENT* values.

**e.** Write a statement to assign to *extendedGrid* the matrix with *substrate* as the first column, the columns of *extendRows*, and *constNutrient* as the last column.

**Quick Review Question 11** Following the pseudocode of this section, write the function *biofilm* in *MATLAB*, assuming the definition begins as follows:

function [bacGrids, ntGrids] = biofilm(m, n, probInitBacteria,…

diffusionRate, p, t )

**Quick Review Question 12** Suppose *g* is a matrix representing a bacteria grid. create the color map to display each empty cell as yellow (full red and full green), a cell with a bacterium as green, and each cell with a dead bacterium as medium gray (level 0.5).

**Quick Review Question 13** Suppose *g* is a matrix representing a nutrient grid. Give the graphics commands to display the grid in 10 shades of gray, where a cell with zero nutrient appears white and one with *MAXNUTRIENT* appears black.

Answers to Quick Review Questions

**QRQ 1**

function [ NutrientGrid ] = initNutrientGrid( m,n )

%INITNUTRIENTGRID Function to return an initialized Nutrient Grid

% initNutrientGrid( m,n ) returns an m-by-n matrix with each

% element having the value MAXNUTRIENT

global MAXNUTRIENT

NutrientGrid = zeros(m, n) + MAXNUTRIENT;

end

**QRQ 2** [0.0, 0.3, 0.4, 0.5, 1.0]

**QRQ 3**

**a.** bacteriaGrid = zeros(m, n) + EMPTY;

**b.** for i = 1:m  
 if rand < probInitBacteria

bacteriaGrid(i, 1) = BACTERIUM;

end

end

**QRQ 4**

**a.** Last row of *mat*

**b.** All *BORDER* values

**QRQ 5**

**a.** (4, 5)

**b.** (4, 7)

**c.** (12, 6)

**d.** (1, 6)

**QRQ 6**

**a.** pos = find(lst == EMPTY);

**b.** newi = i - 1; newj = j - 1;

**c.** if length(pos) == 0, a = newi; b = newj; …

**d.** r = randi(length(pos));

**e.** newi - 1, newj

**f.** m, newj

**g.** newi + 1, newj

**h.** 1, newj

**i.** newi, newj + 1

**j.** newi, newj - 1

**QRQ 7**

**a.** n = size(nutritionGrid, 2);

**b.**

function [ bacGrid ] =grow( bacteriaGrid, nutritionGrid, p )

global BACTERIUM DEAD

bacGrid = bacteriaGrid;

m = size(nutritionGrid, 1);

n = size(nutritionGrid, 2);

prob = probGrow(bacteriaGrid,nutritionGrid,p);

extBacGrid = extendBacteriaGrid(bacteriaGrid);

extNutGrid = extendNutrientGrid(nutritionGrid);

for i= 2:m+1

for j= 2:n+1

if extBacGrid (i,j) == BACTERIUM

if extNutGrid(i,j) <= 0

bacGrid(i-1,j-1)= DEAD;

elseif rand(1) < 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;

end

end

end

end

end

**QRQ 8** 0, because the amount of nutrient in a cell cannot fall below 0.

**QRQ 9** *nutGrid*(*i*, *j*)  (*nutGrid*(*i*, *j*) - *CONSUMED*)

if *nutGrid*(*i*, *j*) < 0.0

*nutGrid*(*i*, *j*) = 0.0

**QRQ 10**

**a.** extendRows = [mat(end, :, :); mat; mat(1, :, :)];

**b.** m = size(mat, 1);

**c.** substrate = zeros(m + 2, 1);

**d.** constNutrient = MAXNUTRIENT \* ones(m + 2, 1);

**e.** extendedGrid = [substrate extendRows constNutrient];

**QRQ 11**

function [bacGrids, nutGrids] = biofilm(m, n, probInitBacteria,…

diffusionRate, p, t )

bacteriaGrid = initBacteriaGrid(m, n, probInitBacteria);

nutrientGrid = initNutrientGrid(m, n);

bacGrids = [bacteriaGrid];

nutGrids = [nutrientGrid];

for i = 1:t

extNutrientGrid = extendNutrientGrid(nutrientGrid);

nutrientGrid = applyDiffusionExtended(extNutrientGrid,…

diffusionRate);

bacteriaGrid = grow(bacteriaGrid, nutrientGrid, p);

nutrientGrid = consumption(bacteriaGrid, nutrientGrid);

bacGrids = [bacGrids ; bacteriaGrid];

nutGrids = [nutGrids ; nutrientGrid];

end

end

**QRQ 12** map = [1 1 0;

0 1 0;

0.5 0.5 0.5];

colormap(map);

**QRQ 13** colormap(gray(10));

image(1 - g/MAXNUTRIENT);