Biofilm Model

File: Biofilm.nb

To accompany
"Biofilms : United They Stand, Divided They Colonize"
By Angela B.Shiflet and George W.Shiflet
Wofford College, Spartanburg, South Carolina

Based on
Picioreanu, Cristian, 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.

- **Initialize an *m*-by-*n* matrix of nutrients with *initNutrient* in all cells**

```
In[1]:= MAXNUTRIENT = 1.0;
        CONSUMED = 0.1;

        Clear[initNutrientGrid];
        initNutrientGrid[m_, n_] := Table[MAXNUTRIENT, {m}, {n}];
```

- **Function to return an initialized *m*-by-*n* matrix with a bacterium in a cell of the second column with probability *probInitBacteria***

```
In[5]:= EMPTY = 0;
        BACTERIUM = 1;
        DEAD = 2;
        BORDER = 3;

        Clear[initBacteriaGrid];
        initBacteriaGrid[m_, n_, probInitBacteria_] := Module[{emptyGrid, onSurface, grid},
          emptyGrid = Table[EMPTY, {n - 1}, {m}];
          onSurface = {Table[If[RandomReal[] < probInitBacteria, BACTERIUM, EMPTY], {m}]};
          grid = Join[onSurface, emptyGrid];
          Transpose[grid]
         ]
```

- **Function to take a bacteria grid and to extend it in each direction so that there are periodic boundary conditions to the north and south, a column of *BORDER* to the west, and a column of *EMPTY* values to the east**

```
In[11]:= Clear[extendBacteriaGrid]
         extendBacteriaGrid[mat_] := Module[{ matNS, trans, listBORDER, listEMPTY, transEW},
           (* glue on wrap of N & S rows *)
           matNS = Join[Take[mat, -1], mat, Take[mat, 1]];

           (* glue first column of zeros and last column of initNutrient's *)
           trans = Transpose[matNS];
           listBORDER = {Table[BORDER, {Length[matNS]}]};
           transEW = Join[listBORDER, trans, listBORDER];
           Transpose[transEW]
          ]
```

- **Function to take a nutrient grid and to extend it in each direction so that there are periodic boundary conditions to the north and south, a column of zeros to the west, and a column of all *MAXNUTRIENT* values to the east**

```
In[13]:= Clear[extendNutrientGrid]
         extendNutrientGrid[mat_] := Module[{ matNS, trans, listZeros, listNutrient, transEW},
            (* glue on wrap of N & S rows *)
            matNS = Join[Take[mat, -1], mat, Take[mat, 1]];

            (* glue first column of zeros and last column of initNutrient's *)
            trans = Transpose[matNS];
            listZeros = {Table[0, {Length[matNS]}]};
            listNutrient = {Table[MAXNUTRIENT, {Length[matNS]}]};
            transEW = Join[listZeros, trans, listNutrient];
            Transpose[transEW]
           ]
```

- **Function to return the new nutrient value in a cell by diffusion**

```
In[15]:= Clear[diffusion]
         diffusion[diffusionRate_, site_, N_, NE_, E_, SE_, S_, SW_, W_, NW_] :=
           (1 - 8 diffusionRate) * site + diffusionRate (N + NE + E + SE + S + SW + W + NW)
```

- **Function to take an extended nutrient grid and to return a new grid with diffused nutrients**

```
In[17]:= Clear[applyDiffusionExtended]
         applyDiffusionExtended[matExt_, diffusionRate_] :=
           Module[{m, n, site, N, NE, E, SE, S, SW, W, NW, i, j},
             m = Length[matExt] - 2;
             n = Length[Transpose[matExt]] - 2;
             (*Gets the location of the cells surrounding the cell*)
             Table[
               site = matExt[[i, j]];
               N = matExt[[i - 1, j]];
               NE = matExt[[i - 1, j + 1]];
               E = matExt[[i, j + 1]];
               SE = matExt[[i + 1, j + 1]];
               S = matExt[[i + 1, j]];
               SW = matExt[[i + 1, j - 1]];
               W = matExt[[i, j - 1]];
               NW = matExt[[i - 1, j - 1]];
               diffusion[diffusionRate, site, N, NE, E, SE, S, SW, W, NW],

               {i, 2, m + 1}, {j, 2, n + 1}]
           ];
```

- **Function to return general probability of growth**

```
In[19]:= Clear[probGrow]
         probGrow[bacteriaGrid_, nutritionGrid_, p_] := Module[{selLst, tot},
            selLst = Pick[nutritionGrid, bacteriaGrid, BACTERIUM];
            tot = Total[Flatten[selLst]];
            If[tot > 0, p / tot, 0]
           ]
```

- **Function to return indices of random empty neighbor, accounting for periodic boundary conditions in the north-south direction We cannot expand to to the far west or far east because the first and last columns of the extended bacteria matrix have all *BORDER* values.**

```
In[21]:= Clear[pickNeighbor]
         pickNeighbor[i_, j_, m_, N_, E_, S_, W_] := Module[{lst, pos, newi, newj, rand},
           lst = {N, E, S, W};
           pos = Flatten[Position[lst, EMPTY]];
           newi = i - 1; (* indices in un-extended matrix *)
           newj = j - 1;
           If[pos == {}, {newi, newj}, (* no choice *)
            rand = RandomInteger[{1, Length[pos]}];
            If[pos[[rand]] == 1, If[newi > 1, {newi - 1, newj}, {m, newj}], (* north *)
             If[pos[[rand]] == 2, {newi, newj + 1}, (* east *)
              If[pos[[rand]] == 3, If[newi < m, {newi + 1, newj}, {1, newj}], (* south *)
               {newi, newj - 1}]]]](* west *)
          ]
```

- **Function to grow biofilm, where each bacterium has a chance to grown in a random empty direction**

```
In[23]:= Clear[grow];
         grow[bacteriaGrid_, nutritionGrid_, p_] :=
          Module[{bacGrid, extBacGrid, extNutGrid, m, n, prob, newi, newj},
           bacGrid = bacteriaGrid;
           m = Length[nutritionGrid];
           n = Length[Transpose[nutritionGrid]];
           prob = probGrow[bacteriaGrid, nutritionGrid, p];
           extBacGrid = extendBacteriaGrid[bacteriaGrid];
           extNutGrid = extendNutrientGrid[nutritionGrid];
           Do[If[extBacGrid[[i, j]] == BACTERIUM,
             If[extNutGrid[[i, j]] ≤ 0, bacGrid[[i - 1, j - 1]] = DEAD,
              If[RandomReal[] < 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]]];
               (* Adjust indices for un-extended grid *)
               bacGrid[[newi, newj]] = BACTERIUM
              ]
             ]
            ],
            {i, 2, m + 1}, {j, 2, n + 1}
           ];
           bacGrid
          ]
```

- **Function for consumption of substrate**

```
In[25]:=  consumption[bacteriaGrid_, nutritionGrid_] := Module[{m, n, nutGrid},
           m = Length[nutritionGrid];
           n = Length[Transpose[nutritionGrid]];
           nutGrid = nutritionGrid;
           Do[
            (*Print[i," ",j];*)
            If[bacteriaGrid[[i, j]] == BACTERIUM , nutGrid[[i, j]] = Max[0.0, nutGrid[[i, j]] - CONSUMED]
            ],
            {i,  m}, {j, n}];
           nutGrid
          ]
```

- **Function for biofilm simulation**

```
In[26]:= Clear[biofilm]
         biofilm[m_, n_, probInitBacteria_, diffusionRate_, p_, t_] :=
          Module[{bacteriaGrid, nutrientGrid, extNutrientGrid, bacGrids, nutGrids},
           bacteriaGrid = initBacteriaGrid[m, n, probInitBacteria];
           nutrientGrid = initNutrientGrid[m, n];
           bacGrids = {bacteriaGrid};
           nutGrids = {nutrientGrid};
           Do[
            extNutrientGrid = extendNutrientGrid[nutrientGrid];
            nutrientGrid = applyDiffusionExtended[extNutrientGrid, diffusionRate];
            bacteriaGrid = grow[bacteriaGrid, nutrientGrid, p];
            nutrientGrid = consumption[bacteriaGrid, nutrientGrid];
            AppendTo[bacGrids, bacteriaGrid];
            AppendTo[nutGrids, nutrientGrid],
            {t}];
           {bacGrids, nutGrids}
          ]
```

- **Function to display a list of bacteria grids starting at 1 through entire list of grids**
  Empty (*EMPTY* = 0) shows yellow; bacterium (*BACTERIUM* = 1) shows green;
  dead bacterium (*DEAD* = 2) shows dark gray.

```
In[51]:= Clear[showBacteriaGraphs]
         showBacteriaGraphs[graphList_] := Module[{t, g, trans},
           rasterList = {};
           Do[
            g = graphList[[t]];   (* grid at (t - 1) time step *)
            AppendTo[rasterList, Graphics[Raster[Reverse[g]] /.
               {EMPTY → {1, 1, 0}, BACTERIUM → {0, 1, 0}, DEAD → {0.5, 0.5, 0.5}}]],
            {t, Length[graphList]}
           ];
           ListAnimate[rasterList]
          ]
```

- **Function to display a list of nutrient grids starting at 1 through entire list of grids in grayscale, where the most nutrient is black and the least is white**

```
In[30]:= Clear[showNutrientGraphs]
         showNutrientGraphs[graphList_] := Module[{t, g},
           rasterList = {};
           Do[
            g = graphList[[t]];   (* grid at (t - 1) time step *)
            AppendTo[rasterList, Graphics[Raster[Reverse[1 - g]]]],
            {t, Length[graphList]}
           ];
           ListAnimate[rasterList]
          ]
```

- **Tests**
  To run a test, change the format of the cell to the Input style and execute.

```
In[53]:= {bacGrids, nutGrids} = biofilm[10, 10, .5, .1, 1, 40];
         showBacteriaGraphs[bacGrids]
         showNutrientGraphs[nutGrids]
```

In[38]:=

```
{bacGrids, nutGrids} = biofilm[50, 50, .5, .1, .6, 100];
showBacteriaGraphs[bacGrids]
showNutrientGraphs[nutGrids]
```

```
(* test with timing *)
SeedRandom[48]
startBiofilmTime = AbsoluteTime[];
{bacGrids, nutGrids} = biofilm[50, 20, .5, .1, 1, 500];
startBacteriaGraphTime = AbsoluteTime[];
showBacteriaGraphs[bacGrids]
startNutrientGraphTime = AbsoluteTime[];
showNutrientGraphs[nutGrids]
stopNutrientGraphTime = AbsoluteTime[];
biofilmTime = startBacteriaGraphTime - startBiofilmTime
bacteriaGraphTime = startNutrientGraphTime - startBacteriaGraphTime
nutrientGraphTime = stopNutrientGraphTime - startNutrientGraphTime
```

```
{bacGrids, nutGrids} = biofilm[50, 50, .5, .1, 1, 100];
showBacteriaGraphs[bacGrids]
showNutrientGraphs[nutGrids]
```

```
{bacGrids, nutGrids} = biofilm[50, 50, .5, .1, 1, 500];
showBacteriaGraphs[bacGrids]
showNutrientGraphs[nutGrids]
```