Mathematical modeling of interactions between colonic crypts

Nils Erlanson
Martin Edin
Abstract

Mathematical modeling of interactions between colonic crypts

Nils Erlanson, Martin Edin

Mathematical modeling of colonic crypts has been done for almost half a century, partly as colorectal cancer is widely suspected to originate in the crypts. This paper will investigate how cells proliferate in colonic crypts, and the interaction with adjacent crypts by mathematical modeling. The model uses a pressure gradient to simulate the proliferation. The pressure gradient is solved by using the finite element method of a geometry which resembles colonic crypts. Stochastic time steps are taken via events calculated by Gillespie’s algorithm and the resulting model simulates cells wandering along the colonic crypts and releasing when there is enough pressure on a given cell. Biologically, the shedding of the cells are noticed to occur at the surface of the crypts. However, in this model cells release more frequently in the bottom due to the pressure distribution. This could imply that the cells have a more active nature in order to be able to release at the top of the crypts.
1 Populärvetenskaplig sammanfattning

För att förstå och återskapa beteenden av biologiska processer och förändringar i människokroppen, har det under de senaste årtiondena lagts mycket fokus på matematisk modellering. Matematisk modellering används idag frequenter inom naturvetenskapliga samt samhällsvetenskapliga sammanhang för att förstå olika system och förutsäga beteenden.

Målet med detta projekt är att återskapa cellers spridning i den mänskliga tarmen. Miljontals små fack, kallade krypter, utgör tarmens geometri, med stamceller som ständigt förökar sig. Tarmcancer tros ha sitt ursprung från mutationer inom dessa krypter. Spridningen av celler samverkar mellan krypterna och celler som när ytan kommer på släppa från tarmväggen när cellerna stigit upp ur krypterna.

Med hjälp av en modell i programmet MATLAB om cellers spridning kommer interaktionen mellan olika krypter simuleras. Med matematiska metoder löses en tryckfördelning hos cellpopulationer och slumpmässiga händelser driver simulationen. Händelserna leder till spridning av de förnyande cellerna i krypterna och om trycket blir nog stort släpper cellerna från tarmväggen.

Den framtagna modellen klarar av att simulera denna passiva process, men på grund av att trycket på cellerna nere i krypterna är större släpper största delen celler i botten. Detta kan antyda på att cellspridningen sker via en aktiv mekanism.
Contents

1 Populärvetenskaplig sammanfattning 2

2 Introduction 4

3 Theory 4
   3.1 Biological theory .................................................. 4
   3.2 Mathematical theory ............................................... 5
      3.2.1 Finite element method ...................................... 5
      3.2.2 Gillespie’s algorithm ....................................... 6

4 Model 6
   4.1 Basic model .......................................................... 6
   4.2 Creating geometry .................................................. 6
   4.3 Initiating the simulation .......................................... 7
   4.4 Degrees of freedom and pressure gradient ..................... 8
   4.5 Probability intensities ........................................... 9
   4.6 Events ............................................................... 10

5 Results 10
   5.1 Function of model ................................................ 10
   5.2 Expanded geometry ............................................... 11

6 Discussion 12
   6.1 General behaviour ................................................ 12
   6.2 Passive or active process ....................................... 13
   6.3 Difficulties ......................................................... 13

7 Conclusion 14

8 Appendix 16
2 Introduction

Mathematical modeling is a powerful tool to investigate hypothesis of different observed biological, physical or chemical behaviours. One example is to simulate how cells proliferate in the intestine. The process which the model will try to simulate is a close up at the cell proliferation of the outer layer of the intestine. This layer is built up by creases called colonic crypts and the simulation will investigate the interaction between them. The process is driven by stem cells residing in the bottom of the crypt. Proliferation pushes the cells along the wall and interacts with cells from other crypts due to overpopulation of cells they will shed into the intestine.

Mathematically, this is modelled as each crypt having a single stem cell in the bottom. These cells are the only ones allowed to proliferate. As the stem cell divides, a single slot gains one cell, to then contain two cells and therefore a higher pressure originates from that place. The Laplace-equation is solved in order to gain a pressure gradient of the geometry, which drives to proliferation. Different probabilities depending on the pressure will force the cells off the boundary. If a cell loses contact with the boundary it will be excluded from the model from that time forward. MATLAB discretizes the geometry and to solve the pressure gradient, the finite element method is used. The events and the duration of the time step are stochastically calculated with Gillespie’s algorithm. Consequently, the model is event driven.

The aim of the project is to be able to simulate the behavior of cell shedding in the intestine, recreating biological behavior. This is done to expand knowledge of how cells proliferate under specific circumstances. In order to grasp the difference in behaviour by changing the reaction constants and geometries, some schematic graphic and animations will be included in the report.

3 Theory

3.1 Biological theory

The human intestine contain an outer layer of colonic crypts that are renewed every 2-3 days due to long-living stem cells that proliferate at the bottom of the crypts.\cite{1} In the large intestine, around $2 \cdot 10^7$ of tube formed crypts form a creased geometry, see Figure 1. This is one of the most rapidly renewing mammalian tissues with a complete turnover of 3-5 days. The overpopulation of cells down the crypts builds pressure and the differentiated cells ascend up the crypt wall and as the cells reach the surface they undergo apoptosis, i.e natural cell death, and are shed into the intestinal lumen. \cite{2}

Due to the cell growth in the base of the crypt, the migration is thought to be a mitotic activity. The process is described as a passive pressure-driven process that force the cells up the wall and ultimately release into the lumen. However, an active cell movement with an active base membrane flow and cell shedding is also a possibility. \cite{1}
3.2 Mathematical theory

In the simulation two main computational method are used. The finite element method is used to solve the pressure gradient of the geometry and Gillespie’s algorithm is used to advance the proliferation.

3.2.1 Finite element method

Finite element method, called FEM, is a numerical method often used for big simulations of mechanical engineering, biomechanics, aeronautical etc. The simulations often solves problems of heat transfer, pressure gradients or electromagnetic potential. One of the biggest strengths of the FEM-method is its potential to work with complex geometries.

The first step of the method involves discretizing the space to find a solution to a partial differential equation. This is often done by triangulation, especially for 2-dimensional problems. It is also possible to use polygons or squares. The discretization results in a mesh consisting of nodes. From this, a basis function is chosen. Commonly piece-wise smooth and linear. To each node there is a base function, \( \phi_i \), assigned. The base functions are continuous throughout the space but are only nonzero in the region surrounding the node. An approximation of the solution, \( U_h \), is solved by computing a weight function \( u_j \) for each node. Such that the approximation can be solved with a linear system of equations, in the form:

\[
U_h = \sum_{j=1}^{n} u_j \phi_j
\]
where \( n \) is the number of nodes.[4]

### 3.2.2 Gillespie’s algorithm

The Gillespie algorithm is an algorithm within probability theory that generates a statistically correct solution of a stochastic equation. A short summary of the steps, somewhat modified to the process used, are presented below:

**Step 1** Initialize cell population and define reaction constants for different events.

**Step 2** Generate random number to determine which event that will occur, as well as the time interval to the next event. The probability of an event to occur is proportional to the number of possible events.

**Step 3** Increase the simulation time with the time step randomly generated in Step 2. Depending on what event that occurred, update cell population.

**Step 4** Unless simulation time is expired, go back to Step 2. [5]

### 4 Model

The model that will simulate the proliferation of cells is driven by Gillespie’s algorithm. Consequently, the simulation is event driven, such that the time steps are depending on the number of possible events and not by a given time interval. Each step solves a pressure gradient with the Finite element method, through discretization with triangulation, and a system of linear equations with matrices derived from the method. Following this short introduction is a closer look at the different parts of the model.

#### 4.1 Basic model

An already existing model which simulates cells proliferating out through a simple geometry is used to build the model. It has both the Gillespie algorithm implemented and the FEM solution for the pressure gradient.

The basic model has a working visualization using triangulation of the points from the mesh. From the triangulation the model uses a method to create voxels. These voxels are in one of three different states. Empty, singly occupied or doubly occupied and the states are represented with different colours.

#### 4.2 Creating geometry

The first step of the model is creating a geometry which resembles the colonic crypts. It is done with MATLAB’s PDE toolbox. The crypts are simplified to triangles with ratios of depth, width and length similar to actual colonic crypts. [6]. The line between the crypts represents the intestine wall. This is also referred to as the top of the crypts. The second step of the finite element method is triangulation which is also done using PDE toolbox, see Figure 2. This results in a mesh of the geometry, where the parameters that represents the triangulation, in form of vertices, points and edges, are exported from the toolbox and
used in the simulation. Each node of the triangulation represents a slot where cells can exist. Figure 2 shows a geometry of two colonic crypts with the triangulation from PDE toolbox.

![Triangulated mesh of colonic crypts](image)

Figure 2: Geometry resembling the colonic crypts created by Matlab’s PDE tool box. The surface of the crypts is represented by $y$-axis=0, and the bottom of the crypt is everything below. The area above the $y$-axis represents the lumen were cells can shed when released from the boundary.

### 4.3 Initiating the simulation

The FEM step of creating a linear system of equations is done by exporting parameters from the mesh of PDE toolbox. From the parameters a stiffness matrix and a mass matrix is created. These matrices are used to solve the pressure gradient of the system.

The initial population is created before the simulation begins, which is the first step of the Gillespie algorithm. The nodes in the bottom of each crypt are the only ones which are allowed to proliferate, i.e. stem cells.

A quotient between the length and edges is also needed to be able to choose events correctly, called gradquotient. The gradquotient is an integration of the edge lengths over the distance from each node in a given direction, shown in figure 3. This is computed using variables from the voxels created in the basic model 4.1.
4.4 Degrees of freedom and pressure gradient

To classify and keep track of the state of each voxel, different "Degree of freedom" are created to represent the event that may occur. All active cells as well as the empty voxels, which are located next to active cells, form the base of the model. The singly occupied voxels represents the boundary movement and how the population may spread. The overly occupied voxels represents how the pressure builds in the geometry, forcing the cells to spread. Finally, the layer above of the boundary represents the cells that release from the geometry due to pressure. Cells that are released from the geometry are considered dead and removed from the geometry in the next visualization frame. The states are enumerated in the discretized geometry to determine the the events. See section 4.6. For a visualization of the states, see Figure 4.
4.5 Probability intensities

The second step of Gillespie’s Algorithm, i.e. chose a random event, is taken via computing the probability intensity of either a boundary movement, movement of a cell from a doubly occupied voxel or a cell division event. The movement intensities are proportional to the number of cells in the different states, but also through the reaction constants previously mentioned. The reaction constants scale the intensities of different events, allowing some events to happen more frequently, they are also used to modify the simulation to a desired behaviour. The proliferation rate is proportional to the number of voxels in the simulation.

The event and the time step for the next event are both calculated stochastically, according to Gillespie’s algorithm. Both calculations are scaled to the number of possible events and the values of the probabilities, such that if many events have a high probability to happen the duration of the time step decreases.
4.6 Events

There are three possible events. Singly occupied voxels, representing the boundary movement, can move to an empty voxel. The doubly occupied voxels are able to move one cell to either an empty voxel or to a nearby singly occupied voxel. The third possible event is cell division, where a singly occupied voxel becomes doubly occupied.

Using the "Degrees of Freedom", presented in 4.4, the movements are done by first finding possible voxels to move from. Given a possible event of a voxel, there can be several different neighbors to migrate to. Due to the pressure gradient and reaction constants, there are different probabilities to where it actually moves, and ultimately this event is determined stochastically.

For visualization purposes, the cell population’s state are saved with predetermined time steps depending on the duration of the simulation. Thus, numerous events occurs between every visualization step.

5 Results

5.1 Function of model

By creating geometries that fulfill the biological requirements and by triangulating with a spacing that is not too refined, the resulting meshes according to figure 2 have good conditions for the objective. Through adjusting the reaction constants, the model’s cells are initially able to "wander" up along the crypt’s walls. After a layer of single cells have filled the voxels of the crypt, doubly occupied voxels starts appearing on the walls of the crypts. Eventually, the surface gets overpopulated as well, forcing the cells to release from the boundary. A ratio of 25% of cells that were shed at the surface of the geometry were achieved.
5.2 Expanded geometry

An expanded geometry with four proliferating crypts was simulated. Using the same simulation time, the ratio of cells that were shed on the surface were 25% and due to larger probability intensities the time steps were shorter and thus more events occurred.
6 Discussion

6.1 General behaviour

The model used in this simulation has a relatively simple approach. Despite this, by scaling the reaction constants correctly, the desired behavior was somewhat achieved. The desired behaviour was to make the cells release at the surface rather than at the bottom and this was achieved when the movement between voxels on the boundary were more likely to occur rather than releasing from the boundary. If the constants are inadequate, all cells release in the bottom due to the pressure always being greater than at the surface of the geometry. Despite this, with appropriate reaction constants the model achieved 25% ratio of cells that were shed at the surface. To further improve this, the reaction constant that gives a cell shedding event could be altered, making it more likely to release at the surface rather than releasing in the crypt. This method would however imply that the cells "know" their position and therefore suggest an active process.

Comparing the different geometries, two and four crypts, some logical behavior is observed. Both geometries build larger pressure at the center, resulting in more overly occupied voxels with more released cells in the center region. Due to the edges of the geometries being open and not affected by pressure from their neighbouring crypt, the important simulation behavior is limited to the center. Another noticeable behavior is the case of more than two crypts, the whole geometry acts as a pressure gradient with a higher pressure at the center of the population, as seen in figure 6. Red voxels represents an active pressure and a higher density of red voxels indicates that there is an overall greater pressure in that area. As the density of red voxels decreases towards the edges the overall pressure diminishes as well. All this could be translated as a macro pressure gradient, with its greatest pressure at the center.

Because of the model being event driven another unwanted behaviour appears more clearly as the geometry increases in size. Several events cannot happen at the same time. This interferes with a constant proliferation of stem cells or cells moving far from each other.
6.2 Passive or active process

The cell density in the crypt exceeds the density of the surface of the geometry. This leads to a higher pressure in the crypts, resulting in more cells that release in the bottom. According to the biological theory, the release will mainly occur after the cells ascended to the surface. If the process is passive, the rates of the proliferation is presumably different than our model, forcing the cells upward the boundary rather than releasing at the bottom. Possible solutions to improve the model would be to track the age of the cells and making them more likely to release after a certain age. Another possible improvement is to alter the reaction constants such that they depend on a acidity gradient. The intestine would vary in acidity depending on the depth, making the cells more likely to release at the surface.

Due to the fact that there always will be a higher pressure in the bottom of the crypt, the likelihood for cells to release at the bottom will always be greater than for them to shed at the top of the geometry. Because of this inherent pressure difference it would be hard to get the correct behaviour and it is reasonable that it could be an active process.

6.3 Difficulties

One problem throughout this project was the visualization aspect. The visualization from the basic model only works on a convex geometry as the method creates open voxels with extended lines to infinity at the edge of the geometry. On a non convex geometry, these lines intersect and creates misshaped voxels, see Figure 7. As the geometry was non convex this problem was encountered. This was solved by scaling down the visualization to a more simple approach. The voxels are still a part of the code via calculating gradquotients, however in the visualization only the nodes are highlighted to represent the state each voxel is in.

Since the gradquotient is calculated with values from the voxels, some edge lengths were incorrect. The method creates abnormal voxels with incorrect edge lengths, which leads to some inaccurate probability intensities. Also as mentioned before, some lines are of infinity length and when there is no pressure difference, it leads to Not-a-Number values in the probability intensities.
7 Conclusion

A few behaviours are apparent in the model, mainly that the pressure is at its greatest in the bottom of the crypts. Ergo more cells will always shed in the crypts instead of the surface. This is an inherent problem of the model. It could be improved by implementing reaction constants that changes with the age of the cells, a pH gradient over the crypts or improving the geometry to more accurately resemble the crypts. However, regardless of the improvements the inherent nature of the pressure gradient implies that the biological process might be an active mechanism. It is also apparent that bigger geometries with more crypts makes the model crackle, in form of a macro pressure gradient and events not being able to happen independently of each other. Therefore the interaction of the colonic crypts simulated with this model is most accurately depicted with only two crypts.
References


8 Appendix

% Proliferation of two stem cells in separate crypts.
% Two crypts

% Martin Edin, Nils Erlanson

% Repeatable results
rng(222);

% Creating mesh, rectangular geometry with two crypts.
% Loads PDmesh with output variables p, e and t.
load('Geomesh5.mat');

% Triangulate, used to calculate a gradquotient.
DT = delaunayTriangulation(p,e(1:2,:));
[V,R] = voronoiDiagram(DT);

% Switch format somewhat
patchmax = max(cellfun('prodofsize',R_);
R = nan(size(R_1),patchmax);
for i = 1:size(R_1), R(i,1:size(R_i,2)) = R_i;
end
% (allows for visualization by the patch-function)

% Each node in the mesh is a (midpoint of a) voxel
Nvoxels = size(p,2);

% Assemble minus the Laplacian on this grid (ignoring BCs) as well as
% the Mass-matrix
[L,M] = assema(p,t,1,1,0);

% The (lumped) mass matrix gives the element volume
dM = full(sum(M,2));

% Explicitly invert the lumped mass matrix and filter the diffusion matrix
[i,j,s] = find(L);
s = s./dM(i);
keep = find(i == j);
i = i(keep); j = j(keep); s = s(keep);

% Rebuild L, ensuring that the diagonal equals minus the sum of the
% off-diagonal elements
L = fsparse(i,j,s,[Nvoxels Nvoxels]);
L = L+fsparse(1:Nvoxels,1:Nvoxels,-full(sum(L,2))');

% Static neighbor matrix
N = fsparse(i,j,1,size(L));
% Neighbor representation for each node
neigh = full(sum(N,2));

% Distance between two voxel midpoints: D1(k) = norm(P(:,i(k)),P(:,j(k)))
D1 = sqrt(sum((p(:,i)-p(:,j)).^2))';

% Edge length: L1 = edge length of edge between voxel i and j
L1 = zeros(size(D1));
for k = 1:numel(i)
    n = fsetop('intersect',R(i(k),:),R(j(k),:));
    n(isnan(n)) = [];
    L1(k) = norm(diff(V(n,:)));    end

% Gradquotient
gradquotient = fsparse(i,j,L1./D1, size(L));

%Locating the two stem cells in bottom of crypts
 [~,i1] = min(((p(1,:)+0.75).^2+(p(2,:)+1).^2);
 [~,i2] = min(((p(1,:)-0.75).^2+(p(2,:)+1).^2);

% Specified cells that proliferate
pdof = [i1 i2]';

% locating boundary
bound = boundary(p',1);
keep = find(p(2,bound) < 0.02 );
bound = bound(keep);

% Locating the layer "just above" the intestine boundary, called death DOF (ddof)
% Specified layer of DOFs where U=0. Cells that move to this DOF releases...
% .. from the intestine wall and is considered dead.
U = fsparse(bound,1,1,[Nvoxels 1]);

ddof = find(N*(U ~= 0) > 0 & U == 0);

% Creating a boundary matrix to acquire different reaction constants for
% boundary movement
VU = (U > 0);

% Creating initial population consisting of two stem cells in the bottom of
% the crypts
U = fsparse(pdof,1,1,[Nvoxels 1]);

% Diffusive pressure rates
Drate1 = 0.0001;  % from boundary to non boundary
Drate2 = 1000;   % boundary --> boundary
Drate3 = 0.01;   % into already occupied voxel
Drate_ = [Drate1 Drate2; NaN Drate3];

% Proliferation rate
lam=2/Nvoxels;
% simulation interval
Tend = 30000;
% solution recorded at this grid:
tspan = linspace(0,Tend,201);
% Sparse solution vector D to record cells that release from the boundary to
% ddof
D= zeros(Nvoxels,1);
D= fsparse(D);
% representation of solution: cell−vector of sparse matrices
% Usave will represent active cells, Dsave will represent dead cells.
Usave = cell(1,numel(tspan));
Dsave = cell(1,numel(tspan));
if ~exist ('report_progress', 'var')
    report_progress = true;
end
if report_progress
    report(tspan,U,'init');
else
    report(tspan,U,'none');
end

if tt <= tspan(end)
    Dsave{1} = D;
    Usave{1} = U;
    i = 1;
    % logic for reuse of LU−factorizations
    updLU = true;
    La = struct('X',0,'L',0,'U',0,'p',0,'q',0,'R',0);

    % counter to check events within while loop and number of dead cells
    counter=0;
    p_counter=0;
    two_counter=0;
    one_counter=0;
    death_counter_UP=0;
    death_counter_DOWN=0;

    % classify the Degrees−Of−Freedoms (DOFs for short)
    % active DOFs: occupied voxels
    adof = find(U);
    % boundary DOFs _which may move_: containing one cell per voxel and
    % with an empty voxel besides it
    bdofm = find(N*(U > 0) < neigh & U == 1);
    % source DOFs containing more than one cell per voxel (actually 2)
    sdof = find(U > 1);
    % source DOFs _which may move_: with a voxel containing less number of
    % cells next to it (actually 1 or 0)
sdofm = find(N*(U > 1) < neigh & U > 1);

% injection DOFs: the layer of voxels "just outside" adof where we may
% inject a BC (pressure 0 of the free matrix)
1dof = (N*(U ~= 0) > 0 & U == 0);
1dof = find(1dof);

% "All DOFs" = adof + 1dof, like the "hull of adof"
Adof = [adof; 1dof];

% The above will be enumerated within U, a Nvoxels-by-Nvoxels sparse
% matrix. Determine also a local enumeration, eg. [1 2 3
% ... numel(Adof)].
Adof_ = (1:numel(Adof))';
 [~,ix] = fsetop('ismember',bdof_m',Adof');
bdof_m_ = Adof_(ix);
 [~,ix] = fsetop('ismember',sdof',Adof');
sdof_ = Adof_(ix);
 [~,ix] = fsetop('ismember',sdof_m',Adof');
sdof_m_ = Adof_(ix);
 [~,ix] = fsetop('ismember',idof',Adof');
idof_ = Adof_(ix);
 [~,ix] = fsetop('ismember',pdof',Adof');
pdof_ = Adof_(ix);

if updLU

% pressure Laplacian – factorization is reused
La.X = L(Adof,Adof);

% selecting all injection DOFs
La.1 = fsparse(idof_,idof_,1,size(La.X));
% remove eqs (rows) for injection DOFs, replace with direct injection

% factorize
updLU = false; % assume we can reuse
end

% RHS source term proportional to the over-occupancy (the rest of the
% zeros ensure the homogeneous Dirichlet BC is satisfied at idof)
Pr = full(fsparse(sdof_1,1./dM(sdof),[size(La.X,1) 1])); % RHS first...
Pr(La.q) = La.U\(La.L\(La.R(:,La.p)\Pr)); % ..then the solution

% compute intensities of possible events

% moving boundary DOFs: each empty neighbour voxel is associated with
% a flow rate proportional to the pressure gradient in that
% direction integrated over the corresponding edge
[ii,jj,gq] = find(gradient(bdof_m,Adof)); % neighbours...
keep = find(U(Adof(jj_)) == 0); % ...to move to
ii = reshape(ii(keep),[],1);
jj_ = reshape(jj_(keep),[],1);
gq = reshape(gq(keep),[],1);
grad = fsparse(ii,1,gq.*max((Pr(bdof_m_(ii))−Pr(jj_)),0)...
   .∗Drate_(2∗VU(Adof(jj_))+1),numel(bdof_m));
moveb = full(grad);

% checking for cases where moveb gives inf*0 and removing that event
if any(isnan(moveb))
    check = isnan(moveb);
    moveb(check) = 0;
end

% Cells in a doubly occupied voxel may move by the same physics
[ii_, jj_, gq] = find(gradquotient(sdof_m,Adof_)); % neighbours...
keep = find(U(Adof(jj_)) < 2); % ...to move to
ii = reshape(ii(keep),[],1);
jj_ = reshape(jj_(keep),[],1);
gq = reshape(gq(keep),[],1);
% remove any possibly remaining negative rates
grad = fsparse(ii,1,gq.*max(Pr(sdof_m_(ii))−Pr(jj_),0).*...  
   Drate_(2∗VU(Adof(jj_)+U(Adof(jj_))+1),numel(sdof_m));
moves = full(grad);

% checking for cases where moves gives inf*0 and removing that event
if any(isnan(moves))
    check = isnan(moves);
    moves(check) = 0;
end

% Proliferation intesity
if any(U(pdof)>=2)
    prolif=0;
else
    prolif = lam∗U(pdof);
end

%Intensities of possible events: Boundary, source and proliferation
intens = [moveb; moves; prolif];

% waiting time until next event and the event itself
lambda = sum(intens);
dt = −reallog(rand)/lambda;
rnd = rand∗lambda;
cum = intens(1);
ix_ = 1;
while rnd > cum
    ix_ = ix_+1;
    cum = cum+intens(ix_);
end
% (now ix_ points to the intensity which fired first)
% Record values for visualization, Usave for active cells and Dsave for dead cells.
if tspan(i+1) < tt+dt
    iend = i+find(tspan(i+1:end) < tt+dt,1,'last');
    Usave(i+1:iend) = {U};
    Dsave(i+1:iend) = {D};
    i = iend;
    if iend == numel(Usave), break; end
end

if ix_ <= numel(moveb)
    % movement of a boundary (singly occupied) voxel
    ix_ = bdof_m_(ix_);
    ix = Adof(ix_);

    % select neighbor to move to
    jx_ = find(N(ix_,Adof(jx_)) == 0); % only allow moves to less populated voxels
    % the pressure gradient drives the movement
    rates = full(gradquotient(ix_,Adof(jx_)))*max(Pr(ix_)-Pr(jx_),0).* ... 
       Drate_(2*VU(Adof(jx_))+1);
    m = find(cumsum(rates) > rand*sum(rates),1,'first');
    n = Adof(jx_(m));

    %execute event: move from ix to n
    %checking if a cell moves to ddof, then it releases from boundary
    if any(ismember(n,ddof))
        U(ix) = U(ix)-1; %From active..
        U(n) = 0; %..to dead
        D(n) = D(n)+1; %Cell moved to ddof and recorded
        %Record where the cells are released
        if p(2,n)<-0.02
            death_counter_DOWN = death_counter_DOWN+1;
        else
            death_counter_UP = death_counter_UP+1;
        end
    end %move from ix to n
    U(ix) = U(ix)-1;
    U(n) = U(n)+1;
end;
updLU = true;
one_counter = one_counter+1;
elseif ix_ <= numel(moveb)+numel(moves)
    % movement of a cell in a doubly occupied voxel
    ix_ = ix_-numel(moveb);
    ix_ = sdof_m_(ix_);
    ix = Adof(ix_);
    % select neighbor to move to
jx_ = find(N(ix, Adof));
jx_ = jx_(U(Adof(jx_)) < 2); % only allow moves to less populated voxels
% the pressure gradient drives the movement
r rates = full(gradquotient(ix, Adof(jx_)))' * max(Pr(ix_)-Pr(jx_), 0) .* ...
  Drate_(2*VU(Adof(jx_))+U(Adof(jx_))+1);
% (thinning of negative gradients)
m = find(cumsum(rates) > rand*sum(rates), 1, 'first');
n = Adof(jx_(m));

% execute event: move from ix to n
if U(n) == 0
  updLU = true;
end

% checking if a cell in a doubly occupied voxel moves to ddof
% then it releases from boundary
if any(ismember(n, ddof))
  U(ix) = U(ix)-1; % From active...
  U(n) = 0; %.to dead
  D(n) = D(n)+1; % Cell moved to ddof and recorded
  % Record where the cells are released
  if p(2,n) < -0.02
    death_counter_DOWN = death_counter_DOWN+1;
  else
    death_counter_UP = death_counter_UP+1;
  end
else
  U(ix) = U(ix)-1;
  U(n) = U(n)+1;
end;
two_counter = two_counter+1;
else
  % otherwise the stem cells proliferate
  ix_ = ix_-numel(moveb)-numel(moves);
  ix_ = pdof_(ix_);
  ix = Adof(ix_);
  U(ix) = U(ix)+1;
  p_counter = p_counter+1;
end
tt = tt+dt;
report(tt, U, '');
counter = counter+1;
end
report(tt, U, 'done');

% return:
% Simulation
M = struct('cdata',{}, 'colormap',{});
figure(1);
set(gcf, 'Renderer', 'opengl')
for i = 1:numel(Usave)
  plot(p(1,ddof),p(2,ddof), 'o', 'Color', [0 0 0]);
hold on

axis([-1.5 1.5 -1 1]); axis equal; axis off

ii = find(Usave{i} == 1);
plot(p(1,ii),p(2,ii),'','MarkerSize',30,'Color',[0 1 0]);

jj = find(Usave{i} == 2);
plot(p(1,jj),p(2,jj),'','MarkerSize',30,'Color',[1 0 0]);
plot(p(1,pdof),p(2,pdof),'','MarkerSize',30,'Color',[0 0 1]);

if i==1
    kk=find(Dsave{i} ~=0);
    plot(p(1,kk),p(2,kk),'','MarkerSize',30,'Color',[0 0 0]);
else
    kk=find(Dsave{i} ~=Dsave{i-1});
    plot(p(1,kk),p(2,kk),'','MarkerSize',30,'Color',[0 0 0]);
end

title(sprintf('t = %f',tspan(i)));
M(i) = getframe(gcf);
hold off
end

, 'delaytime',0,'loopcount',0);