CompuCell3D Quick Start Guide

Version 3.7.0

Maciej Swat, Abbas Shirinifard, Ariel Balter, Nikodem Poplawski, James A. Glazier

Biocomplexity Institute and Department of Physics, Indiana University, 727 East 3rd Street, Bloomington IN, 47405-7105, USA

Please send corrections and Revisions to: mswat@indiana.edu
Last Updated: Sunday, July 14, 2013

Table of Contents

1 Cell Sorting

One relatively simple CompuCell3D simulation models biological cell sorting. In this simulation you start with a mixture of different cell types with different adhesivities to each other.

Embryonic cells of two different types, when dissociated, randomly mixed, and reaggregated can spontaneously sort to reestablish coherent homogenous tissues. Both complete and partial cell sorting (in which large clusters of one cell type are trapped inside a continuous structure of another type) have been observed experimentally in vitro in embryonic cells. Sorting is a key step in regeneration of a normal animal from aggregates of dissociated cells of adult hydra and involves neither cell division nor differentiation but only spatial rearrangement of cell positions. Physically cell sorting is caused by differences in adhesivities. In this simulation you will be able to verify how different hierarchies of adhesion coefficients will lead to different types of sorting.

In this set of exercises we will explore the role of the area (or volume constraint) and the effects of differential surface adhesivity. As you recall from lectures, these are described by the following equations:

$E_{volume} = \lambda_{volume}(V_{cell} - V_{target})^2$ - volume constraint

$E_{adhesion} = \sum_{i,j,neighbors} J(\tau_{\sigma(i)}, \tau_{\sigma(j)})(1-\delta_{\sigma(i),\sigma(j)})$ -contact energy

We may add surface constraint but it is not required for cell sorting to work and for the sake of simplicity will be omitted.
You will be given a template xml file (*cellsort_2D.xml*) with simulation description. For most of the exercises your task will be to slightly modify this file, run the simulation and describe the results. To add another twist to the exercises we do not include step-by-step instruction what needs to be modified. We leave it to you as an exercise itself. However, if you feel you are stuck **ask for help.**

Important: One thing to remember here is that this is a 2D simulation. In CompuCell3D volume means number of pixels occupied by a cell. Surface means number of pixel sides that have contact with other cells (we exclude lattice boundaries from surface calculations).
It is important to realize that if your cells are only one pixel thick, then the surface of such cells (and here I mean true surface) is numerically equal to perimeter. **This means that area of the cell on plane z=0 and z=1 does not count towards the surface**. In this way we simulate 2D behavior in CompuCell3D.Therefore if you look at a "flat" 2D cell from the top and calculate its surface area you will conclude that it is equal to what CompuCell3D volume. In addition to this is you calculate the perimeter of such "flat" 2D cell you will get a quantity which CompuCell3D calls surface.

In 3D there are no surprises and volume and surface of cells have their regular meaning in reality and in CompuCell3D

Throughout the exercises we use the following convention:

L – denotes"light" cells (NonCondensing in CompuCell3D terminology)
D – denotes "dark" (Condensing in CompuCell3D terminology)
M – denotes "medium"(Medium in CompuCell3D terminology)

Example:J_{LD} denotes adhesion coefficient between light (NonCondensing) and dark (Condensing) cells.

N – denotes NonCondensing cells
C – denotes Condensing cells
M – denotes Medium

Example:J_{NC} denotes adhesion coefficient between NonCondensing and Condensing cells.

Before you begin, please copy ***cellsort_2D.xml*** to your private directory. Issue the following command:

```
cp /Users/mswat/CompuCellFull5_install/cellsort_2D.xml <your private
directory>
```

Remember also to create a separate directory for every simulation that you run and of course copy cellsort_2D.xml to this new directory.

To create directory use the following command:

`mkdir <directory>` . Example:

```
mkdir Exercise_1_a
```

After you copy cellsort_2D.xml to this directory you may rename it to more a meaningful name:

```
mv cellsort_2D.xml cellsort_exercise_1_a.xml
```

Above we renamed file called `cellsort_2D.xml` to `cellsort_exercise_1_a.xml`.
In Unix renaming a file is done using `mv` command

Hint: *In all of the exercises below it can happen that cells may stick to the wall. The simplest way to avoid this type of is to set boundary conditions in x and y direction (since we are in 2-D).*

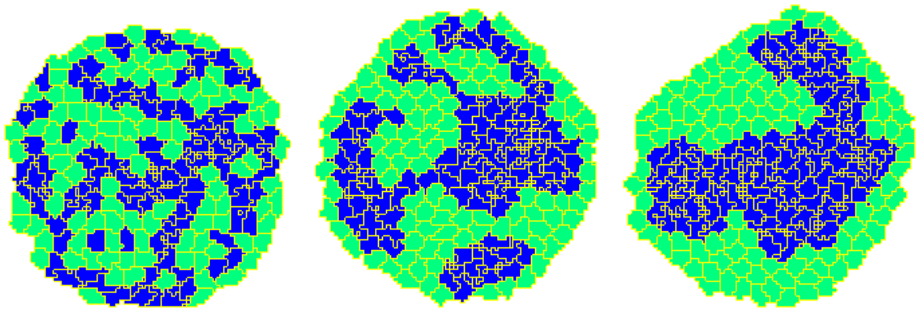


Figure 5. Cell sorting simulation. Snapshots were taken at T=30MCs, T=1500MCS and T=9990 MCS.

First let's look at the entire CC3DML description file for cell sorting before going into more detail:

```
<CompuCell3D>
 <Potts>
   <Dimensions x="100" y="100" z="1"/>
   <Anneal>10</Anneal>
   <Steps>10000</Steps>
   <Temperature>10</Temperature>
   <Flip2DimRatio>1</Flip2DimRatio>
   <NeighborOrder>2</ NeighborOrder >
 </Potts>

 <Plugin Name="Volume">
   <TargetVolume>25</TargetVolume>
   <LambdaVolume>2.0</LambdaVolume>
 </Plugin>

 <!--<Plugin Name="Surface">
   <TargetSurface>25</TargetSurface>
   <LambdaSurface>2.0</LambdaSurface>
 </Plugin> -->

 <Plugin Name="CellType">
    <CellType TypeName="Medium" TypeId="0"/>
    <CellType TypeName="Condensing" TypeId="1"/>
    <CellType TypeName="NonCondensing" TypeId="2"/>
 </Plugin>

 <Plugin Name="Contact">
   <Energy Type1="Medium" Type2="Medium">0</Energy>
   <Energy Type1="NonCondensing" Type2="NonCondensing">16</Energy>
   <Energy Type1="Condensing"    Type2="Condensing">2</Energy>
   <Energy Type1="NonCondensing" Type2="Condensing">11</Energy>
   <Energy Type1="NonCondensing" Type2="Medium">16</Energy>
   <Energy Type1="Condensing"    Type2="Medium">16</Energy>
   <NeighborOrder>2</NeighborOrder>
 </Plugin>

 <Plugin Name="CenterOfMass"/>
```

```
<Steppable Type="BlobInitializer">
  <Gap>0</Gap>
  <Width>5</Width>
  <CellSortInit>yes</CellSortInit>
  <Radius>40</Radius>
  <!--Engulfment BottomType="Condensing" TopType="NonCondensing"
  EngulfmentCoordinate="y" EngulfmentCutoff="50"/-->
</Steppable>
```

1.1 Volume and Surface Constraints

```
<Plugin Name="Volume">
   <TargetVolume>25</TargetVolume>
   <LambdaVolume>2.0</LambdaVolume>
</Plugin>

<Plugin Name="Surface">
   <TargetSurface>20</TargetSurface>
   <LambdaSurface>1.5</LambdaSurface>
</Plugin>
```

These two plugins inform CompuCell3D that the Hamiltonian will have two additional terms associated with volume and surface conservation. That is when spin flip is attempted one cell will increase its volume and another cell will decrease (unless one of the cells is `Medium`). Notice that `Medium` is a cell type with unconstrained volume. Volume constraint essentially ensures that cells maintain the volume which close (this depends on thermal fluctuations) to target volume. The role of surface plugin is analogous to the role played by volume plugin - to "preserve" surface. Note that surface plugin is commented out in the example above.

Energy terms for volume and surface constraints have the form:

$$\mathbf{E}_{volume} = \lambda_{volume} (\mathbf{V}_{cell} - \mathbf{V}_{target})^2$$

$$\mathbf{E}_{surface} = \lambda_{surface} (\mathbf{S}_{cell} - \mathbf{S}_{target})^2$$

Note: performing single pixel copy may cause surface change in more that two cells – this is especially true in 3D.

1.2 Calculating Surface and Volume Constraints

Now that we have introduced volume and surface constraint we recommend that you go over few exercises that will teach you how to properly choose volume and surface constraint parameters so that cells do not freeze and do not disappear. One can of course use trial and error method but it is instructive to learn how these parameters cam be estimated rather than guessed. If you are

new to CompuCell3D you may skip this section at this time, however we strongly encourage you do come back to these exercises , as they show how to systematically pick parameters of most important energy terms.
This set of exercises was written by Dr. Nikodem Poplawski.

1.2.1 Exercise v1.

a) Consider a 2D cell which consists of 1 (one) pixel. The energy of such a system is $H_1 = 8J_{cm} + \lambda_V (V_T - 1)^2$, where J_{cm} is the adhesion coefficient between the cell and the medium. Here, 8 is the number of the second nearest neighbors in 2D (of course, you can reduce or extend the neighbors).
First, set $T = 0$. Your cell should disappear, if the energy of the system containing only the medium, $H_0 = \lambda_V V_T^2$, is lower, $H_0 < H_1$. This occurs if $\lambda_V < \lambda_{V0}$, where $\lambda_{V0} = \frac{8J_{cm}}{2V_T - 1}$ (check it, you can set $J_{cm} = 10$ and $V_T = 25$, for which $\lambda_{V0} = 1.63$). Otherwise, the cell should survive. For values of λ_V much higher than λ_{V0}, your cell will grow until its volume reaches V_T .

b) Try $T > 0$. In this case, the processes increasing the energy of the system have a nonzero probability, but λ_{V0} should still be the approximate threshold for stability of a 1-pixel cell. Check it, repeating point a).

1.2.2 Exercise v2.

Now, consider a cell which is a square $N \times N$. The energy of this cell is $H_N = (12N - 4)J_{cm} + \lambda_V (N^2 - V_T)^2$. Assuming that the cells can take the form of squares only (otherwise the calculations would be too lengthy), we can ask the question what the equilibrium value N_0 is. This value is given by $\frac{\partial H_N}{\partial N} = 0$, and since $\frac{\partial H_N}{\partial N} = 12J_{cm} + 4\lambda_V N(N^2 - V_T)$, we arrive at $N_0^3 - V_T N_0 + \alpha = 0$, where $\alpha = \frac{3J_{cm}}{\lambda_V} \geq 0$. The above equation is cubic in N_0, and has three real solutions (one stable corresponding to equilibrium) if $\alpha < \alpha_{\max}$, where $\alpha_{\max} = 2\left(\frac{V_T}{3}\right)^{3/2}$. Otherwise, there is only one real root which is negative, thus there is no equilibrium value of N . Therefore, if $\lambda_V < \lambda'_{V0}$, where

$\lambda'_{V0} = \dfrac{3J_{cm}}{2\left(\dfrac{V_T}{3}\right)^{3/2}}$, our cell will shrink and disappear (note that $\lambda'_{v0} < \lambda_{v0}$ so we are in the region where a single pixel is unstable).

For $N = 5$, $J_{cm} = 10$, and $V_T = 25$, start from λ_V much larger than λ'_{v0} (here equal to 0.62) and set $T = 5$. Your cell should approach the target volume V_T.
Decrease λ_V. The cell volume should tend to a value smaller than V_T.
Try λ_V just a little larger than λ'_{v0}. The cell volume should approach the equilibrium value $\dfrac{V_T}{3}$.
Try λ_V smaller than λ'_{v0}. Your cell should shrink and disappear. There is no equilibrium volumes smaller than $\dfrac{V_T}{3}$.
You may want to play with different values of V_T.

1.2.3 Exercise v3.

In the Cellular Potts Model, there are two critical temperatures. Around the temperature of dissociation, $T_{c1} = 8\delta E$, where $\delta E = J_{cm} - \dfrac{J_{cc}}{2}$, a single pixel has a long life-time, and a cell may dissociate (single spins disconnect from the rest of the cell). When the temperature is of the order of $T_{c2} = N^2\delta E$, for which the energy and entropy of the system are comparable (the temperature of spinoidal decomposition), the identities of the cell pixels and the medium pixels merge. The cell should "evaporate".
Begin with a $N \times N$ square cell ($N = 5$) with $V_T = N^2 = 25$, $J_{cm} = 10$, $J_{cc} = 2$, and λ_V much (not too much) larger than λ'_{v0} (here equal to 0.62). Start from $T = 0$ (nothing unusual should happen). Then, increase T until it reaches $T_{c1} = 72$. Check if the cell dissociates. Increase T further until it reaches $T_{c2} = 225$. See what happens. There should be a competition between T and λ_V in trying to save the cell from disappearance.

Let us come back to the description of the cell sort simulation.

1.3 Contact energy

Now let's take a look at the most important plugin in the cell sort simulation –contact energy plugin. It is a somewhat more complicated that corresponding plugin for foam simulation but the idea is the same:

```
<Plugin Name="Contact">
   <Energy Type1="Medium" Type2="Medium">0</Energy>
   <Energy Type1="NonCondensing" Type2="NonCondensing">16</Energy>
   <Energy Type1="Condensing"    Type2="Condensing">2</Energy>
   <Energy Type1="NonCondensing" Type2="Condensing">11</Energy>
   <Energy Type1="NonCondensing" Type2="Medium">16</Energy>
   <Energy Type1="Condensing"    Type2="Medium">16</Energy>
   <NeighborOrder>2</NeighborOrder>
</Plugin>
```

1.4 Blob Initializer

The last object that shows up in the configuration file is a steppable called BlobInitializer. Remember, we mentioned that steppables are called every MCS. That is true, except when sole task of a steppable is to initialize cell field. In this case steppable is called once at the beginning of the simulation. What this initializer does, it creates a blob of cells. Each cell is a cube 5x5x1 (`<Width>5</Width>`) and they are tightly packed (`<Gap>0</Gap>`). The additional line `<CellSortInit>yes</CellSortInit>` is used exclusively for cellsort simulation and tells the initializer that cells types will be only 0,1,2 or `Medium`, `Condensing`, `NonCondensing` if you prefer. You can also specify radius of the blob although this is not a requirement. If you do not specify the radius it will be equal to *2/5*x_lattice_dimension*. If you want to increase or decrease radius from its default value, use `<Radius>40</Radius>` option. Any space in the lattice unfilled with cells becomes `Medium` *i.e. e*ffectively all the cells are immersed in `Medium` (unless the radius is so big that cells fill entire lattice).
This initializer is one of CompuCell3D stock initializers, so that you do not need to prepare your own PIF initialization file.

```
<Steppable Type="BlobInitializer">
   <Gap>0</Gap>
   <Width>5</Width>
   <CellSortInit>yes</CellSortInit>
   <Radius>40</Radius>
   <--<Engulfment BottomType="Condensing" TopType="NonCondensing"
   EngulfmentCoordinate="y" EngulfmentCutoff="50"/> -->
 </Steppable>
```

In your initial simulation you will omit Engulfment entry for BlobInitializer. That's why it is commented out now.

The presented syntax of the BlobInitializer is here for compatibility reasons with older versions of CompuCell3D. The recommended syntax of this steppable is shown in the example below:

```
<Steppable Type="BlobInitializer">
   <Region>
     <Gap>0</Gap>
     <Width>5</Width>
     <Radius>40</Radius>
     <Center x="100" y="100" z="0"/>
     <Types>Condensing,NonCondensing</Types>
   </Region>
</Steppable Type="BlobInitializer">
```

We can define many blob-like regions in the simulation by using multiple Region definitions. For example, the following definition of BlobInitializer:

```
<Steppable Type="BlobInitializer">
     <Region>
       <Radius>30</Radius>
       <Center x="40" y="40" z="0"/>
       <Gap>0</Gap>
       <Width>5</Width>
       <Types>Condensing,NonCondensing</Types>
    </Region>

     <Region>
       <Radius>20</Radius>
       <Center x="80" y="80" z="0"/>
       <Gap>0</Gap>
       <Width>3</Width>
       <Types>Condensing</Types>
    </Region>
</Steppable>
```

will result in the initial configuration as show on Figure 6.

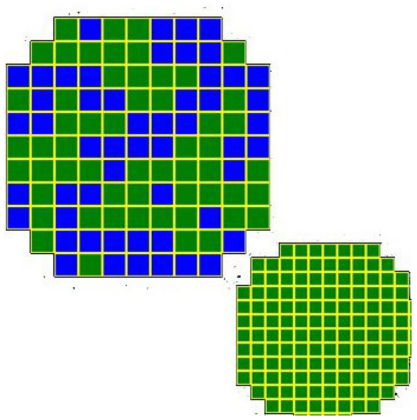


Figure 6. Defining Multiple regions using BlobInitializer steppable.

Note: When user specifies more than one cell type between `<Types>` tags (notice, the types have to be separated with ',' and there should be no spaces) then cells for this region will be initialized with types chosen randomly from the provided list (here the choices would be `Condensing`, `NonCondensing`).
Remark: If one of the type names is repeated inside `<Types>` element this type will get greater

weighting means probability of assigning this type to a cell will be greater. So for example `<Types> Condensing,NonCondensing,NonCondensing,NonCondensing </Types>` `Condensing` will assigned to a cell with probability 1/4 and `NonCondensing` with probability ¾ - see example and Figure 7 below:

```
<Steppable Type="BlobInitializer">
     <Region>
       <Radius>40</Radius>
       <Center x="50" y="50" z="0"/>
       <Gap>0</Gap>
       <Width>5</Width>
       <Types>Condensing,NonCondensing,NonCondensing,NonCondensing</Types>
    </Region>
 </Steppable>
```

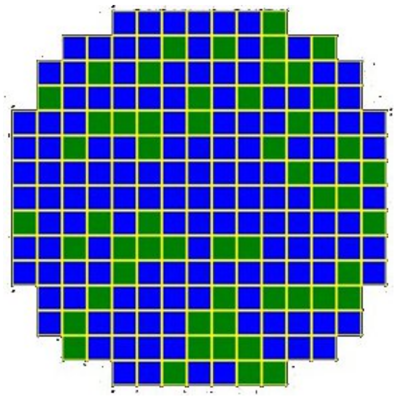


Figure 7. Initial condition where 75% cells are `NonCondensing` and 25% are `Condensing`

Now let's do some exercises.

1.4.1 Exercise 1

As we did for foams, explore the effects of changing the temperature on the pattern. We now have additional parameters (the strength of the volume constraint and the target volume as well as the surface energy), although, as before, only the ratio of energy to temperature matters.

a) If the temperature T=0 what happens (should freeze)
b) If t is intermediate what happens?
c) If t is very large what happens? (edges should become rough)
d) What is the range of T over which these three regimes occur (is there a large intermediate regime)?

1.4.2 Exercise 2

Now look at what happens as the strength of the area constraint changes?
a) If λ_v=0 what happens? (should disappear)
b) If λ_v is intermediate, what happens?
c) If λ_v is very large what happens? (Should freeze)
d) What is the range of λ_v over which these three regimes occur (is there a large intermediate regime)?

1.4.3 Exercise 3

Now we will look at the effects of the contact energies. You have five energies to play with.

1. Let $0<J_{NC}<J_{CC}<J_{NN}<J_{NM}=J_{CM}$. What happens? Make sure that you adjust lambda and T to be in the "middle" regime
2. Let $0<J_{CC}<J_{NC}<J_{NN}<J_{NM}=J_{CM}$. What happens? Now slowly vary J_{NC} from being just above J_{CC} to being just below J_{NN}. Does the final pattern change? Does the kinetics of the pattern evolution change?
3. Let $0<J_{CC}<J_{NN}<J_{NC}<J_{NM}=J_{CM}$. What happens? Is the result the same or different from b)? If J_{NC} is just above J_{NN} is the result different from if J_{NC} is much bigger than J_{NN}?
4. Let $0<J_{CC}<J_{NC}<J_{NN}<J_{CM}<J_{NM}$. What happens? Is the result different if the difference between J_{CM} and J_{NM} is small from what happens if it is very large?
5. Let $0<J_{CC}<J_{CM}<J_{NN}<J_{NM}<J_{ND}$. What happens? Why?
6. Let $0<J_{NM}=J_{CM}<J_{CC}<J_{NC}<J_{NN}$. What happens? Why?

Which contact energy hierarchy leads to simulations shown on Figure 7 and 8?

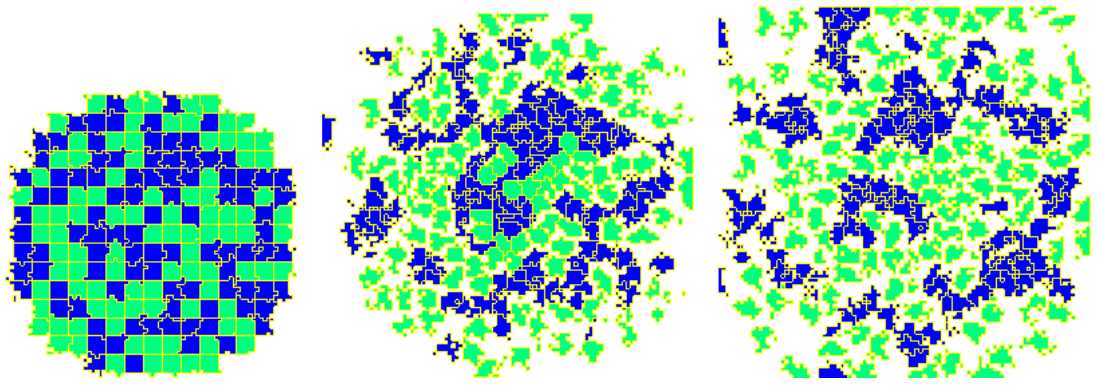


Figure 7. Cell dissociation example. Snapshots were taken at T=0 MCS ,T= 100 MCS and T=300 MCS

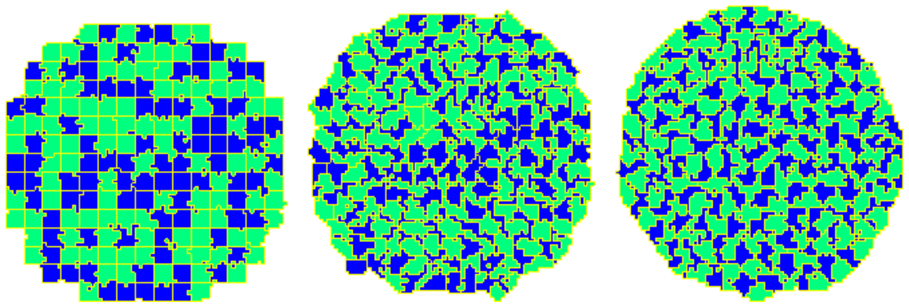


Figure 8. Checkerboard patterning. Snapshots were taken at T= 0 MCS, T=10 MCS and T=300 MCS. Notice how quickly final pattern is established

Remark:
For exercises4,5,6 please use file
/Users/mswat/CompuCellFull5_install/cellsort_engulfment_diffusion_2D.xml

as a template (copy it into your private directory). Depending on the exercise you will need to modify it. However you will not need to type everything from scratch.

1.4.4 Exercise 4

Now repeat Exercise 3 starting with the case of engulfment (light cells on the bottom and dark cells on the top)?

Hint: *You can quickly initialize blob of cells with cells of one type at the bottom and of another at the top. Simply use, the following syntax in your BlobInitialzer Steppable*:

```
<Steppable Type="BlobInitializer">
  <Gap>0</Gap>
  <Width>5</Width>
  <CellSortInit>yes</CellSortInit>
  <Radius>40</Radius>
  <Engulfment BottomType="Condensing" TopType="NonCondensing"
  EngulfmentCoordinate="y" EngulfmentCutoff="50"/>
</Steppable>
```

As you can see it looks almost the same as in the previous exersices except there is new line which describes engulfment. As you can infer from this syntax, you may specify which cells are at the top, which are at the bottom, and by changing the value of `EngulfmentCutoff` you may fix how many cells of a given type there should be. Try to play with this value (remember it must be positive integer) and figure out by yourself how it works. Try to find also what `EngulfmentCoordinate` means . The allowed values are `x`, `y`, `z`. See also Figure 8 below.

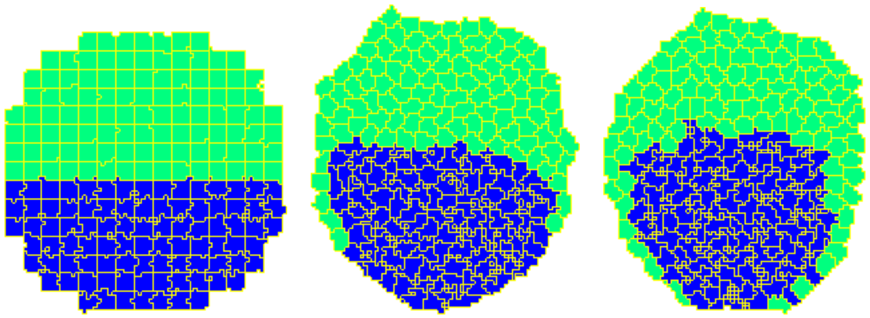


Figure 9. Cell engulfment simulation. Snapshots were taken at T=0 MCS, T=3000 MCS and T=9990 MCS.

1.4.5 Exercise 5

Actually, when cells stick together, the energy between them should be negative.

1. Repeat Exercise 3 for all energies negative (same hierarchy). What happens?

2. Now include the surface area constraint as well. Repeat the exercises in Exercise 3. Are the results different? Vary the target surface area as a function of the target volume? What happens? For a fixed target surface area, vary the strength of the corresponding constraint. Can you identify the same three regimes as in exercises 1 and 2? What are these three regimes in this case?

3. Finally, let J_{NN}, J_{CC} and J_{NC} be negative and J_{NM} and J_{CM} be positive. How do your results compare to part 1) and 2)?

1.4.6 Exercise 6 (requires Python scripting)

The diffusion of cells is a critical property. Use only one cell type and a fixed value of the target volume and surface area and their constraints. Measure the diffusion constant of a single cell (the slope of a plot of mean displacement vs t^2).

1. How does the diffusion constant vary with T?

2. How does the diffusion constant vary with J?

3. How does the diffusion constant vary with lambda?

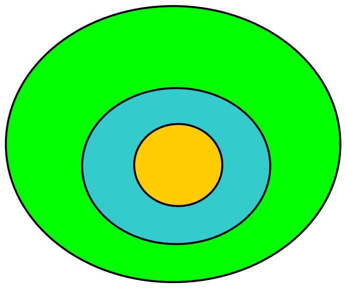
4. We usually think that only energy differences matter, not absolute energy values. Do two diffusion experiments, one where J>0 and the other when J<0, but all J and other parameters are the same. Are the diffusion constants the same? Why?

5.

This exercise should be done with Python scripting.

1.4.7 Exersice 7

a) Find as set of adhesion energies that give rise to a three-layer concentric structure:



b) If you have made the adhesion energies between medium and the different cell types different, set these energies to be the same and repeat.

1.4.8 Exercise 7

Repeat Exercise 6 for four concentric cell layers.

Hints:
1) Look at the ratios of energies between J_{11} and J_{22} for the two layer case. Can this apply to the three and four layer cases?
2) If the number of cells of each type is the same, the outer layers will become very thin. To create more of the outer layer cell types, use the feature in the blob initializer: list types of which you want more copies multiple times (number of cells of that type is proportional to the number of the times that type is listed).

2 Chemotaxis in CompuCell3D

Chemotaxis is one of most important phenomena in almost every aspect of cell biology. In layman terms, the chemotaxis is a directed movement of cell or an organism toward (or away from) a chemical source (or up/down the chemical concentration gradient). If the cell moves up the gradient, we say the chemical is a chemo-attractor, if the opposite takes place the substance is called chemo-repellent. The underlying mechanism of chemotaxis might be very complicated and in fact is a subject of major research effort. Fortunately, from GGH point of view chemotaxis is quite easy implement in the model. As you probably, know, all we need to is to introduce new term in the Hamiltonian, which would favor these spin flips that occur in the direction of increasing/decreasing concentration (depending if we model chemo-attractants or chemo-repellents)

The simplest term which does this looks as follows:

$$\Delta E = -\lambda\left(c\left(x_{destination}\right) - c\left(x_{source}\right)\right) \qquad \textbf{(1)}$$

where λ denotes chemotaxis strength coefficient and $c\left(x_{destination}\right)$ is concentration at position at which the spin flip takes place and $c\left(x_{source}\right)$ is a concentration at flip-neighbor.
An alternative form of chemotaxis term which is frequently used in the GGH simulations is the following one:

$$\Delta E = -\lambda\left(c\frac{\left(x_{destination}\right)}{a + c\left(x_{destination}\right)} - c\frac{\left(x_{source}\right)}{a + c\left(x_{source}\right)} \right) \qquad (2)$$

a denotes here saturation coefficient.

In this set of exercises you will play with chemical fields inside CompuCell3D, diffusion, decay, constants, and chemotactic properties of cells. One of the exercises will be described only (no xml file will be provided) and you are expected to write PIF, and xml files for this particular simulation. You will not have to use any scripting language for PIF file generation.

2.1.1 Exercise 1

Before you begin, make sure to copy *amoebae_2D.xml, amoebaConcentrationField_2D.txt* and *amoebae_2D.pif* files from *Demos/amoebae* to your private directory.

This simulation consists of two cells – amoeba and bacteria – surrounded by a chemical, which serves as chemo-attractant/chemo-repellent.

Open up the simulation in the Player, look at the concentration field and try changing chemotaxis parameters.

a) Start with very small λ Can you observe chemotaxis at all?
b) Start increasing λ What happens?
c) Reverse sign of λ for one type of cell (bacteria for example). Is chemo-attractant becoming chemo-repellent?

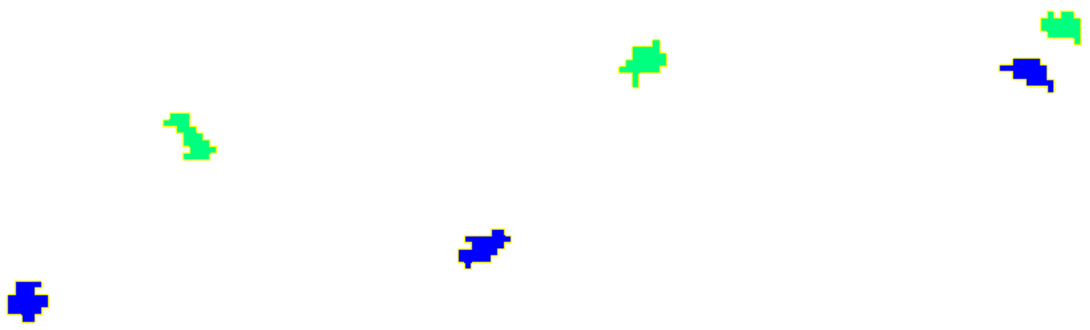


Figure 10. Demonstration of chemotaxis in CompuCell3D. Both cells are attracted to to upper left corner of the lattice. Snapshots were taken at T=10 MCS, T=70 MCS and T=200 MCS

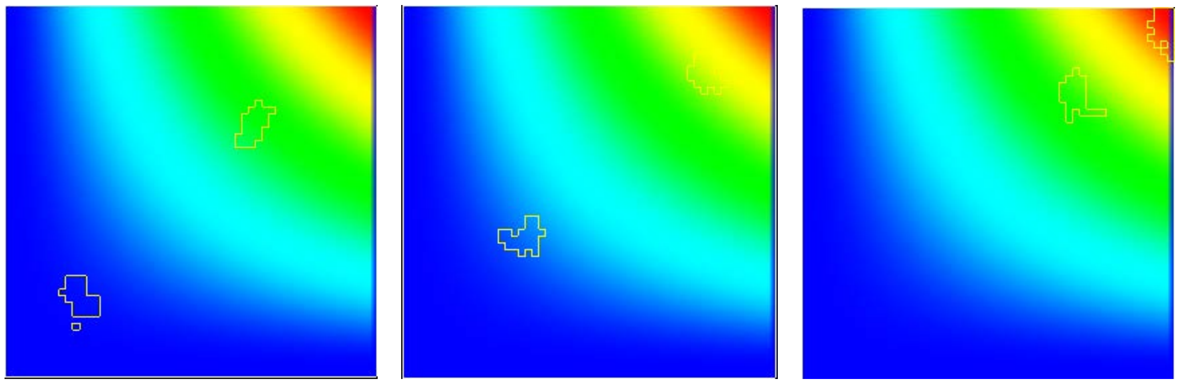


Figure 11. Chemoatractant concentration view. Snapshots were taken at T=10 MCS, T=70 MCS and T=200 MCS

Python equivalent of *Demos/amoebae/amoebae_2D.xml* configuration file is stored in *Demos/PythonOnlySimulationsExamples/amoebae-2D-new-syntax.py*

2.2 Solving diffusion equations using CompuCell3D

As a next step we will see how we can use CompuCell3D as a simple PDE solver. We will qualitatively examine the solution of the diffusion equation (with pulse initial condition) and check numerical stability limits.
Copy the following files: *diffusion_2D.xml diffusion_2D.pulse.txt* from CompuCell3D installation directory to your private directory. Open xml file in the editor and see what sections are needed in order to use CompuCell3D as a PDE solver. As you can see most of the plugins have been removed. You need to, however, include CellType plugin and list of all the cell types that you are going to use in PDE solver description. By default you are required to list **`Medium`** Observe that in the Potts section of the xml file, the value of Flip2DimRatio element has been changed to 0.0.

```
<Flip2DimRatio>0.0</Flip2DimRatio>
```

This prevents CompuCell3D from doing any spin flips. Nevertheless steppables (PDE solvers are steppables) will still run.

The way to impose initial condition for the diffusion equation is to use initial concentration file and specify its name in the `DiffusionData` section of the FlexibleDiffusionSolver:

```
<Steppable Type="FlexibleDiffusionSolverFE">
     <DiffusionField>
        <DiffusionData>
            <FieldName>FGF</FieldName>
            <DiffusionConstant>0.000</DiffusionConstant>
            <DecayConstant>0.100</DecayConstant>

            <ConcentrationFileName>
            diffusion_2D.pulse.txt
            </ConcentrationFileName>

          <!--DoNotDecayIn>Medium</DoNotDecayIn-->
        </DiffusionData>
    </DiffusionField>
 </Steppable>
```

In our case file name is *diffusion_2D.pulse.txt*. The format of the file is really simple:

```
x  y  z  concentration
```

where `x`, `y`, z denote value of the coordinate of a given pixel, and concentration denotes numerical value of chemical concentration at that pixel.

To create a pulse in the middle of the 55x55 lattice all we need is in fact one single line:

```
27   27   0   2000.0
```

As you can see it is quite straightforward to use CompuCell3D as a PDE solver.

Now let's do some exercises.

Remark: you have to switch views in the Player from Cell Field to chemical concentration (e.g. FGF) to see simulation results.

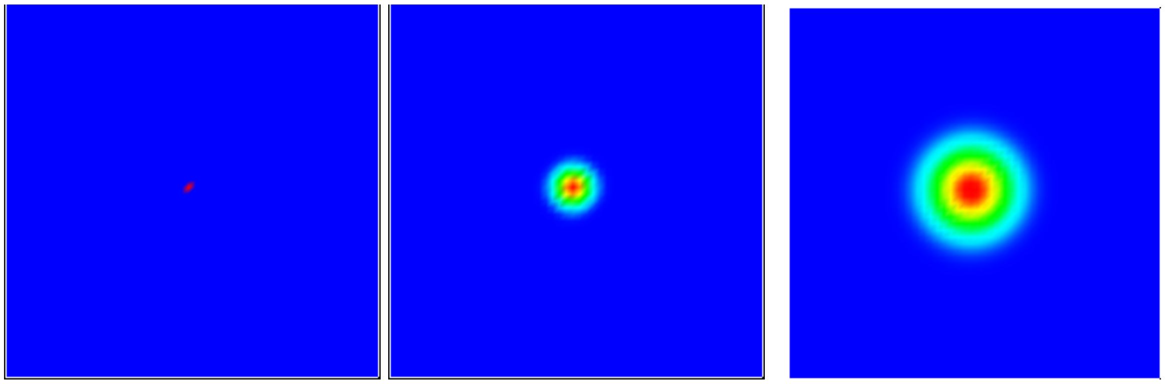


Figure 12. Solving diffusion equation using CompuCell3D. Snapshots of the FGF field were taken at T=0 MCS, T=200 MCS and T=990 MCS

For completeness, and because of relatively small size, we show configuration file *Demos/PythonOnlySimulationsExamples/diffusion-2D-new-syntax.py* where we use Python syntax to setup simulation which solves diffusion equation.

Remark: Depending on your preference you may use either CC3DML or Python to configure CompuCell3D simulations. We will keep using CC3DML to explain CompuCell3D concepts as CC3DML can be easily indented for readability while Python syntax, being indentation sensitive, cannot. Regardless whether you use Python or CC3DML you are passing the same information to CC3D.

```
def configureSimulation(sim):
   import CompuCellSetup
   from XMLUtils import ElementCC3D

   cc3d=ElementCC3D("CompuCell3D")
   potts=cc3d.ElementCC3D("Potts")
   potts.ElementCC3D("Dimensions",{"x":55,"y":55,"z":1})
   potts.ElementCC3D("Steps",{},1000)
   potts.ElementCC3D("Temperature",{},0)
   potts.ElementCC3D("Flip2DimRatio",{},0.0)

   cellType=cc3d.ElementCC3D("Plugin",{"Name":"CellType"})
   cellType.ElementCC3D("CellType",\
   {"TypeName":"Medium", "TypeId":"0"})

   flexDiffSolver=cc3d.ElementCC3D("Steppable",\
   {"Type":"FlexibleDiffusionSolverFE"})
   diffusionField=flexDiffSolver.ElementCC3D("DiffusionField")
   diffusionData=diffusionField.ElementCC3D("DiffusionData")
   diffusionData.ElementCC3D("FieldName",{},"FGF")
   diffusionData.ElementCC3D("DiffusionConstant",{},0.10)
   diffusionData.ElementCC3D("DecayConstant",{},0.0)
   diffusionData.ElementCC3D("ConcentrationFileName",{},\
   "Demos/PythonOnlySimulationsExamples/diffusion_2D.pulse.txt")

   CompuCellSetup.setSimulationXMLDescription(cc3d)
```

Remark: Notice above, that in Python “\” is a line continuation symbol and we had to use it few times to properly format the listing

2.2.1 Exercise 2

Run simulation with different values of diffusion constants:
start with small diffusion constant ~ 0.01
Increase diffusion constant until you reach numerical instability regime. How can you tell that solution is unstable? What is the critical value of the diffusion constant at which numerical method breaks? Do you know why solution becomes unstable?
Add decay term to the diffusion equation. Does the presence of the decay term influence stability/instability?

Now try changing boundary conditions on the lattice from no flux to periodic and see the concentration pattern:

```
<Boundary_x>Periodic</Boundary_x>
<Boundary_y>Periodic</Boundary_y>
```

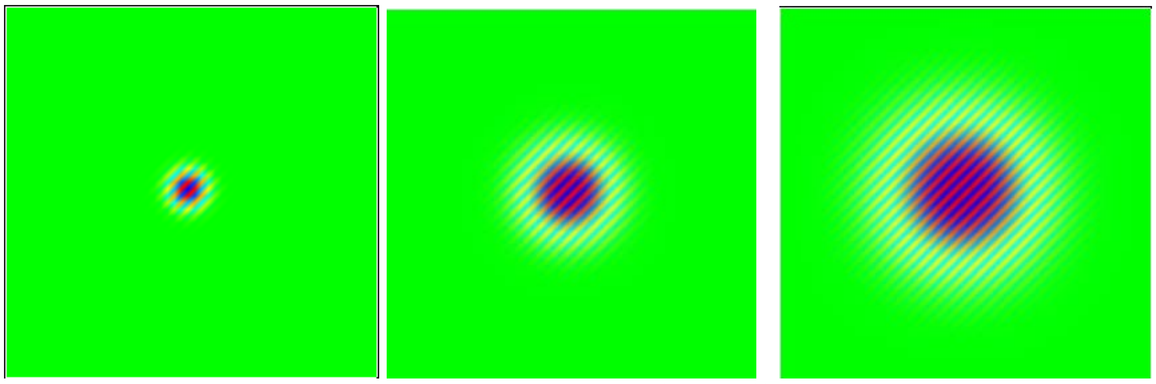


Figure 13. Numerical instabilities in the solution of the diffusion equation in 2 dimensions. Snapshots were taken at T=0 MCS, T=70 MCS and T=200 MCS

2.2.2 Exercise 3

Let’s come back to amoebae_2D.xml simulation. One way to track cells in the chemical field view is to let cells chemotax and let the chemical decay inside the cell only. This way the place that was occupied by the cell would be marked as a depleted concentration region. To enable “cell tracking” make decay constant non-zero and uncomment the line

```
<!--DoNotDecayIn>Medium</DoNotDecayIn-->
```

We hope you remember how to comment and uncomment xml elements. With this line present, the chemical will decay everywhere except Medium, which means it will decay only inside cells. Now you can switch to concentration view and see how cell tracking works. Are there any pathological effects that you observe? What values of decay constants have you used?

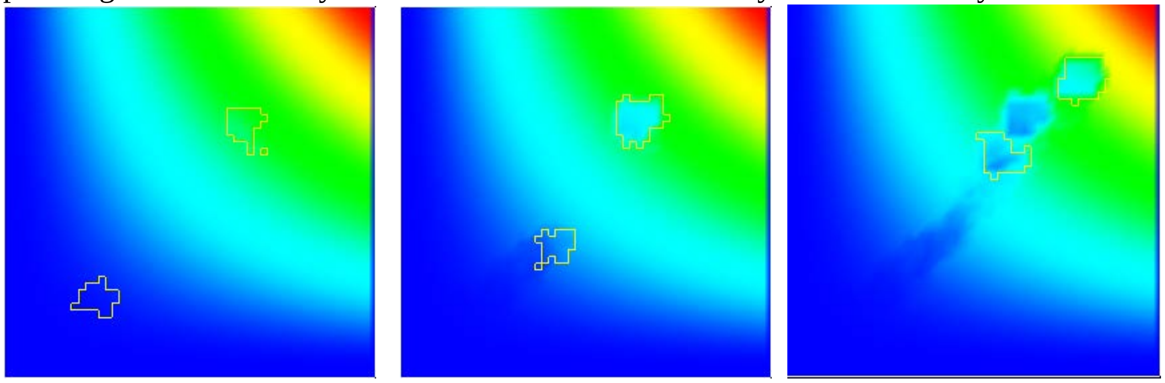


Figure 14. Demonstration of “chemical tracking” in CompuCell3D. Snapshots were taken at T=10 MCS, T=70 MCS and T=200 MCS. The smudge of depleted concentration is achieved by enabling concentration decay in the region occupied by cells.

2.3 Bacterium Macrophage Simulation

The xml file for this exercise will be written by you. The idea of the simulation is the following: you have two cells (bacterium and macrophage) placed in the maze. Bacterium secretes a chemical (call it ATTR – for attractant) which is free to diffuse everywhere except walls of the maze. Another cell, the macrophage is attracted to ATTR and chemotacts up the chemical gradient. Eventually macrophage will reach stationary bacterium. Your task here will be to write xml simulation file and find reasonable parameters for diffusion/decay of chemical and chemotaxis.

Shortcut: if you are in a hurry you may use prewritten xml file that comes with CompuCell3D – *Demos/bacterium_macrophage/bacterium_macrophage 2D.xml*. However, please do make sure you understand what's in it. In particular see how we wrote pif file *bacterium_macrophage_2D.pif*.
Python equivalent of *Demos/bacterium_macrophage/bacterium_macrophage 2D.xml* is stored in *Demos/PythonOnlySimulationsExamples/bacterium_macrophage-player-new-syntax.py*

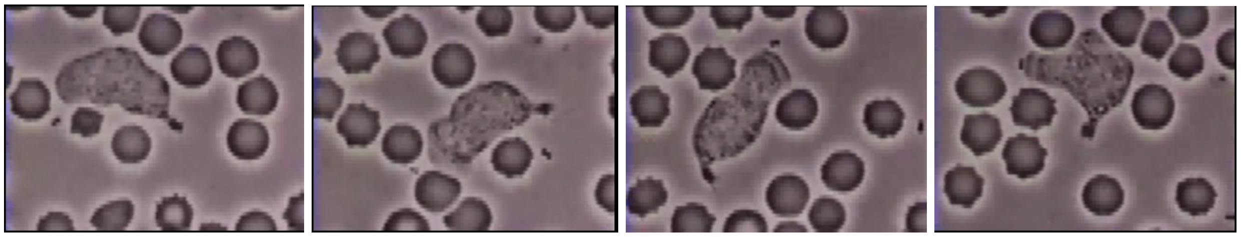


Figure 15. Snapshots of experiment where macrophage chases bacterium in the blood stream. In the last snaphot macrophage is about to capture the bacterium.
This video is taken from a 16-mm movie made in the 1950s by the late David Rogers at Vanderbilt University. It was given to me via Dr. Victor Najjar, Professor Emeritus at Tufts University Medical School and a former colleague of Rogers. It depicts a human polymorphonuclear leukocyte (neutrophil) on a blood film, crawling among red blood cells,

notable for their dark color and principally spherical shape. The neutrophil is "chasing" *Staphylococcus aureus* microorganisms, added to the film. The chemoattractant derived from the microbe is unclear but may be complement fragment C5a, generated by the interaction of antibodies in the blood serum with the complement cascade, and/or bacterial *N*-formyl peptides. Blood platelets adherent to the underlying glass are also visible. Notable is the characteristic asymmetric shape of the crawling neutrophil with an organelle-excluding leading lamella and a narrowing at the opposite end culminating in a "tail" that the cell appears to drag along. Contraction waves are visible along the surface of the moving cell as it moves forward in a gliding fashion. As the neutrophil relentlessly pursues the microbe it ignores the red cells and platelets. However, its leading edge is sufficiently stiff (elastic) to deform and displace the red cells it bumps into. The internal contents of the neutrophil also move, and granule motion is particularly dynamic near the leading edge. These granules only approach the cell surface membrane when the cell changes direction and redistributes its peripheral "gel." After the neutrophil has engulfed the bacterium, note that the cell's movements become somewhat more jerky, and that it begins to extend more spherical surface projections. These bleb-like protruberances resemble the blebs that form constitutively in the M2 melanoma cells missing the actin filament crosslinking protein filamin-1 (ABP-280) and may be telling us something about the mechanism of membrane protrusion. **Information credit**: Thomas P. Stossel (Brigham and Women's Hospital and Harvard Medical School), June 22, 1999

2.3.1 Exercise 4

a) First, let’s construct the maze and place there the two cells. You do not need to get fancy here. A maze is simply few rectangular blocks of type `Wall`. Make sure that in the `CellType` plugin type `Wall` is declared as frozen i.e. it does not participate in spin flips. You can freeze any cell type by adding extra attribute `Freeze=””` to the line where you declare this cell type, as shown below:

```
<Plugin Name="CellType">
    <CellType TypeName="Medium" TypeId="0"/>
    <CellType TypeName="Bacterium" TypeId="1"/>
    <CellType TypeName="Macrophage" TypeId="2"/>
    <CellType TypeName="Wall" TypeId="3" Freeze=””/>
 </Plugin>
```

Now, you need write PIF file for the simulation. I have included sample PIF file below. If you do not remember PIF file syntax, see earlier exercises on foams form more explanations.

```
0 Wall 10 20 10 30 0 0
1 Wall 25 40 35 50 0 0
.
.
.
10 Bacterium 5 5 5 5 0 0
11 Macrophage 70 70 70 70 0 0
```

Notice that `z_low` and `z_high` are 0 and 0 – which means we are in 2D. There is nothing that would prohibit us from being in 3D but in 2D the simulation will run faster.

To see initial configuration you may turn off spin flips (how?) and run simulation. Make any corrections as necessary. Once your maze looks nice and cells are positioned properly you may start coding FlexibleDiffusionSolver

b) **The main difference is that there is no initial chemical concentration in the system** i.e. the line

```
<ConcentrationFileName>amoebaConcentrationField_2D.txt</ConcentrationFileNam>
```

should be removed.

Now you need to tell CompuCell3D that Bacterium should secrete chemical ATTR at a certain rate. The way to do it is to use `SecretionData` section inside FlexibleDiffusionSolver. With that , every MCS each pixel occupied by `Bacterium` will increase its concentration by given amount (here it will be 0.5)

```
<Steppable Type="FlexibleDiffusionSolverFE">
  <DiffusionField>
    <DiffusionData>
      <FieldName>ATTR</FieldName>
      <DiffusionConstant>0.100</DiffusionConstant>
      <DecayConstant>0.000</DecayConstant>
      <DoNotDiffuseTo>Wall</DoNotDiffuseTo>
    </DiffusionData>

    <SecretionData>
      <Secretion Type=”Bacterium”>0.5</Secretion>
    </SecretionData>

  </DiffusionField>
</Steppable>
```

Notice that we have also added line which keeps chemical from diffusing into the wall.
c) Add chemotaxis for `Macrophage`.
d) Set reasonable surface and volume constraints for `Bacterium` and `Macrophage`.
e) Is it necessary to add Contact plugin?
f) Put together xml simulation description file and run the actual simulation. Which parameters need to be fine-tuned?

g) So far we have been using the simplest chemotaxis energy formula. Now, lets try energy term with saturation coefficient. To accomplish that you need to specify extra attribute in the chemotaxis description:

```
<ChemotaxisByType Type="Bacterium" Lambda="200" SaturationCoef=”10”/>
```

The appearance of additional attribute **`SaturationCoef`** tells CompuCell3D to use alternative formula for chemotaxis. Notice also that the meanings of λ coefficients in the two chemotaxis formulas are different and you will need to find new values of λ for the new form of chemotaxis energy term.

h) How would you simulate process in which Macrophage eats Bacterium?
i) What can be done to prevent cells from sticking to the borders of the lattice or to the cells of type Wall?

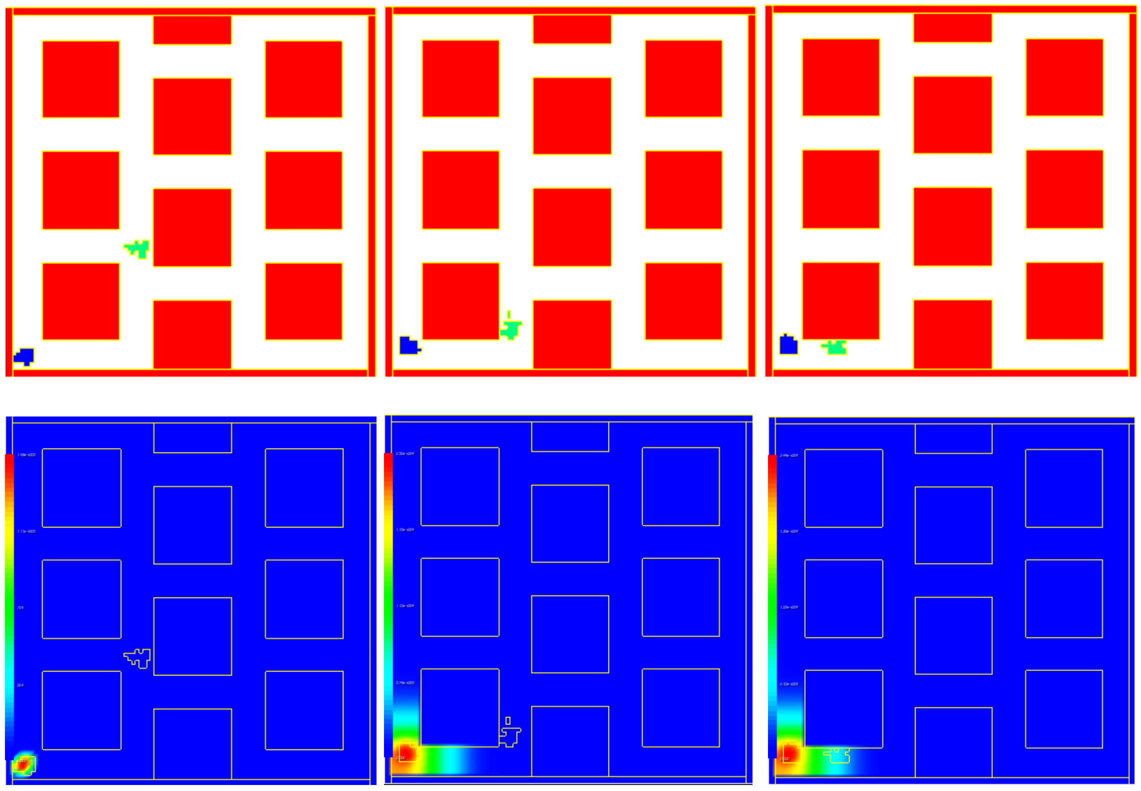


Figure 16. Most basic bacterium-macrophage simulation. Red blood cells are static in this simulation. Bacterium is secreting chemical signal which attracts macrophage. Sizes of bacterium and macrophage are roughly the same. We show, both, cell field and chemoattractant (here called `ATTR`) views. Snapshots were taken at T=10 MCS T=660 MCS and T=720 MCS

2.3.2 Exercise 5

In this exercise we will make a series of modification to the `bacterium_macrophage_2D.xml` file and will come up with some interesting models.

a) make decay constant non-zero (*bacterium_macrophage_2D_v2.xml*)
b) Increase size of macrophage and make bacterium smaller

(*bacterium_macrophage_2D_v4.xml*). Make sure you understand how to use VolumeFlex and Surface Flex plugins. You may need to fine tune surface and volume parameters so that cells do not disappear or fragment.
c) Add one more macrophage and three bacteria cells (*bacterium_macrophage_2D_v6.xml*) and see corresponding pif file. Try placing macrophages in different initial positions .
d) Try changing decay constant (*bacterium_macrophage_2D_v7.xml*). What are the results?
e) If you observer cells sticking to the walls try adjusting contact energies . How? (see *bacterium_macrophage_2D_v8.xml*)
f) Now instead of static Walls cells in the middle of the lattice introduce Red Cells that can actually move. (*bacterium_macrophage_2D_9.xml*).
g) Introduce Chemoreppelant secreted by macrophage. Pick secretion values, decay constants and chemotaxis constant. Note, that only bacterium is sensitive to chemorepelant. How do you code it in xml? (*bacterium_macrophage_2D_v10.xml*).

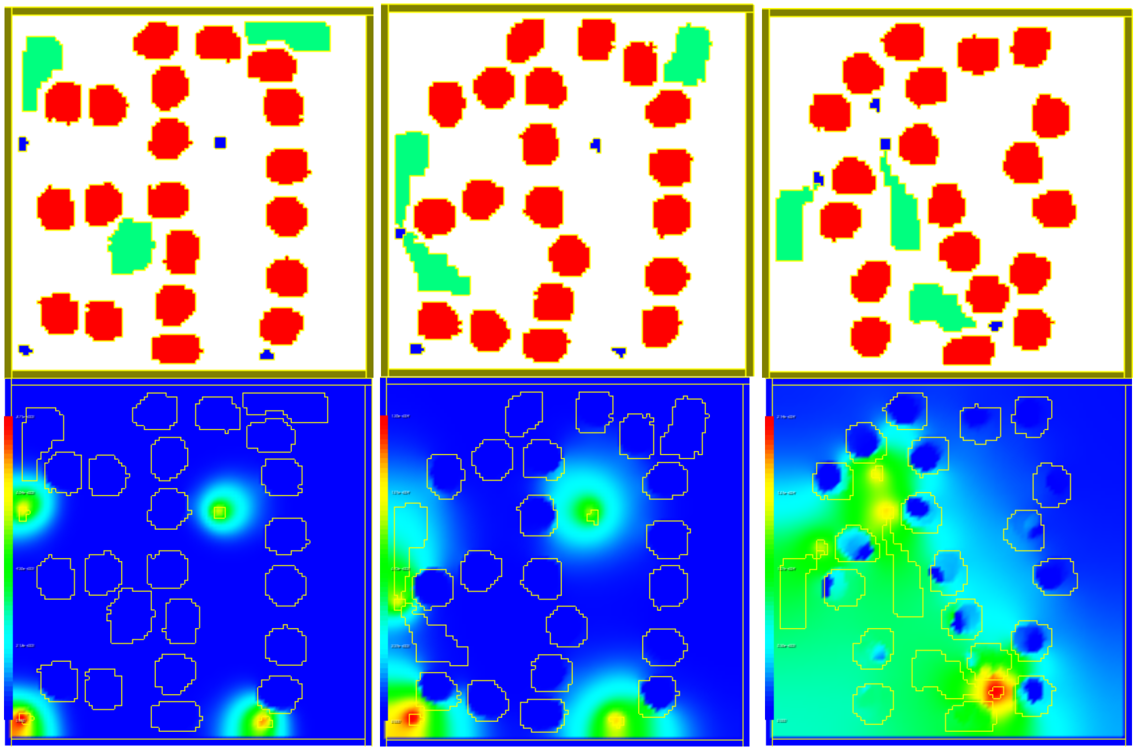


Figure 17. Realistic-looking bacterium macrophage simulation. Red blood cells are motile. We show, both, cell field and chemoattractant (here called ATTR) views. Snapshots were taken at T=400 MCS , T=1550 MCS and T=7840 MCS.

This exercise demonstrates what it takes to build and test a model using CompuCell3D. Yes, it is an iterative process, where you start from simple model and start adding features, but as you can see it is not that painful.