<table>
<thead>
<tr>
<th>Section</th>
<th>File Name</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>6.10.4.</td>
<td><code>Output_02.m</code></td>
<td>68</td>
</tr>
<tr>
<td>6.10.5.</td>
<td><code>Output_03.m</code></td>
<td>69</td>
</tr>
<tr>
<td>6.10.6.</td>
<td><code>Output_04.m</code></td>
<td>69</td>
</tr>
<tr>
<td>6.10.7.</td>
<td><code>Output_05.m</code></td>
<td>70</td>
</tr>
<tr>
<td>6.11.</td>
<td><strong>ABM Of Bacteria Animation Code</strong></td>
<td></td>
</tr>
<tr>
<td>6.11.1.</td>
<td><code>ABM_Animation_2013_07_17_V1.m</code></td>
<td>71</td>
</tr>
<tr>
<td>6.11.2.</td>
<td><code>CellColorsMode.m</code></td>
<td>78</td>
</tr>
<tr>
<td>6.11.3.</td>
<td><code>ProtoEnvironment.wrl</code></td>
<td>79</td>
</tr>
<tr>
<td>6.11.4.</td>
<td><code>ProtoSphere.wrl</code></td>
<td>80</td>
</tr>
<tr>
<td>6.11.5.</td>
<td><code>ProtoCylinder.wrl</code></td>
<td>81</td>
</tr>
<tr>
<td>6.11.6.</td>
<td><code>ProtoWall.wrl</code></td>
<td>82</td>
</tr>
</tbody>
</table>
ABSTRACT

The increased availability of observational and experimental techniques at the single-cell level, including automated microscopy and microfluidics, motivates the development and application of individual/agent based models (IBMs/ABMs). ABMs simulate populations of individual cells and can be used to interpret observations from these experiments and put them into broader ecological context. This paper presents an ABM of bacteria. Individual cells are represented by a spherocylinder. Processes include growth, division, biomechanical (shoving) and biochemical (quorum sensing) interaction between cells and between cells and their environment. A post-processing routine generates graphs and animations that can be compared directly to individual-based observations. Two case studies are presented, where the model is used to simulate experiments from the literature. The first experiment investigates aging in *E. coli* by following individual cells as they grow and reproduce over time through automated time-lapse microscopy. The second experiment investigates synchronization via quorum sensing in a population of genetically engineered *E. coli* cells. These results show that agent based models (ABMs) of bacteria can be used to reproduce modern experimental data at the single-cell level.
ACKNOWLEDGEMENTS

I would like to thank my advisor Professor Ferdi Hellweger for his guidance with immense expertise and patients in developing the model and the thesis.

I would like to especially thank Saurabh Mahapatra from MathWorks for his help in early stages of the visualization development.

Also I would like to acknowledge the special contribution of Stefan Savev that made possible implementation of MATLAB Coder in the model and thus making the model efficient and more fruitful.

I want to express also my gratitude to Jonathon Pendlum who helped me further in developing more features of the model suitable for MATLAB Coder translation.

I want to thank to Professor Kai-tak Wan for his great, chalenging and intellectually stimulating class, “Solid mechanics of cells and tissues”, for the inspirational opportunity to talk with him about science and my research, for his kind, warm and wonderfull personality.

I would like to thank all my colleagues and friends in the Civil & Environmental Engineering Department at Northeastern University: Xiaodan Ruan, Neil Fredrick, Nan Cai, Alexandra Schellenger, Katrina Smith-Mannschott, Salma Mozaffari, Ali Chiplack and Naji Kasem for the wonderful interaction during my stay in the department.

I would like to thank also to Professor Behrooz Satvat for his valuable advices and also to Cécile Braun-Jones, Lisa O’Neill, Esther Cohen and team from Graduate School of Engineering for their assistance in all administrative matter. Also I am extremly grateful to Daniel Saulnier and Robert Tillman for their time and willingness to help me.

I owe my gratitude to my friends Milen and Desislava Raytchevi for the thrilling conversations about science and research, also for their support and advices.

I am also grateful for the support and care to all my friends: Alexandra Takeva, Cecilia Busuioc, Irinéa Batista, Maria Somleva, Nikolay Nikolaev, Sev Gercheva, Stanislava Nikolova, Teodora Zheleva, Vesela Hristova.

I would like to thank my husband, Kalin Kostadinov, for his continuous support and his love that made my life brighter and worth living.
1. INTRODUCTION

Microbes are individuals that interact with themselves and their environment resulting in complex system behavior. Over the past decades microbial ecology has changed immensely with the advancement of single-cell observational and experimental techniques. This has led to an increased appreciation of the individuality of microbes. There is now a growing understanding that intrapopulation heterogeneity plays an important role in many microbial ecology phenomena (Hellweger and Bucci, 2009; Lidstrom and Konopka, 2010).

The traditional population level modeling (PLM) approach does not account for heterogeneity in the bacteria population and it does not produce predictions at the single-cell level comparable to individual based observations. These needs can be fulfilled with the individual/agent based modeling (IBM/ABM) approach. ABMs simulate populations of individual cells and can be used to interpret observations from single-cell experiments and put them into broader ecological context (Hellweger and Bucci, 2009).

This paper presents an ABM of bacteria. The model simulates individual cells as spherocylinders and includes growth, division, shoving and quorum sensing. The model is applied to two case studies from the literature and the results are compared directly to single-cell observations from these experiments.

The experiment in the first case study investigates aging and death in *E. coli* as a model organism that divides symmetrically, where a single *E. coli* cell grows to a 2D colony on a microscope slide cavities. Observations include tracking individual cells and measuring their physical parameters in an automated fluorescence microscopy system through up to nine generations of growth (Stewart et al., 2005). The model was setup to reproduce the development of the colony from a single cell and animations from model output were compared directly with the video recorded in the experiment. The comparison between the movies shows close resemblance in the bacteria colony development. However, the model colony forms a more round shape than observed, which maybe due to a lack of stickiness (adhesion) between agents in the model.

The second case study is an experiment that investigated the intercellular communication (quorum sensing) in a population of genetically engineered *E. coli* cells. Cells produce and sense
a chemical signal, acyl-homoserine lactone (AHL), which activates its own repressor (AiiA). This network architecture is the core regulatory module for creation of circadian clock networks and is similar to other synthetic oscillators (Danino et al., 2010). The results of the model show that a burst of oscillation of LuxI starting from the middle of the colony is spreading to the periphery similarly to the observed experimental results. In the model the peaks of the luminescence producing substance are more pronounced and spatially uniform, and synchronization in burst production occurs more rapidly than observed. The differences between the model and experimental result may be because the model is not simulating the heterogeneity in the cell density in the chamber, and this leads to much more rapid synchronization of the luminescent signal than in the experiment.
2. MODEL DESCRIPTION

2.1. Overview

The model simulates bacterial cells and parcels of the environment. Agents are rod-shaped bacteria of strain *E. coli*. Individual cells are modeled explicitly and each bacterium is represented by a spherocylinder with a fixed diameter, $d_0$. The distance between the caps (hemispheres) is denoted as $l_c$ and changes with growth and division. The total distance between the two poles of the cells is symbolized as $l_t$. Points A, B, C and D are used to define the position and orientation of the cell, these points represent the centers of the hemispheres, the middle point of the axis, middle point at the cell surface (see Fig. 1). Cell state variables are listed in Table 17.

![Figure 1. Schematic representation of bacterial cell](image1)

The environment is characterized by its total length, width and height, denoted respectively with $X_t$, $Y_t$ and $Z_t$. The dimensions and number of the parcels in x, y and z direction are represented by $idx_g2$, $idy_g2$, $idz_g2$, $inx_g2$, $iny_g2$, and $inz_g2$ variables (see Fig. 2). Parcel state variables are listed in Table 24.

![Figure 2. Schematic representation of the chamber](image2)

The time discretization and the spatial discretization of the environment and its basic characteristics are presented in Table 13.
2.2. Growth

The volume of a cell \((v)\) changes as a function of the specific growth rate \((\mu)\), which is a nonlinear function of the maximum specific growth rate \((\mu_{\text{max}})\) and the pressure \((p)\) exerted on the cell surface by other cells or the walls of the chamber.

\[
\frac{dv}{dt} = \mu v \quad \{1\}
\]

and

\[
\mu = \mu_{\text{max}} \frac{k_{mp}^{n_p}}{k_{mp}^{n_p} + p^{n_p}}, \quad \{2\}
\]

where \(k_{mp}\) is the half-saturation constant \(n_p\) is the exponent. Cells grow by elongation. First the new volume is computed. Then, assuming that the radius of the cell (spherocylinder) is constant, the new elongated lc is determined and then the new cell coordinates are computed and updated.

2.3. Division

When a cell reaches a threshold volume it divides. The threshold volume is defined as double the minimum cell volume (see Table 19). The cell splits into two almost identical cells in terms of volume. Once the new volume is determined the coordinates of the cells are computed and updated.

The new volumes of the mother and daughter cells is:

\[
V_{\text{Cell1}} = Vsf_0 \quad \{3\}
\]
\[
V_{\text{Cell2}} = V - V_{\text{Cell1}} \quad \{4\}
\]

Where \(V\) is the volume of the mother cell at division, \(V_{\text{Cell1}}\) and \(V_{\text{Cell2}}\) are the volumes of the mother and the daughter cell respectively. The coefficient \(sf_0\) is the split fraction and is equal to \(0.5 + 0.1R\), where \(R\) is random number drawn from a standard uniform distribution. The randomization introduced at division contributes to desynchronizing the population.

The fixed cylinder geometry leads to a discontinuity at division. That is because the total length of a mother cell is less than the total end-on-end length of its two daughter cells (if total volume
or mass is conserved). The difference is $0.67r_0$. If the total length is conserved, daughter cells overlap and this can create a large force. If the daughter cells are moved out, this can result in overlap with adjacent cells, and also a large force. These large forces can lead to numerical instability. This behavior is controlled by the $f_{DL}$ parameter where, $f_{DL} = 0$ has no overlap and $f_{DL} = 1$ has an overlap to produce constant total length.

Both cells inherit the intracellular concentrations from the mother cell. Quorum sensing and concentrations are described in Section 2.5. This includes the whole Hi concentration record and Hi, AiiA, LuxI mass and concentration, as well as the cell volume before the growth process from the previous time step. Assuming that all concentrations (as Hi, AiiA and LuxI) have the same value in both cells, the mass of these substances (MHi, MAiiA and MLuxI) is computed based on the concentration and the cell volume later in the QS process using division correction coefficient (DCC). DCC is computed as the ratio between the new volume after the division and the old one before division, when there is no division the DCC has value of one.

### 2.4. Shoving And Moving

Cells are not motile, but as they grow they shove each other and also push against the walls of the environment. First all resulting forces and torques from agent-agent interaction are computed, and then all forces and torques from agent – wall interaction are computed. A force and torque occurs when the spherocylinders of the two cells overlap and this is modeled as interaction between virtual viscoelastic spheres, see Fig. 3. The centers of the virtual spheres are at the closest points between the axes of the cells (Volfson at al., 2008). Once the presence of contact between two agents is determined, the resulting force and torque acting on the agents are computed. Then, the sum of all forces acting on each cell is computed and this is used to determine the displacement of the cells.

First each agent is examined to determine if it is in contact with other agents. For this, the model uses virtual spheres with radius $r_0$ (the radius of the spherocylinder). Cells are in contact if the virtual spheres overlap, which occurs when the distance between them is less than or equal to the diameter of the spherocylinder. The overlap is defined as the difference between the diameter and virtual sphere’s centers distance. The force between two virtual spheres is:
\[ f_m = k_f (d_0 - d_1) \] \textcolor{red}{\{5\}}

where \( f_m \) the magnitude of the normal force, \( k_f \) is the spring constant and the overlap is \((d_0 - d_1)\), \( d_1 \) is distance between the centers of the virtual spheres. The force vector for the first agent is found by the following expression:

\[
\overrightarrow{pf_1} = \frac{(c_1 - c_2)}{d_1} f_m,
\]

\textcolor{red}{\{6\}}

where \( c_1 \) and \( c_2 \) are centers of the virtual spheres, and \( \frac{(c_1 - c_2)}{d_1} \) is the unit vector of the force. The force for the second agent forming the “contact couple” is

\[
\overrightarrow{pf_2} = -\overrightarrow{pf_1}
\]

\textcolor{red}{\{7\}}

To compute the torque vector for the first cell:

\[
\overrightarrow{pt_1} = \overrightarrow{pr_1} \times \overrightarrow{pf_1}, \text{ where }
\]

\[
\overrightarrow{pr_1} = 0.5 \overrightarrow{c_1} + 0.5 \overrightarrow{c_2} - pc_1,
\]

\textcolor{red}{\{8\}}

\textcolor{red}{\{9\}}

where \( pc_1 \) is the middle of the axis of the first agent, for the second respectively the torque vector is:

\[
\overrightarrow{pt_2} = \overrightarrow{pr_2} \times \overrightarrow{pf_2}, \text{ where }
\]

\[
\overrightarrow{pr_2} = 0.5 \overrightarrow{c_1} + 0.5 \overrightarrow{c_2} - pc_2,
\]

\textcolor{red}{\{10\}}

\textcolor{red}{\{11\}}

\( pc_2 \) is the middle of the axis of the second agent. The force and torque vectors summed in the state variables of the agents as it follows as \( f_m, apm \) and \( apt \), for both agents in the contact couple. This procedure repeats until the force and torque arising from each contact interaction between agents is described. Then a similar approach is used to compute the agent – wall interaction. Following this, the states variables as \( f_m, apm \) and \( apt \) contain the total force and torque from all agent – agent and agent – wall interactions, which are used to compute
displacements. In the next steps we compute how much each cell should move and/or rotate as result of the shoving. First the mass of the agent is computed, as

\[ m = v \rho, \]

where \( v \) is the volume of the cell, and \( \rho \) is the cell density. The moment of inertia (in x, y, and z direction) is computed and then used further to calculate the displacement and rotation for the agent. After that the new agent coordinates \( \text{apa}, \text{apb}, \text{apc}, \text{apd} \) are computed and updated. For more details of this procedure see Litvin and Fuentes, 2004.

2.5. Quorum Sensing

Quorum sensing is the production of a chemical signal, release, detection and reaction to it. The model simulates genetically engineered cells and the equations governing the process were adopted from Danino et al., 2010. The molecule of acyl-homoserine lactone (AHL) is produced enzymatically by the LuxI synthase, diffuses through the cell membrane and in this way acts as mediator of intercellular coupling in the bacterial population. When the AHL molecules binds intracellularly with the constitutively expressed LuxR receptor, a positive feedback loop is initiated by triggering expression of LuxI. In the same time the LuxR–AHL complex activates a time-delayed negative feedback loop by starting the PluxI-driven production of AiiA, which is a protein that degrades AHL. PluxI-driven expression of a green fluorescent protein (yemGFP) is triggered also by the LuxR–AHL complex. Regular pulses of AHL are produced (observed) due the dynamic interactions of the positive and negative feedback loops. (Danino et al., 2010).
Simultaneous In and Out AHL transport through the membrane of each cell enables individuals to auto synchronize with the others and thus produce coordinated bursts of fluorescence, demonstrated in the experiment.

In the model each cell produces signaling molecule (AHL) also called autoinducer, denoted as Hi for AHL concentration and MHi for mass of AHL. The concentration and the mass of the repressor regulating the production of AHL, produced again from each cell is AiiA and MAiiA. The reaction of agents is expressed by production of substance, responsible for luminescence of the cells denoted as concentration and mass as LuxI and MLuxI. As the signaling molecule is produced in the cell, it permeates the cell membrane and diffuses in the environment in the microfluidic chamber. All cells can sense the autoinducer (by this assessing the number of other agents) and react accordingly. The concentration and the mass of AHL molecule outside cell is denoted as He and MHe. Following equations adapted for individual cells describe these processes:

\[
\frac{dM_{\text{AiiA}_i}}{dt} = \left( C_A \left[ 1 - \left( \sum_{n=1}^{N} V_{\text{cell}_i} \left( \frac{1}{V_{\text{parcel of cell}_i}} \right) \right)^4 \right] \right) P_{\text{cell}_i}(\alpha, \tau) - \frac{\gamma_{\text{AiiA}_i}}{1+f(A_{\text{cell}_i}+\text{LuxI}_{\text{cell}_i})} V_{\text{cell}_i} \tag{13}
\]

\[
\frac{dM_{\text{LuxI}_i}}{dt} = \left( C_I \left[ 1 - \left( \sum_{n=1}^{N} V_{\text{cell}_i} \left( \frac{1}{V_{\text{parcel of cell}_i}} \right) \right)^4 \right] \right) P_{\text{cell}_i}(\alpha, \tau) - \frac{\gamma_{\text{LuxI}_i}}{1+f(A_{\text{cell}_i}+\text{LuxI}_{\text{cell}_i})} V_{\text{cell}_i} \tag{14}
\]

\[
P_{\text{cell}_i}(\alpha, \tau) = \frac{\delta+\sigma \left( h_{\text{cell}_i}(t-\tau) \right)^2}{1+k_1 \left( h_{\text{cell}_i}(t-\tau) \right)^2} \tag{15}
\]
\[
\frac{\partial M_{\text{H,cell}_i}}{\partial t} = \left( \frac{b_{\text{Lux}_i}c_{\text{cell}_i}}{1+k_{\text{Lux}_i}c_{\text{cell}_i}} - \gamma_{\text{Aii}A_{\text{cell}_i}}c_{\text{cell}_i} \right) + D \left( H_{\text{e,parcel of cell}_i} - H_{\text{cell}_i} \right) V_{\text{cell}_i}
\]

\[
\frac{\partial M_{\text{H,parcel}_i}}{\partial t} = -D \sum_{\text{cell}_i} V_{\text{cell}_i} \left( H_{\text{e,parcel}_i} - H_{\text{cell}_i} \right) - \mu_{\text{H, parcel}_i} V_{\text{e, parcel}_i} + D_1 \frac{\partial^2 H_{\text{e, parcel}_i}}{\partial x^2} V_{\text{e, parcel}_i}
\]

The following parameters have values as prescribed in Danino et al., 2010: 

\[C_A, C_I, \gamma_A, \gamma_I, f, d_0, b, g, k, \gamma_H, g, D_2, \delta, \alpha, k_1, \mu, D_1, \text{see Table 25.}\]

The term \[1 - \left( \frac{\sum_{\text{cell}_i=1}^n V_{\text{cell}_i} \frac{1}{V_{\text{cell}_i}^2}}{V_{\text{parcel of cell}_i}^2 d_0} \right)^4 \] is as in (Danino et al., 2010) and describes the slowdown of protein synthesis at high cell density, described as ratio between the volumes of all cells in the given parcel to the total volume of the parcel, e.g. the total volume without the cell volume. When the cell density is high is expected production of proteins to decrease due lower nutrient availability (supply) and increased waste concentration. In equations (11) and (12) the term \[P_{\text{cell}_i}(\alpha, \tau)\] is the Hill function and describes the delayed production of corresponding proteins in dependence of previous AHL concentration in the cell. Quote, (Danino et al., 2010): ‘These delayed reactions mimic the complex cascades of processes (transcription, translation, maturation, etc.) leading to formation of functional proteins.’ Using an Euler explicit scheme and rearranging and correcting with division correction coefficient, (DCC) the equations above have following expression used in the model:

\[
M_{\text{Aii}A_{\text{cell}_i}}^{\text{dt}} = \left( C_A \left[ 1 - \left( \frac{1}{dV_{\text{parcel of cell}_i}} \sum_{\text{cell}_i=1}^n V_{\text{cell}_i} \right)^4 \right] \right) \left( \frac{\delta_{\text{Aii}A_{\text{cell}_i}}}{1 + \gamma_{\text{Lux}_i}c_{\text{cell}_i}} \right) V_{\text{cell}_i}^{\text{dt}} dt + M_{\text{Aii}A_{\text{cell}_i}} DCC_{\text{cell}_i}^{\text{dt}}
\]

\[
M_{\text{Lux}_i}^{\text{dt}} = \left( C_A \left[ 1 - \left( \frac{1}{dV_{\text{parcel of cell}_i}} \sum_{\text{cell}_i=1}^n V_{\text{cell}_i} \right)^4 \right] \right) \left( \frac{\delta_{\text{Lux}_i}c_{\text{cell}_i}}{1 + \gamma_{\text{Lux}_i}c_{\text{cell}_i}} \right) V_{\text{cell}_i}^{\text{dt}} dt + M_{\text{Lux}_i}^{\text{dt}} DCC_{\text{cell}_i}^{\text{dt}}
\]

\[
M_{\text{H}i}^{\text{dt}} = \left( \frac{b_{\text{Lux}_i}c_{\text{cell}_i}}{1+k_{\text{Lux}_i}c_{\text{cell}_i}} - \gamma_{\text{Aii}A_{\text{cell}_i}}c_{\text{cell}_i} \right) V_{\text{cell}_i}^{\text{dt}} + D_2 S_{\text{cell}_i}^i \left( H_{\text{e, parcel of cell}_i} - H_{\text{cell}_i} \right) V_{\text{cell}_i}^{\text{dt}} \right) dt + M_{\text{H}i}^{\text{dt}} DCC_{\text{cell}_i}^{\text{dt}}
\]

\[
M_{\text{H,e, parcel}_i}^{\text{dt}} = -D_2 \sum_{\text{cell}_i} DCC_{\text{cell}_i}^{\text{dt}} S_{\text{cell}_i}^i \left( H_{\text{e, parcel of cell}_i} - H_{\text{cell}_i} \right) - \mu_{\text{H,e, parcel}_i} V_{\text{e, parcel}_i}^{\text{dt}} + \frac{dV_{\text{e, parcel}}^{\text{dt}}}{\partial x^2} \left( H_{\text{e, parcel}_i} - 2H_{\text{e, parcel}_i} + \right) dt + M_{\text{H,e, parcel}_i}^{\text{dt}}
\]
For each cell equations \{18\}, \{19\} and \{20\} executed. The index \textit{cell}_i is equal to the ID of the agent, the index ‘ parcel of cell’ _i denotes the ID of the parcel in which the cell is currently on and \(n^*\) is the number of agents in this parcel. The mass of AHL, AiiA and LuxI is corrected with the division correction coefficient (DCC), which represents the ratio between the new volume of the cell after division and the old volume before division. In this way the concentrations in both cells after division is equalized, as it is expect the concentration in both cells to be the same. \(V^t_{cell_i}\) is the volume of the cell from previous time step and \(V^t_{empty_{parcel_i}}\) is the empty volume of the parcel (without the cell volume) again at previous time step. In equation \{21\} \(\mu\) is decay rate of the external AHL, \(D1\) is diffusion constant of the same. For external AHL periodic boundary conditions are used, (first and last parcel are connected). Diffusion only in x –direction is considered, as in y and z direction there is only one parcel. After all calculations for agents and the parcels are done, the model accounts the dilution of all produced substances in the cells and diffused in the environment (the cells are growing in volume, but the empty part of the parcel (without cells) is shrinking using the following equations:

\[
\begin{align*}
AiiA^t_{cell_i} &= MAiiA^t_{cell_i}/V^t_{cell_i} \quad \{22\} \\
Lux^t_{cell_i} &= MLux^t_{cell_i}/V^t_{cell_i} \quad \{23\} \\
H^t_{cell_i} &= MHi^t_{cell_i}/V^t_{cell_i} \quad \{24\} \\
H^t_{empty_{parcel_i}} &= MH^t_{empty_{parcel_i}}/V^t_{empty_{parcel_i}} \quad \{25\}
\end{align*}
\]

2.6. Aging

The age of the cell poles is tracked in terms of the number of divisions and saved in the variables \textit{aaa} and \textit{aab}. The age does not affect the cell behavior (e.g. reduced growth) in this version of the model.

2.7. Implementation

The model is implemented in MATLAB. The model uses a discrete time step and finite difference. For larger number of agents, the execution time in MATLAB is excessive. For that reason the MATLAB coder for automated translation to C++ language is used. This significantly reduces the model runtime.
The computational sequence is as follows: growth, division, aging, shoving and moving process, and then quorum sensing process. At each print time step the state variables of the agents and the environment are exported to text files. For full list of output files see Table 14.

A visualization routine for the simulation of results was developed using the MATLAB Simulink 3D Animation Toolbox. This routine is described in detail in Section 6.3.
3. MODEL APPLICATION

3.1. Colony Growth

Aging in single cell organisms that reproduce with symmetric division and without a juvenile phase is less obvious than in multicellular organisms or single cell organisms that have asymmetric division. Traditionally it was thought that those organisms do not age and exhibit functional immortality.

Stewart at al., 2005 hypothesized that aging of *E. coli* is related to the inheritance of the old pole in the division process. To explore this hypothesis, they tracked exponentially growing individual cells, with an automated fluorescence microscopy system through up to nine generations of growth and reproduction. They measured physical parameters of each cell over time and tracked the identity of each pole, which allowed them to construct complete lineage of the bacterial colonies. They presented a movie constructed from time lapse images that shows a bacterial monolayer of a microcolony growing on microscope cavity slides, containing 505 cells, see Fig. 5A.

Figure 5. (A) Laboratory experiments (Stewart et al., 2005) and (B) simulation results with ABM of bacteria, time and number of agents are noted on each frame (only for the simulation).

The model was setup to simulate the bacteria colony development from a single cell. The size of the chamber is sufficiently large to ensure that the walls do not constrain the growth of the
colony. The age of the cell poles is tracked in terms of number of divisions. Figure 4 shows a comparison between the frames from the experiment (Stewart at al., 2005) and those generated by the model.

The model colony has a more round shape than observed, which maybe due to a lack of stickiness between agents in the model.

3.2. Quorum Sensing

Development of genetic circuits in living cells, is a leading goal in synthetic biology, because many fundamental physiological processes, such as somitogenesis, cardiac function, respiration, insulin secretion and circadian rhythms are regulated by intercellular coupling mechanisms that lead to synchronized oscillators (Danino et al., 2010).

Danino et al., 2010 constructed a network of genes and proteins in *E. coli* with global intercellular coupling that has the capability of generating synchronized oscillations across a bacterial population. They used microfluidic devices designed for cellular populations at different length scales to investigate the collective synchronization properties along with spatiotemporal waves. In a specially designed microfluidic chip with an extended trapping chamber (2000 μm x 100μm x 0.95μ) a few isolated colonies of *E. coli* began to grow and subsequently merged into a large monolayer, that filled the whole chamber. Cells arranged parallel to each other and perpendicular to the edges of the trap, thus achieving a very tight packing. The highest cell density was reached in the center of the colony and was minimal towards the left moving edge. Therefore the first localized burst of fluorescence propagated from the center outwards and was observed at 100 min. A second burst was also observed near the first location and again began to propagate in both directions of the chamber. This behavior was captured in 460-min image sequence (Movie 3 and 4, Danino et al., 2010) and in space-time diagram, where they plotted the fluorescence intensity as a function of time and distance along the chamber (Fig. 3.b, Danino et al., 2010). To describe quantitatively the mechanisms behind the bulk synchronization and wave propagation they developed a computation model using delayed differential equations for protein and AHL concentrations. Their model consisted of one dimensional array of ‘cells’, each of which was described by the same set of delay-differential equations coupled to common, spatially non-uniform field of extracellular AHL, which was
described by linear diffusion equation with sources and sinks due AHL diffusion through the cell membrane and dilution. The model results presented (Fig. 4.e, Danino et al., 2010) show that, when small AHL perturbation is introduced in the middle of the ‘cell’ array it will initiate waves of LuxI concentrations, which agrees with the experimental results.

To simulate synchronous oscillation observed in the experiment the IBM for bacteria was setup with initial number of agents equal to 20 000 to fill the chamber, see dimensions in table 21. The cell density was uniform (0.72) in each parcel. All cell were perpendicular to the wall, and parallel to each other to achieve the tight packing reached in the experiment. All agents had the same dimensions, total length, \( l_t = 2 \mu m \) and diameter 0.5 \( \mu m \). To alleviate the computational burden and also since the chamber was tightly packed with cells, and there was no space for cell to grow and divide, the growth and division were switched off, and all dimensions of the agents remained constant during the simulation. To introduce some AHL perturbation in the center of the cell colony only the cell with ID 500 (that is located there) was loaded with initial AHL concentration, \( H_i = 400 \mu g/ \mu m^3 \) and all other concentrations, \( A_{iiA} \) and LuxI were equal to zero. The diffusion coefficient in the chamber was set to 4000 \( \mu m^2/min \), the coefficient of AHL external decay rate, \( \mu \) was set to 0.5 \( min^{-1} \). The concentrations \( H_e, A_{iiA}, LuxI \) for each agent and external concentration of AHL of the parcels were outputted for each print time step. To be able to track the burst of fluorescence signal through the chamber the mass of LuxI each parcel was computed by summing the mass of LuxI in each agent in the parcel.

Figure 6 shows space time diagram of mass LuxI for each parcel. Initial burst of LuxI starts around 67 min in the center parcel containing the cell with nonzero initial concentration of AHL. As the time progresses the burst reoccurs with higher intensity at the original location and also moves towards the periphery. This results are similar to the observed behavior in the experiment and model results of Danino at al., 2010. In the lab experiment the traveling waves emerged spontaneously in the middle of the colony and propagated outwards, and at later times partially lost coherence due to inhomogeneity in cell population and intrinsic instability of wave propagation. (Fig. 3b, Danino et al., 2010). In the IBM, bacteria cell density was uniform and constant. The difference between simulation results of IBM for bacteria and model presented by Danino at al, is owing to using different approach and parameter’s values in the models.
Figure 6. Simulation results
4. SUMMARY AND OUTLOOK

An ABM of bacteria was presented, which simulates individual cells and includes growth, division, shoving and quorum sensing. The model was applied to two case studies. The first case study is an experiment that investigated aging in *E. coli*. Individual cells were followed as they grow and reproduce over time through automated time-lapse microscopy. The second case study investigates collective synchronization properties in genetically engineered cells of *E. coli*. These applications demonstrate that the model can produce output at the single-cell level, which can be compared directly to observations.

Future research investigating individual cell properties and their influence on population level behavior may build on this model. The model can be expanded by implementing additional processes. For example, cell death, chemotaxis and other processes. From a computing point of view the model can be integrated with COMSOL Multiphysics models to simulate more holistically diffusion of the extracellular substances, such as AHL. As the number of agents can increase dramatically and causes computational burden, a parallelization will be useful. Separating all model variables from the body of the model by putting them in separate input file will reduce the need to be recompile the model every time, when there is a minor change in the parameters or variables. Implementing an option for restart file in the model will greatly improve its capabilities to run more scenarios in shorter time. Also building an easy to use graphic user interface will make more easily automated execution of large computational experiments aiming to investigate correlation between more parameters and behavior of individual cells and their behavior as a population.
5. REFERENCES


6. SUPPLEMENTARY INFORMATION

6.1. Implementation In MATLAB

The model is implemented in MATLAB and it takes advantage of MATLAB’s efficient computation of matrices. For example, the coordinates of one of the poles (point A) are stored in matrix apa, which has size (dimna x 3), where dimna is the expected maximum number of agents during the simulation. The variables of an agent are located in the matrix row corresponding to the agent’s ID number. For example, for the coordinates of point A for agent with ID Number 5, will be in the apa matrix at row number 5.

6.1.1. Spatial Indexing

One of the most repetitive and computationally burdensome process in the model is shoving, because each agent needs to be checked for potential contact with all other agents. To reduce the number of checks performed, spatial indexing of agents is used. The environment is divided into grids and the check for contact is done only for those agents that are in the vicinity of the current agent (in the adjacent grid parcels). This eliminates unnecessary checks with cells that are too far away to potentially contact the current cell.

Two additional conditions were implemented for those cells in the neighborhood, to further decrease the number of unnecessary checks. The first condition is that the distance between the cell centers should be less or equal to the maximum value of this distance, which happen when the axes of the cells lay on the same line and both cells have their maximum length, (lcd) multiplied by a safety factor, (SF) plus the twice the radius of the spherocylinders, (r0).

The second condition is the shortest distance between the centers of the hemispheres of both agents (min(|A1A2|, |A1B2|, |B1A2|, |B1B2|)) to be less or equal to this distance when it has its maximum value, observed when both cells have their maximum length (lcd), computed as ((SF*lcd/2)² +(2r0)²)⁰·⁵, where SF is safety factor.

The shoving procedure starts with a loop over all agents, and checks each agent with all agents that are in the neighbor grid cells with ID number higher than the checked agent, plus the two other limiting conditions. In case there is contact, the force arising in this interaction is computed
for both agents in the couple, and respectively the variable containing the sum of the forces is updated. Next agent is checked again with all agents in the vicinity until all agents are depleted.

### 6.1.2. Subfunctions

FC4 is the “contact function”, which checks for contact between two agents, by finding the minimum distance between their axes and comparing this distance with twice the radius of the agent and returning the value of this distance and giving 0 or 1 value for contact presence in case the distance is bigger/equal or less than the compared one value.

The function PerpendicularLine4 is a subfunction of the contact function, and it finds the perpendicular from point to a line, by taking the coordinates of the point from which the perpendicular is build and the coordinates of the two points constructing the segment; and returns the coordinates of the point from the perpendicular and belonging to the line, length of the perpendicular, and value 0 or 1 in case the point is outer/inner for the segment.

### 6.1.3. Memory Matrix

The past concentration of AHL used in the Hill function are saved in a Him matrix and fixed index is used. This matrix has $\tau/dt$ number of rows and the number of columns is equal to dimna.

### 6.2. Executing The Model

The model is implemented in MATLAB. As the number of agents increases the model run time becomes long. MATLAB Coder and Microsoft Visual Studio were used to create executable file of the model in C++.

**Prior to using MATLAB Coder for first time, Microsoft Visual Studio or one of the supported compilers ([http://www.mathworks.com/support/compilers/R2013a/index.html](http://www.mathworks.com/support/compilers/R2013a/index.html)) should be installed and then the command ‘mex -setup’ should be executed, see help in MATLAB documentation.**

The creation of a model executable file has the following stages:

#### 6.2.1. Model Setup

1) Open the model (Example: open ABM_2013_05_01_V_05_func.m) and setup for specific run (setup for “tend”, specifying the number of expected max number of agent, initial
coordinates of agents, specific values of some variables, see Tables 12-27 for the values for experiment 1 and experiment 2. Make sure you know the values of the variables “dimna”, “inx_g2”, “SF_Contact”, these values will be used in next steps.

2) Update the variable ‘ModelName’ with the current model name for proper generation of the output files names. Example: Name of the model: ABM_2013_05_01_V_05_func.m, then: ModelName = 'ABM_2013_05_01_V_05_func';

6.2.2. Translation Of MATLAB Script To C++ Code

1) Type in Matlab command window ‘coder’ to open Matlab coder or open it from Matlab interface. Create MATLAB Coder project file with the same name as the name of the model. (Example: Name of the model: ABM_2013_05_01_V_05_func.m ABM_2013_05_01_V_05_func.prj)

2) Input all function that need to be translated into C++ in the coder project, see the inventory list of files in Table 1.

Table 1. Inventory Of Files For Creating Coder Project

<table>
<thead>
<tr>
<th>File name</th>
<th>Note</th>
</tr>
</thead>
<tbody>
<tr>
<td>ABM_Function.m</td>
<td>Model MATLAB Code</td>
</tr>
<tr>
<td>Output_01.m</td>
<td>Function for exporting data in text format, used for output file ‘Step_01’</td>
</tr>
<tr>
<td>Output_02.m</td>
<td>Function for exporting data in text format, used for output file ‘Step_02’</td>
</tr>
<tr>
<td>Output_03.m</td>
<td>Function for exporting data in text format, used for output file ‘Step_03’, ‘Step_09’, ‘Step_10’</td>
</tr>
<tr>
<td>Output_04.m</td>
<td>Function for exporting data in text format, used for output file ‘Step_04’ - ‘Step_08’</td>
</tr>
<tr>
<td>Output_05.m</td>
<td>Function for exporting data in text format, used for output file ‘Step_00’</td>
</tr>
<tr>
<td>FC4.m</td>
<td>Subfunction, the ‘contact function’</td>
</tr>
<tr>
<td>PerpendicularLine4.m</td>
<td>Subfunction of FC4, function for finding a perpendicular from point to a line</td>
</tr>
</tbody>
</table>

3) Define the size and type of each function (see Tables 2-8). Use the values of the variables “dimna”, “inx_g2”, “SF_Contact” set in previous steps.

Table 2 . Input Variables Types And Sizes For Function FC4.m

<table>
<thead>
<tr>
<th>Variable Name</th>
<th>Type</th>
<th>Size</th>
<th>Note</th>
</tr>
</thead>
<tbody>
<tr>
<td>A1</td>
<td>double</td>
<td>1 x 3</td>
<td>Coordinates of point A of the first agent in the contact couple</td>
</tr>
<tr>
<td>B1</td>
<td>double</td>
<td>1 x 3</td>
<td>Coordinates of point B of the first agent in the contact couple</td>
</tr>
<tr>
<td>A2</td>
<td>double</td>
<td>1 x 3</td>
<td>Coordinates of point A of the second agent in the contact couple</td>
</tr>
</tbody>
</table>
B2 double 1 x 3 Coordinates of point B of the second agent in the contact couple
A1B1 double 1 x 3 Coordinates of the vector AB of the first agent in the contact couple
A2B2 double 1 x 3 Coordinates of the vector AB of the second agent in the contact couple
A1A2 double 1 x 3 Coordinates of the vector A1A2
A1B2 double 1 x 3 Coordinates of the vector A1B2
B1A2 double 1 x 3 Coordinates of the vector B1A2
B1B2 double 1 x 3 Coordinates of the vector B1B2
Lsq A1B1 double 1 x 1 Squared length of the vector A1B1
LsqA2B2 double 1 x 1 Squared length of the vector A2B1
LsqA1A2 double 1 x 1 Squared length of the vector A1A2
LsqA1B2 double 1 x 1 Squared length of the vector A1B2
LsqB1A2 double 1 x 1 Squared length of the vector B1A2
LsqB1B2 double 1 x 1 Squared length of the vector B1B2
MinDistance double 1 x 1 \min([LsqA1A2, LsqA1B2, LsqB1A2, LsqB1B2]);

Table 3. Input Variables Types And Sizes For Function Output_01.m Used For Output File ‘Step 01’

<table>
<thead>
<tr>
<th>Variable Name</th>
<th>Type</th>
<th>Size</th>
<th>Note</th>
</tr>
</thead>
<tbody>
<tr>
<td>FileName</td>
<td>char</td>
<td>1 x 71</td>
<td>Name of the output file</td>
</tr>
<tr>
<td>FileNameLength</td>
<td>double</td>
<td>1 x 1</td>
<td>String length of the output file name</td>
</tr>
<tr>
<td>OutputData</td>
<td>double</td>
<td>dimna x 19, EX: 20000 x 19</td>
<td>The data array for output</td>
</tr>
<tr>
<td>OutputSize1</td>
<td>double</td>
<td>1 x 1</td>
<td>Size of the first dimension of the output data array</td>
</tr>
<tr>
<td>OutputSize2</td>
<td>double</td>
<td>1 x 1</td>
<td>Size of the second dimension of the output data array</td>
</tr>
</tbody>
</table>

Table 4. Input Variables Types And Sizes For Function Output_02.m Used For Output File ‘Step 02’

<table>
<thead>
<tr>
<th>Variable Name</th>
<th>Type</th>
<th>Size</th>
<th>Note</th>
</tr>
</thead>
<tbody>
<tr>
<td>FileName</td>
<td>char</td>
<td>1 x 71</td>
<td>Name of the output file</td>
</tr>
<tr>
<td>FileNameLength</td>
<td>double</td>
<td>1 x 1</td>
<td>String length of the output file name</td>
</tr>
<tr>
<td>OutputData</td>
<td>double</td>
<td>(dimna* SF_Contact) x 11, EX: 200 000 x 11</td>
<td>The data array for output</td>
</tr>
<tr>
<td>OutputSize1</td>
<td>double</td>
<td>1 x 1</td>
<td>Size of the first dimension of the output data array</td>
</tr>
<tr>
<td>OutputSize2</td>
<td>double</td>
<td>1 x 1</td>
<td>Size of the second dimension of the output data array</td>
</tr>
</tbody>
</table>

Table 5. Input Variables Types And Sizes For Function Output_03.m Used For Output Files ‘Step 03’, ‘Step 09’, ‘Step 10’ And ‘Step 11’

<table>
<thead>
<tr>
<th>Variable Name</th>
<th>Type</th>
<th>Size</th>
<th>Note</th>
</tr>
</thead>
<tbody>
<tr>
<td>FileName</td>
<td>char</td>
<td>1 x 71</td>
<td>Name of the output file</td>
</tr>
<tr>
<td>FileNameLength</td>
<td>double</td>
<td>1 x 1</td>
<td>String length of the output file name</td>
</tr>
<tr>
<td>OutputData</td>
<td>double</td>
<td>1 x (inx_g2+2), EX: 1 x 202</td>
<td>The data array for output</td>
</tr>
<tr>
<td>OutputSize1</td>
<td>double</td>
<td>1 x 1</td>
<td>Size of the first dimension of the output data array</td>
</tr>
<tr>
<td>OutputSize2</td>
<td>double</td>
<td>1 x 1</td>
<td>Size of the second dimension of the output data array</td>
</tr>
</tbody>
</table>
Table 6. Input Variables Types And Sizes For Function Output_04.m Used For Output Files ‘Step 04’, ‘Step 05’, ‘Step06’, ‘Step07’, ‘Step08’

<table>
<thead>
<tr>
<th>Variable Name</th>
<th>Type</th>
<th>Size</th>
<th>Note</th>
</tr>
</thead>
<tbody>
<tr>
<td>FileName</td>
<td>char</td>
<td>1 x 71</td>
<td>Name of the output file</td>
</tr>
<tr>
<td>FileNameLength</td>
<td>double</td>
<td>1 x 1</td>
<td>String length of the output file name</td>
</tr>
<tr>
<td>OutputData</td>
<td>double</td>
<td>1 x (dimna +2), Ex: 1 x 20002</td>
<td>The data array for output</td>
</tr>
<tr>
<td>OutputSize1</td>
<td>double</td>
<td>1 x 1</td>
<td>Size of the first dimension of the output data array</td>
</tr>
<tr>
<td>OutputSize2</td>
<td>double</td>
<td>1 x 1</td>
<td>Size of the second dimension of the output data array</td>
</tr>
</tbody>
</table>

Table 7. Input Variables Types And Sizes For Function Output_04.m Used For Output Files ‘Step 00’

<table>
<thead>
<tr>
<th>Variable Name</th>
<th>Type</th>
<th>Size</th>
<th>Note</th>
</tr>
</thead>
<tbody>
<tr>
<td>FileName</td>
<td>char</td>
<td>1 x 71</td>
<td>Name of the output file</td>
</tr>
<tr>
<td>FileNameLength</td>
<td>double</td>
<td>1 x 1</td>
<td>String length of the output file name</td>
</tr>
<tr>
<td>OutputData</td>
<td>double</td>
<td>1 x 90</td>
<td>The data array for output</td>
</tr>
<tr>
<td>OutputSize1</td>
<td>double</td>
<td>1 x 1</td>
<td>Size of the first dimension of the output data array</td>
</tr>
<tr>
<td>OutputSize2</td>
<td>double</td>
<td>1 x 1</td>
<td>Size of the second dimension of the output data array</td>
</tr>
</tbody>
</table>

Table 8. Input Variables Types And Sizes For Function PerpendicularLine4.m

<table>
<thead>
<tr>
<th>Variable Name</th>
<th>Type</th>
<th>Size</th>
<th>Note</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>double</td>
<td>1 x 3</td>
<td>Coordinates of point A from the line AB</td>
</tr>
<tr>
<td>C</td>
<td>double</td>
<td>1 x 3</td>
<td>Coordinates of point C from which a perpendicular line is build to the line AB</td>
</tr>
<tr>
<td>AB</td>
<td>double</td>
<td>1 x 3</td>
<td>Coordinates of the vector AB</td>
</tr>
<tr>
<td>AC</td>
<td>double</td>
<td>1 x 3</td>
<td>Coordinates of the vector AC</td>
</tr>
<tr>
<td>LsqAB</td>
<td>double</td>
<td>1 x 1</td>
<td>Squared length of the vector AB</td>
</tr>
</tbody>
</table>

4) Set MATLAB Coder properties in the ‘Build’ tab: for ‘Output type’ choose ‘C/C ++ Executable’ and mark on “generate code only”.

5) “ProjectSettings” adjustments: Open ‘More settings’, then in ‘All Settings’ tab, choose for ‘Language’ C ++; In “Speed” tab mark only “Saturate on integer overflow”, unmark everything else; in “Custom Code ” tab, copy and paste the first lines the functions “myoutput.cpp” and “PrintTimeReferencedVariable.cpp”, followed by semicolon or see below the text:

```c
void PrintTimeReferencedVariable (char * DateOut);
void myoutput(const char* _filename, int fileNameLen, double* data, int n, int m);
```

**Note:** This text can be found also in HelpFile1.txt.

6.2.3. Creation Of Executable File Of ABM For Bacteria With Microsoft Visual Studio

1) Create Microsoft Visual Studio project, (win32 Console application) with the same name as the model and choose proper place for the project. In “Application Settings” tab choose “empty project”, (unmark everything else), press “Finish”.

2) Add the two missing in the generation custom written functions for an output and transformation of the date of execution to a string, namely “myoutput.cpp” and “PrintTimeReferencedVariable.cpp” to the folder (codegen → exe → “Name of the Matlab Coder Project”), where Matlab Coder placed the translated in C++ files.

3) Add Header and Source Files in the project from the folder generated from Matlab Coder. Check the inventory list in Table 9

Table 9. Inventory Of Files After Generated From MATLAB Coder (When Biomechanics ON) And The Two Additional Files Need For Generation Of Executable File In Microsoft Visual Studio

<table>
<thead>
<tr>
<th>File name</th>
<th>Note</th>
</tr>
</thead>
<tbody>
<tr>
<td>ABM_2013_05_01_V_05_func.cpp</td>
<td>Files, automatically generated from MATLAB Coder after translation from MATLAB to C++</td>
</tr>
<tr>
<td>ABM_2013_05_01_V_05_func_data.cpp</td>
<td></td>
</tr>
<tr>
<td>ABM_2013_05_01_V_05_func_emxutil.cpp</td>
<td></td>
</tr>
<tr>
<td>ABM_2013_05_01_V_05_func_initialize.cpp</td>
<td></td>
</tr>
<tr>
<td>ABM_2013_05_01_V_05_func_terminate.cpp</td>
<td></td>
</tr>
<tr>
<td>det.cpp</td>
<td></td>
</tr>
<tr>
<td>FC4.cpp</td>
<td></td>
</tr>
<tr>
<td>Output_01.cpp</td>
<td></td>
</tr>
<tr>
<td>Output_02.cpp</td>
<td></td>
</tr>
<tr>
<td>Output_03.cpp</td>
<td></td>
</tr>
<tr>
<td>Output_04.cpp</td>
<td></td>
</tr>
<tr>
<td>Output_05.cpp</td>
<td></td>
</tr>
<tr>
<td>PerpendicularLine4.cpp</td>
<td></td>
</tr>
<tr>
<td>power.cpp</td>
<td></td>
</tr>
<tr>
<td>rand.cpp</td>
<td></td>
</tr>
<tr>
<td>rdivide.cpp</td>
<td></td>
</tr>
<tr>
<td>ABM_2013_05_01_V_05_func.h</td>
<td></td>
</tr>
</tbody>
</table>
4) Fix the names of the ABM function in the “myoutput.cpp” where is required to be specified the name of the functions to be included. Example: If the name of the function prepared for compile is **ABM_2013_03_01_V_03_func**, the file “myoutput.cpp” should look like this:

```c
#include "ABM_2013_03_01_V_03_func.h"
#include "ABM_2013_03_01_V_03_func_initialize.h"
...
And also in:
int main(){
clock_t start = clock(), diff;
ABM_2013_03_01_V_03_func_initialize();
ABM_2013_03_01_V_03_func();
```

5) Add optional display of the number of the agents and current time during the execution of the ABM for bacteria. Open cpp file containing the model, for example **ABM_2013_07_11_V_02_func.cpp** and put the following: **#include <stdio.h>** in beginning, where all other needed libraries are included. Search in the file for “**>= ntp)** {” and put the following after: **printf("Time is %Lf minutes and the Number of Agents is %Lf \n", t, nat);**

See the example below:
/* % P R O C E S S: OUTPUT */
if (t >= ntp) {
    printf("Time is %lf minutes and the Number of Agents is %lf
", t, nat);
    /* */
/* % age in terms of number of divisions and Hi, AlIa, LuxI Concentration */
for (i = 0; i < 1000; i++) {

Note: The text that is needed to be placed can be found in HelpFile2.txt.

6) Compile the project in release mode.

7) Create the folder with name “matlab-results-from-cpp” on the C disk, to match the path for output folder (C:\matlab-results-from-cpp) declared in myoutput.cpp file or change the path for the desired location.

6.2.4. Run Of Executable ABM For Bacteria

1) Run ABM for bacteria exe file. All output files will be generated in C:\matlab-results-from-cpp.

6.3. Animation Routine

The general procedure for the animation script is as follows: All agents that are expected to be born, during the simulation are created, the same for the virtual spheres if, they will be shown in the visualization. All agents are set to be transparent. The walls of the environment are created also in the beginning. The script opens the first output file with the coordinates of the agents (‘Step-01’) and sets the current time to the time from the input file, opens the second output file, containing the coordinates of the virtual spheres. Further the script reads the first output file row by row and gives the proper position and color of the corresponding agent, and sets its transparency to zero, next it checks if the current time is equal to the time from the input file, if there is no change next agent is depicted, and this procedure continues until the current time and time from the input file differs, this means that all agents for the current moment are depicted, and it is time to record the image frame for the animation. If the virtual spheres are set to be shown and the time read from the second input file (‘Step-02’) is equal to the current time, before recording the frame the same procedure as with the agents is repeated until all virtual spheres for the current time are shown. After recording the frame, all agents and their virtual spheres are set to be transparent, and the current time is updated and the process of reading,
drawing an agent, check for the difference between current time and the time read from the input file repeats until the end of the file is reached.

The figure below represent the procedures in the animation creation process:

Figure 7. Animation Procedure

6.4. Execution Of The Animation

To be able to run the animation Simulink 3D Animation toolbox should be installed.

1) Place all input (from C:\matlab-results-from-cpp) and animation files in one folder. See Table 10 for full inventory list of files.
Table 10. Files Inventory For Animation

<table>
<thead>
<tr>
<th>Input File name / Input 'type'</th>
<th>Description/Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>INPUTFILE 'Step0'</td>
<td>Example Filename: 2013-02-03-09-52-07-Step-00-Vars-From-ABM_2013_02_03_V_02_func.txt</td>
</tr>
<tr>
<td></td>
<td>Output File 'Step0' (see Table 14)</td>
</tr>
<tr>
<td>INPUTFILE ‘Step01’</td>
<td>Example Filename: 2013-02-03-09-52-07-Step-01-From-ABM_2013_02_03_V_02_func.txt</td>
</tr>
<tr>
<td></td>
<td>Output File 'Step01' (see Table 14)</td>
</tr>
<tr>
<td>INPUTFILE ‘Step02’</td>
<td>Only needed, if Virtual Sphere will be displayed. Example Filename: 2013-02-03-09-52-07-Step-02-From-ABM_2013_02_03_V_02_func.txt</td>
</tr>
<tr>
<td></td>
<td>Output File 'Step02' (see Table 14)</td>
</tr>
<tr>
<td>INPUTFILE ‘Step03’</td>
<td>Only needed, if external concentration will be displayed. Example Filename: 2013-02-03-09-52-07-Step-03-He-From-ABM_2013_02_03_V_02_func.txt</td>
</tr>
<tr>
<td></td>
<td>Output File 'Step03' (see Table 14)</td>
</tr>
<tr>
<td>ABM_Animation.m</td>
<td>Main file for creating the visualization</td>
</tr>
<tr>
<td>CellColorsMode.m</td>
<td>Subfunction of the main file, generates matrix that contains cell colors.</td>
</tr>
<tr>
<td>ProtoEnvironment.wrl</td>
<td>Contains environment, where all agents will be created also a definition for the text (the counter showing the current time frame and current number of agents)</td>
</tr>
<tr>
<td>ProtoCylinder.wrl</td>
<td>Definition of the primitive cylinder used to create the body of the agent</td>
</tr>
<tr>
<td>ProtoSphere.wrl</td>
<td>Definition of the primitive sphere used to create the body of the agent</td>
</tr>
<tr>
<td>ProtoWall.wrl</td>
<td>Definition of the primitive wall used to create the walls of the environment</td>
</tr>
</tbody>
</table>

2) Open ABM_Animation.m and set the variables FilenameStep0, FilenameStep1, FilenameStep2 and FilenameStep3 to the names of the corresponding input text files with their extensions. For example: FilenameStep1 = '2012-10-31-23-23-01-Step-01-From-ABM-2012-10-09-V35_func.txt';

3) Choose values for the following variables (parameters) between 0 and 1, which corresponds to ON/OFF mode (see Table 11, Section 6.5):

Recording, Manual, ShowContacts, ShowExternalConcentration, ShowParcelVerticalBorderWalls, CellColorGradient;

4) Choose values for the following variables (parameters):

NumberOfFrames (example values: 1, 2, 4, 24;), ColorMode and ID (example values: from 1 to 6 for the ColorMode, in case ColorMode = 3, choose value for ID from 1 to the maximum number of bacteria) See detailed description in Table 11.

5) Execute the animation script from the MATLAB editor or in command window (type: eval ABM_Animation)
### 6.5. Tables Of Animation Input Files And Variables

#### Table 11. Animation Variables

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>FilenameStep0</td>
<td></td>
<td>Name of input file ‘Step00’</td>
</tr>
<tr>
<td>FilenameStep1</td>
<td></td>
<td>Name of input file ‘Step01’</td>
</tr>
<tr>
<td>FilenameStep2</td>
<td></td>
<td>Name of input file ‘Step02’</td>
</tr>
<tr>
<td>FilenameStep3</td>
<td></td>
<td>Name of input file ‘Step03’</td>
</tr>
<tr>
<td>Recording</td>
<td>0/1</td>
<td>Boolean variable, controls if animation will be recorded, 0 – Off, 1 - On</td>
</tr>
<tr>
<td>NumberOfFrames</td>
<td>1-24</td>
<td>Controls the number of frames, that will be recorded in the animation</td>
</tr>
<tr>
<td>Manual</td>
<td>0/1</td>
<td>If equal to 1, press any key to proceed to the next frame; if equal 0, the animation will run automatically</td>
</tr>
<tr>
<td>WindowSize</td>
<td>[0 0 1920 1080];</td>
<td>Size of the animation window, first two numbers give position of the window, second two numbers give the size of the figure</td>
</tr>
<tr>
<td>ShowContacts</td>
<td>0/1</td>
<td>Boolean variable, controls if the virtual spheres will be shown, 0 - Off, 1 - On</td>
</tr>
<tr>
<td>ShowExternalConcentration</td>
<td>0/1</td>
<td>Boolean variable, controls if the external concentration will be On (1), or Off (0)</td>
</tr>
<tr>
<td>CellColorGradient</td>
<td>0/1</td>
<td>Boolean variable, controls if the cell concentration will be depicted with one color gradient, On (1), or Off (0)</td>
</tr>
<tr>
<td>ColorMode</td>
<td>1-6</td>
<td>Controls the cells colors. Choose from following options in dependence of the variable ‘ColorMode’ and ‘ID’</td>
</tr>
<tr>
<td>ID</td>
<td></td>
<td>The ID of the agent that will be with different color, if ColorMode = 3</td>
</tr>
<tr>
<td>DataStep1</td>
<td></td>
<td>Data imported from input file ‘Step01’</td>
</tr>
<tr>
<td>DataStep0</td>
<td></td>
<td>Data imported from input file ‘Step00’</td>
</tr>
<tr>
<td>MaxNumberOfBacteria</td>
<td></td>
<td>The number of cells that will be created in the animation</td>
</tr>
<tr>
<td>DataStep2</td>
<td></td>
<td>Data imported from input file ‘Step02’</td>
</tr>
<tr>
<td>MaxNumberOfContacts</td>
<td></td>
<td>The number of virtual spheres that will be created in the animation</td>
</tr>
<tr>
<td>Lx</td>
<td></td>
<td>The length of the environment in x direction</td>
</tr>
<tr>
<td>Ly</td>
<td></td>
<td>The length of the environment in y direction</td>
</tr>
<tr>
<td>Lz</td>
<td></td>
<td>The length of the environment in z direction</td>
</tr>
<tr>
<td>Ldx</td>
<td></td>
<td>The size of the parcel in x direction</td>
</tr>
<tr>
<td>ParcelNumber</td>
<td></td>
<td>Number of the parcels in the environment in x direction</td>
</tr>
<tr>
<td>t0</td>
<td></td>
<td>Thickness of the wall</td>
</tr>
<tr>
<td>SHIFTING</td>
<td></td>
<td>A vector that corrects the cells coordinates to put them in the middle of the virtual world, the last number controls the zoom</td>
</tr>
<tr>
<td>DefaultCellColor</td>
<td></td>
<td>Default cell color, that will be changed gradually to correspond of the cell LuxI concentration</td>
</tr>
<tr>
<td>CellColor</td>
<td></td>
<td>Matrix that contains cell colors, generated with the function CellColorsMode.m</td>
</tr>
<tr>
<td>VirtualSpheresColors</td>
<td></td>
<td>Matrix that contains virtual spheres colors, generated with the function CellColorsMode.m</td>
</tr>
<tr>
<td>DefaultParcelColor</td>
<td></td>
<td>Default parcel color, that will be changed gradually to correspond of the external parcel concentration</td>
</tr>
<tr>
<td>VerticalMainWallColor</td>
<td></td>
<td>Color of the main vertical walls</td>
</tr>
<tr>
<td>Parameter Name</td>
<td>Description</td>
<td></td>
</tr>
<tr>
<td>----------------------------------------</td>
<td>-----------------------------------------------------------------------------</td>
<td></td>
</tr>
<tr>
<td>BottomMainWallColor</td>
<td>Default environment color at the bottom</td>
<td></td>
</tr>
<tr>
<td>MainWallsNumber</td>
<td>Number of the main walls</td>
<td></td>
</tr>
<tr>
<td>WallTransparency</td>
<td>Wall transparency, 0 – solid, 1 – transparent;</td>
<td></td>
</tr>
<tr>
<td>WallTransparency2</td>
<td>Wall transparency2, 0 – solid, 1 – transparent;</td>
<td></td>
</tr>
<tr>
<td>ParcelVerticalBorderWallsColor</td>
<td>Color of the main vertical walls, used to show the borders of the parcels</td>
<td></td>
</tr>
<tr>
<td>ParcelBottomBorderWallsColor</td>
<td>Color of the main bottom walls, used to show the external parcel concentration</td>
<td></td>
</tr>
<tr>
<td>Date</td>
<td>Date when the animation is created, used to create animation file name, string, with format 'yyyy-mm-dd-HH-MM-SS'</td>
<td></td>
</tr>
<tr>
<td>FileNameVideoOutput</td>
<td>Name of the video output file</td>
<td></td>
</tr>
<tr>
<td>VirtualWorld</td>
<td>Virtual world object</td>
<td></td>
</tr>
<tr>
<td>Figure</td>
<td>Figure, containing the virtual world object</td>
<td></td>
</tr>
<tr>
<td>VideoObject</td>
<td>Video object, that will contain all frames of the animation</td>
<td></td>
</tr>
<tr>
<td>ProtoSphere</td>
<td>Proto sphere object</td>
<td></td>
</tr>
<tr>
<td>ProtoCylinder</td>
<td>Proto cylinder object</td>
<td></td>
</tr>
<tr>
<td>ProtoWall</td>
<td>Proto wall object</td>
<td></td>
</tr>
<tr>
<td>DefaultCylinder</td>
<td>Axis of a default cylinder, used to compute rotation of cell axis</td>
<td></td>
</tr>
<tr>
<td>Wall(i), i = 1-6</td>
<td>Wall object with number from 1 to 6</td>
<td></td>
</tr>
<tr>
<td>Wall(i).WallTrans, i = 1 – 6</td>
<td>Transparency of the main walls</td>
<td></td>
</tr>
<tr>
<td>Wall(i).WallColor, i = 1 – 6</td>
<td>Color of the main walls</td>
<td></td>
</tr>
<tr>
<td>Wall(i).WallScale, i = 1 – 6</td>
<td>Size of the main walls</td>
<td></td>
</tr>
<tr>
<td>Wall(i).WallPosition, i = 1 – 6</td>
<td>Position of the main walls</td>
<td></td>
</tr>
<tr>
<td>MaxNumberVertWalls</td>
<td>Maximum number of the vertical walls, that define parcels borders</td>
<td></td>
</tr>
<tr>
<td>WallNumber</td>
<td>Current number of the vertical walls</td>
<td></td>
</tr>
<tr>
<td>ParcelVerticalBorderWalls</td>
<td>Parcel vertical border wall object</td>
<td></td>
</tr>
<tr>
<td>ParcelVerticalBorderWalls.WallTrans</td>
<td>Parcel vertical border wall transparency</td>
<td></td>
</tr>
<tr>
<td>ParcelVerticalBorderWalls.WallColor</td>
<td>Parcel vertical border wall color</td>
<td></td>
</tr>
<tr>
<td>ParcelVerticalBorderWalls.WallScale</td>
<td>Parcel vertical border wall size</td>
<td></td>
</tr>
<tr>
<td>ParcelVerticalBorderWalls.WallPosition</td>
<td>Parcel vertical border wall position</td>
<td></td>
</tr>
<tr>
<td>ParcelNumberBottom</td>
<td>Number of the parcel bottom border wall</td>
<td></td>
</tr>
<tr>
<td>ParcelBottomBorderWalls</td>
<td>Parcel bottom border wall object</td>
<td></td>
</tr>
<tr>
<td>ParcelBottomBorderWalls.WallTrans</td>
<td>Parcel bottom border wall transparency</td>
<td></td>
</tr>
<tr>
<td>ParcelBottomBorderWalls.WallScale</td>
<td>Parcel bottom border wall size</td>
<td></td>
</tr>
<tr>
<td>ParcelBottomBorderWalls.WallPosition</td>
<td>Parcel bottom border wall position</td>
<td></td>
</tr>
<tr>
<td>i</td>
<td>Counter</td>
<td></td>
</tr>
<tr>
<td>SphA</td>
<td>Sphere object 1, used to construct the body of the cell</td>
<td></td>
</tr>
<tr>
<td>SphA.SphTrans</td>
<td>Sphere object 1 transparency</td>
<td></td>
</tr>
<tr>
<td>SphA.SphPosition</td>
<td>Sphere object 1 position</td>
<td></td>
</tr>
<tr>
<td>SphA.SphColor</td>
<td>Sphere object 1 color</td>
<td></td>
</tr>
<tr>
<td>SphB</td>
<td>Sphere object 2, used to construct the body of the cell</td>
<td></td>
</tr>
<tr>
<td>SphB.SphTrans</td>
<td>Sphere object 2 transparency</td>
<td></td>
</tr>
<tr>
<td>SphB.SphPosition</td>
<td>Sphere object 2 position</td>
<td></td>
</tr>
<tr>
<td>SphB.SphColor</td>
<td>Sphere object 2 color</td>
<td></td>
</tr>
<tr>
<td>Cylinder</td>
<td>Cylinder object, used to construct the body of the cell</td>
<td></td>
</tr>
<tr>
<td>Cylinder.CylTrans</td>
<td>Cylinder object transparency</td>
<td></td>
</tr>
<tr>
<td>Variable</td>
<td>Description</td>
<td></td>
</tr>
<tr>
<td>----------------------</td>
<td>------------------------------------------------------------------------------</td>
<td></td>
</tr>
<tr>
<td>Cylinder.CylColor</td>
<td>Cylinder object color vector</td>
<td></td>
</tr>
<tr>
<td>Cylinder.CylPosition</td>
<td>Cylinder object position vector</td>
<td></td>
</tr>
<tr>
<td>Cylinder.CylRotation</td>
<td>Cylinder object rotation vector</td>
<td></td>
</tr>
<tr>
<td>Cylinder.CylScale</td>
<td>Cylinder object scale vector</td>
<td></td>
</tr>
<tr>
<td>SphV1</td>
<td>Virtual sphere object 1, used to construct the body of the cell</td>
<td></td>
</tr>
<tr>
<td>SphV1.SphTrans</td>
<td>Virtual sphere object 1 transparency</td>
<td></td>
</tr>
<tr>
<td>SphV2</td>
<td>Virtual sphere object 2, used to construct the body of the cell</td>
<td></td>
</tr>
<tr>
<td>SphV2.SphTrans</td>
<td>Virtual sphere object 2 transparency</td>
<td></td>
</tr>
<tr>
<td>tPrev</td>
<td>Time from the previous frame</td>
<td></td>
</tr>
<tr>
<td>i2</td>
<td>Main counter for the cells</td>
<td></td>
</tr>
<tr>
<td>i3</td>
<td>Main counter for the virtual spheres</td>
<td></td>
</tr>
<tr>
<td>FileID</td>
<td>File ID of input file ‘Step01’</td>
<td></td>
</tr>
<tr>
<td>FileID2</td>
<td>File ID of input file ‘Step02’</td>
<td></td>
</tr>
<tr>
<td>FileID3</td>
<td>File ID of input file ‘Step03’</td>
<td></td>
</tr>
<tr>
<td>DataV</td>
<td>Matrix containing the data for the virtual spheres</td>
<td></td>
</tr>
<tr>
<td>tVirtSph</td>
<td>Current time from input file ‘Step02’</td>
<td></td>
</tr>
<tr>
<td>Data</td>
<td>Matrix, containing the data imported from input file ‘Step01’</td>
<td></td>
</tr>
<tr>
<td>t</td>
<td>Time imported from input file ‘Step01’</td>
<td></td>
</tr>
<tr>
<td>n</td>
<td>ID of the cell imported from input file ‘Step01’</td>
<td></td>
</tr>
<tr>
<td>A</td>
<td>Cell Point A coordinates imported from input file ‘Step01’</td>
<td></td>
</tr>
<tr>
<td>B</td>
<td>Cell Point B coordinates imported from input file ‘Step01’</td>
<td></td>
</tr>
<tr>
<td>AB</td>
<td>Vector AB, containing the axis of the cell</td>
<td></td>
</tr>
<tr>
<td>a1</td>
<td>Coefficient for color gradient mode</td>
<td></td>
</tr>
<tr>
<td>b1</td>
<td>Coefficient for color gradient mode</td>
<td></td>
</tr>
<tr>
<td>IDContact</td>
<td>ID of the cell contact imported from input file ‘Step02’</td>
<td></td>
</tr>
<tr>
<td>SphV1.SphColor</td>
<td>Virtual Sphere Object 1 Color</td>
<td></td>
</tr>
<tr>
<td>SphV2.SphColor</td>
<td>Virtual Sphere Object 2 Color</td>
<td></td>
</tr>
<tr>
<td>SphV1.SphPosition</td>
<td>Virtual Sphere Object 1 Position</td>
<td></td>
</tr>
<tr>
<td>SphV2.SphPosition</td>
<td>Virtual Sphere Object 2 Position</td>
<td></td>
</tr>
<tr>
<td>image_capture</td>
<td>Current image to be saved in the video object</td>
<td></td>
</tr>
<tr>
<td>VirtualWorld.TEXT.string</td>
<td>Text object, used to display time and number of agents in the animation</td>
<td></td>
</tr>
</tbody>
</table>
6.6. Tables Of Global Variables In ABM Of Bacteria

Table 12 Time Constants And Initial Parameters

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Value Exp. 1</th>
<th>Value Exp. 2</th>
<th>Units</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>t</td>
<td>0.001</td>
<td>0.001</td>
<td>min</td>
<td>Current time of the simulation (Initially t = dt)</td>
</tr>
<tr>
<td>dt</td>
<td>0.001</td>
<td>0.001</td>
<td>min</td>
<td>Time step of the simulation</td>
</tr>
<tr>
<td>tend</td>
<td>500</td>
<td>401</td>
<td>min</td>
<td>Time end, time period of the simulation</td>
</tr>
<tr>
<td>dtp</td>
<td>1</td>
<td>1</td>
<td>min</td>
<td>Time print step</td>
</tr>
<tr>
<td>tps</td>
<td>1</td>
<td>1</td>
<td>min</td>
<td>Time print start</td>
</tr>
<tr>
<td>ntp</td>
<td>1</td>
<td>1</td>
<td>min</td>
<td>Next time print</td>
</tr>
<tr>
<td>dimna</td>
<td>20000</td>
<td>20000</td>
<td>-</td>
<td>Dimension number agents , max cell number expected</td>
</tr>
<tr>
<td>nat</td>
<td>1</td>
<td>20000</td>
<td>-</td>
<td>Number agents at current time, current number of agents</td>
</tr>
</tbody>
</table>

Table 13. Output/Input Options

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Value Exp. 1</th>
<th>Value Exp. 2</th>
<th>Units</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>RestartFile</td>
<td>0</td>
<td>0</td>
<td>-</td>
<td>Boolean variable used in the restart file procedure, [ 0- No; 1 – Yes]</td>
</tr>
<tr>
<td>Biomechanics</td>
<td>1</td>
<td>0</td>
<td>-</td>
<td>Boolean variable controls if growth, division and shoving will be ‘on’ in the simulation, [ 0- No; 1 – Yes]</td>
</tr>
<tr>
<td>OutputContacts</td>
<td>1</td>
<td>0</td>
<td>-</td>
<td>Boolean variable used in the contact procedure output , [ 0- No; 1 – Yes]</td>
</tr>
<tr>
<td>OutputVcellOld</td>
<td>0</td>
<td>0</td>
<td>-</td>
<td>Boolean variable controls the output of the cells volume from the previous time step, [ 0- No; 1 – Yes]</td>
</tr>
<tr>
<td>OutputVcellNew</td>
<td>0</td>
<td>0</td>
<td>-</td>
<td>Boolean variable controls the output of the cells volume from the current time step, [ 0- No; 1 – Yes]</td>
</tr>
<tr>
<td>OutputCellDenMem</td>
<td>0</td>
<td>0</td>
<td>-</td>
<td>Boolean variable controls the output of the cell density member for the parcels, [ 0- No; 1 – Yes]</td>
</tr>
<tr>
<td>OutputdCoef</td>
<td>0</td>
<td>0</td>
<td>-</td>
<td>Boolean variable controls the output of the d coefficient for the parcels, [ 0- No; 1 – Yes]</td>
</tr>
<tr>
<td>SF_Contact</td>
<td>10</td>
<td>10</td>
<td>-</td>
<td>Safety factor for the max number of the contacts possible for one agent</td>
</tr>
<tr>
<td>OutputContactMatrix</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>Temporal output contact matrix</td>
</tr>
<tr>
<td>Symbol</td>
<td>Description</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>-----------------</td>
<td>-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>OutputFile'Step00'</td>
<td>Example Filename: 2013-02-03-09-52-07-Step-00-Variables-From-ABM_2013_02_03_V_02_func.txt Contains main variables and parameters of the environment and agents used in the model: [t dt tend dtp tnp dimna nat ln0 lco v0 rho lco lkn np umax SF MaxContactRange1 MaxContactRangeSq1 MaxContactRangeSq2 kf rfx rfy rfz rtx rty rzt X Y T vChamber inx g1 idy g1 idy g1 idz g1 idz g1 SF g1 MaxNat g1 inx g2 idy g2 idy g2 inz g2 idz g2 V1 SF g2 MaXnat g2 Mu D1 D2 Ca Ci GammaA GammaB f tau delta alpha k1 b kCoeff GammaHi g CellDensityCoef d0 OldestRecord HiInitial];</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>OutputFile'Step01'</td>
<td>Example Filename: 2013-02-03-09-52-07-Step-01-From-ABM_2013_02_03_V_02_func.txt Contains agents coordinates, age in terms of divisions for each pole and concentrations of Hi, AiiA and LuxI for each time step. It has the following data structure: [Time IdAgent Ax Ay Az Bx By Bz Cx Cy Cz Dx Dy Dz, aab, Hi AiiA LuxI], where Ax, Ay, Az, Bx, By, Bz, Cx, Cy, Cz, Dx, Dy, Dz, are X, Y, Z coordinates of the points describing the agents, (see Fig. 1). Every print time step the matrix DataRowStep1 is constructed: DataRowStep1 = [tPrint IDPrint apa apb apc apd apa, aab, Hi AiiA LuxI]; But only ‘DataRowStep1(1:nat,:)’ is appended (exported), where nat is the current number of agents.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>OutputFile'Step02'</td>
<td>Example Filename: 2013-02-03-09-52-07-Step-02-From-ABM_2013_02_03_V_02_func.txt The file ‘Step-02’ contains record of coordinates of the virtual spheres and has the following structure: time, Contact number, ID of 1st agent in contact, ID of 2nd agent in contact, coordinates of the first virtual sphere, coordinates of the second virtual sphere and the distance between the virtual spheres. Every print time step it appends to the text file the matrix ‘OutputContactMatrix(1:ContactNumber,:)’, constructed from submatrices and where ‘ContactNumber’ is the the number of contacts at the current time. The submatrix for each contact at the current print step has the following structure: [t ContactNumber i1a i1b C1x C1y C1z C2x C2y C2z distance];</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>OutputFile'Step03'</td>
<td>Example Filename: 2013-02-03-09-52-07-Step-03-He-From-ABM_2013_02_03_V_02_func.txt F external concentration of AHL (He) for each time step for all parcels. Every print time step it appends the matrix : [t nat He];</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>OutputFile'Step04'</td>
<td>Example Filename: 2013-02-03-09-52-07-Step-04-Hi-From-ABM_2013_02_03_V_02_func.txt Contains the internal cell AHL concentration, (Hi) Every print time step it appends the matrix: [t nat Hi];</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>OutputFile'Step05'</td>
<td>Example Filename: 2013-02-03-09-52-07-Step-05-AiiA-From-ABM_2013_02_03_V_02_func.txt Contains the cell AiiA concentration. Every print time step it appends the matrix: [t nat AiiA];</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>OutputFile'Step06'</td>
<td>Example Filename: 2013-02-03-09-52-07-Step-06-LuxI-From-ABM_2013_02_03_V_02_func.txt Contains the cell concentration of LuxI. Every print time step it appends the matrix : [t nat LuxI];</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>OutputFile'Step07'</td>
<td>Example Filename: 2013-02-03-09-52-07-Step-07-VcellOld-From-ABM_2013_02_03_V_02_func.txt Contains the cell volume for the previous time step, (VcellOld) Every print time step it appends the matrix : [t nat VcellOld];</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>OutputFile'Step08'</td>
<td>Example Filename: 2013-02-03-09-52-07-Step-08-VcellNew-From-ABM_2013_02_03_V_02_func.txt Contains the cell volume for the current time step, (VcellNew) Every print time step it appends the matrix : [t nat VcellNew];</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>OutputFile'Step09'</td>
<td>Example Filename: 2013-02-03-09-52-07-Step-09-MLuxIPar-From-ABM_2013_02_03_V_02_func.txt Contains the cell mass of LuxI for each parcel, (sum of MLuxI of all cells in the parcel) Every print time step it appends the matrix: [t nat MLuxIParcel];</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>OutputFile'Step10'</td>
<td>Example Filename: 2013-02-03-09-52-07-Step-10-CellDenMem-From-ABM_2013_02_03_V_02_func.txt Contains ‘Cell Density Member’ for each parcel Every print time step it appends the matrix: [t nat CellDensityMember];</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Example Filename: 2013-02-03-09-52-07-Step-11-dCoef-From-ABM_2013_02_03_V_02_func.txt
Contains d coefficient for each parcel
Every print time step it appends the matrix: $[ t \text{ nat} (dVeOld/(Vt))]$

Table 15 Output Variables

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ModelName</td>
<td>-</td>
<td>Name of the Current Version if the Program, update manually</td>
</tr>
<tr>
<td>Date</td>
<td>-</td>
<td>Date and time, when the simulation is executed as string, format: ‘yyyy-mm-dd-HH-MM-SS’, used for defining the format of the expected variable in the function coder.ceval('PrintTimeReferencedVariable', coder.wref(Date));</td>
</tr>
<tr>
<td>FileName_00</td>
<td>-</td>
<td>Name of output file ‘Step00’</td>
</tr>
<tr>
<td>FileName_01</td>
<td>-</td>
<td>Name of output file ‘Step01’</td>
</tr>
<tr>
<td>FileName_02</td>
<td>-</td>
<td>Name of output file ‘Step02’</td>
</tr>
<tr>
<td>FileName_03</td>
<td>-</td>
<td>Name of output file ‘Step03’</td>
</tr>
<tr>
<td>FileName_04</td>
<td>-</td>
<td>Name of output file ‘Step04’</td>
</tr>
<tr>
<td>FileName_05</td>
<td>-</td>
<td>Name of output file ‘Step05’</td>
</tr>
<tr>
<td>FileName_06</td>
<td>-</td>
<td>Name of output file ‘Step06’</td>
</tr>
<tr>
<td>FileName_07</td>
<td>-</td>
<td>Name of output file ‘Step07’</td>
</tr>
<tr>
<td>FileName_08</td>
<td>-</td>
<td>Name of output file ‘Step08’</td>
</tr>
<tr>
<td>FileName_09</td>
<td>-</td>
<td>Name of output file ‘Step09’</td>
</tr>
<tr>
<td>FileName_10</td>
<td>-</td>
<td>Name of output file ‘Step10’</td>
</tr>
<tr>
<td>FileName_11</td>
<td>-</td>
<td>Name of output file ‘Step11’</td>
</tr>
<tr>
<td>FileName_00_Length</td>
<td>71</td>
<td>Length of the ‘FileName_00’ as a string</td>
</tr>
<tr>
<td>FileName_01_Length</td>
<td>61</td>
<td>Length of the ‘FileName_01’ as a string</td>
</tr>
<tr>
<td>FileName_02_Length</td>
<td>61</td>
<td>Length of the ‘FileName_02’ as a string</td>
</tr>
<tr>
<td>FileName_03_Length</td>
<td>64</td>
<td>Length of the ‘FileName_03’ as a string</td>
</tr>
<tr>
<td>FileName_04_Length</td>
<td>64</td>
<td>Length of the ‘FileName_04’ as a string</td>
</tr>
<tr>
<td>FileName_05_Length</td>
<td>66</td>
<td>Length of the ‘FileName_05’ as a string</td>
</tr>
<tr>
<td>FileName_06_Length</td>
<td>66</td>
<td>Length of the ‘FileName_06’ as a string</td>
</tr>
<tr>
<td>FileName_07_Length</td>
<td>70</td>
<td>Length of the ‘FileName_07’ as a string</td>
</tr>
<tr>
<td>FileName_08_Length</td>
<td>70</td>
<td>Length of the ‘FileName_08’ as a string</td>
</tr>
<tr>
<td>FileName_09_Length</td>
<td>70</td>
<td>Length of the ‘FileName_09’ as a string</td>
</tr>
<tr>
<td>FileName_10_Length</td>
<td>72</td>
<td>Length of the ‘FileName_10’ as a string</td>
</tr>
<tr>
<td>FileName_11_Length</td>
<td>67</td>
<td>Length of the ‘FileName_11’ as a string</td>
</tr>
</tbody>
</table>

Table 16. Output Help Variables

<table>
<thead>
<tr>
<th>Variable</th>
<th>Units</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Print</td>
<td>min</td>
<td>Matrix used in the output process, contains the current time</td>
</tr>
<tr>
<td>IDPrint</td>
<td>-</td>
<td>Matrix used in the output process, contains the ID of all agents</td>
</tr>
</tbody>
</table>
### Table 17. Agent State Variables

<table>
<thead>
<tr>
<th>Variable</th>
<th>Units</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>apa</td>
<td>μm</td>
<td>Coordinates of the first hemisphere center (point A)</td>
</tr>
<tr>
<td>apb</td>
<td>μm</td>
<td>Coordinates of the second hemisphere center (point B)</td>
</tr>
<tr>
<td>apc</td>
<td>μm</td>
<td>Coordinates of the cylinder center (point C)</td>
</tr>
<tr>
<td>apd</td>
<td>μm</td>
<td>Coordinates of the point laying at the equator of the spherocylinder (point D)</td>
</tr>
<tr>
<td>aaa</td>
<td></td>
<td>Number of divisions that the pole with center point A has undergone during the simulation</td>
</tr>
<tr>
<td>aab</td>
<td></td>
<td>Number of divisions that the pole with center point B has undergone during the simulation</td>
</tr>
<tr>
<td>apf</td>
<td>μg μm /min²</td>
<td>Sum of all forces, arising in cell-cell or cell-wall interactions</td>
</tr>
<tr>
<td>asfm</td>
<td>μg μm /min²</td>
<td>Sum of the magnitudes of all forces, arising in cell-cell or cell-wall interactions</td>
</tr>
<tr>
<td>apt</td>
<td>μg μm²/min²</td>
<td>Sum of torque, arising from all cell-cell or cell-wall interactions</td>
</tr>
<tr>
<td>aAB</td>
<td>μm</td>
<td>Vector connecting the centers of the spheres constructing the spherocylinder</td>
</tr>
<tr>
<td>aABsq</td>
<td>μm²</td>
<td>Squared distance between the centers of the hemispheres of the spherocylinder</td>
</tr>
<tr>
<td>VcellOld</td>
<td>μm³</td>
<td>Cell volume from previous computational step</td>
</tr>
<tr>
<td>CellSurfaceOld</td>
<td>μm²</td>
<td>Cell surface from previous computational step</td>
</tr>
<tr>
<td>VcellNew'</td>
<td>μm³</td>
<td>Cell volume the current computational step</td>
</tr>
<tr>
<td>CellSurfaceNew'</td>
<td>μm²</td>
<td>Cell surface at the current computational step</td>
</tr>
<tr>
<td>DivisionCorrection'</td>
<td>μm³</td>
<td>The ratio between the new and the old cell volume, if cell undergoes division, and value equal to one, if there was no division at the current computational step</td>
</tr>
<tr>
<td>MHi</td>
<td>μm³</td>
<td>Intracellular mass of AHL</td>
</tr>
<tr>
<td>dMHi</td>
<td>μm³</td>
<td>Derivative of the intracellular mass of AHL</td>
</tr>
<tr>
<td>Hi</td>
<td>μg/μm³</td>
<td>Concentration of AHL</td>
</tr>
<tr>
<td>MAiiA</td>
<td>μm³</td>
<td>Intracellular mass of AiiA</td>
</tr>
<tr>
<td>dMAiiA</td>
<td>μm³</td>
<td>Derivative of the intracellular mass of AiiA</td>
</tr>
<tr>
<td>AiiA</td>
<td>μg/μm³</td>
<td>Concentration of AiiA</td>
</tr>
<tr>
<td>MLuxI</td>
<td>μm³</td>
<td>Intracellular mass of LuxI</td>
</tr>
<tr>
<td>dMLuxI</td>
<td>μm³</td>
<td>Derivative of the intracellular mass of LuxI</td>
</tr>
<tr>
<td>LuxI</td>
<td>μg/μm³</td>
<td>Concentration of LuxI</td>
</tr>
</tbody>
</table>

### Table 18. Cell Basic Geometric Parameters

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Value</th>
<th>Units</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>r0</td>
<td>0.25</td>
<td>μm</td>
<td>Cell radius</td>
</tr>
<tr>
<td>d0</td>
<td>-</td>
<td>μm</td>
<td>Cell diameter</td>
</tr>
</tbody>
</table>

### Table 19. Cell Growth Parameters Initialization

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Value Exp. 1</th>
<th>Exp. 2</th>
<th>Units</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>lt0</td>
<td>1.2</td>
<td>1.2</td>
<td>μm</td>
<td>Cell minimum total length</td>
</tr>
<tr>
<td>lc0</td>
<td>-</td>
<td>-</td>
<td>μm</td>
<td>Cell minimum cylinder length</td>
</tr>
<tr>
<td>v0</td>
<td>-</td>
<td>-</td>
<td>μm³</td>
<td>Cell minimum volume</td>
</tr>
</tbody>
</table>
rho \(10^{-12}\) \(10^{-12}\) \(\mu g/\mu m^3\) Density of the bacterium, assume homogeneity

vd - - \(\mu m^3\) Cell volume when division occurs

lcd - - \(\mu m\) Cell division cylinder length

ltd - - \(\mu m\) Cell division total length

kmp \(10^{-11}\) \(10^{-11}\) - Growth pressure limitation : half saturation constant

np 10 10 - Growth pressure limitation : exponent

umax 0.0347 0.0347 \(min^{-1}\) Max growth coefficient

SF 1.2 1.2 - Safety Factor

MaxContactRange1 - - \(\mu m\) Max distance between two agent centers

MaxContactRangeSq1 - - \(\mu m^2\) Squared max distance between two agent centers

MaxContactRangeSq2 - - \(\mu m^2\) Squared max minimum distance between two agent segment ends

Table 20. Cell Shoving Parameters Initialization

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Value</th>
<th>Units</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Exp. 1</td>
<td>Exp. 2</td>
<td></td>
</tr>
<tr>
<td>kf</td>
<td>(10^{-7})</td>
<td>(10^{-7})</td>
<td>(\mu g/min^2) Spring constant</td>
</tr>
<tr>
<td>rfx</td>
<td>(10^{-11})</td>
<td>(10^{-11})</td>
<td>- Randomization factor for force x component</td>
</tr>
<tr>
<td>rfy</td>
<td>(10^{-11})</td>
<td>(10^{-11})</td>
<td>- Randomization factor for force y component</td>
</tr>
<tr>
<td>rfz</td>
<td>0</td>
<td>0</td>
<td>- Randomization factor for force z component</td>
</tr>
<tr>
<td>rtx</td>
<td>0</td>
<td>0</td>
<td>- Randomization factor for torque x component</td>
</tr>
<tr>
<td>rty</td>
<td>0</td>
<td>0</td>
<td>- Randomization factor for torque y component</td>
</tr>
<tr>
<td>rtz</td>
<td>(10^{-11})</td>
<td>(10^{-11})</td>
<td>- Randomization factor for torque z component</td>
</tr>
</tbody>
</table>

Table 21. Parameters Of The Environment

<table>
<thead>
<tr>
<th>Variable</th>
<th>Value</th>
<th>Units</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Exp. 1</td>
<td>Exp. 1</td>
<td></td>
</tr>
<tr>
<td>Xt</td>
<td>200</td>
<td>2000</td>
<td>(\mu m) Environment total length in x direction</td>
</tr>
<tr>
<td>Yt</td>
<td>200</td>
<td>100</td>
<td>(\mu m) Environment total length in y direction</td>
</tr>
<tr>
<td>Zt</td>
<td>0.5</td>
<td>0.5</td>
<td>(\mu m) Environment total length in z direction</td>
</tr>
<tr>
<td>Vchamber</td>
<td>-</td>
<td>-</td>
<td>(\mu m^3) Environment total volume</td>
</tr>
</tbody>
</table>

Table 22. Index 1/Grid 1

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Value</th>
<th>Units</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Exp. 1</td>
<td>Exp. 2</td>
<td></td>
</tr>
<tr>
<td>inx_g1</td>
<td>-</td>
<td>-</td>
<td>- Index, number of the &quot;grid 1&quot; boxes, x-direction</td>
</tr>
<tr>
<td>iny_g1</td>
<td>-</td>
<td>-</td>
<td>- Index, number of the &quot;grid 1&quot; boxes, y-direction</td>
</tr>
<tr>
<td>inz_g1</td>
<td>-</td>
<td>-</td>
<td>- Index, number of the &quot;grid 1&quot; boxes, z-direction</td>
</tr>
<tr>
<td>idx_g1</td>
<td>-</td>
<td>(\mu m)</td>
<td>Index, length of the &quot;grid box 1&quot; in x - direction</td>
</tr>
<tr>
<td>idy_g1</td>
<td>-</td>
<td>(\mu m)</td>
<td>Index, length of the &quot;grid box 1&quot; in y - direction</td>
</tr>
<tr>
<td>idz_g1</td>
<td>-</td>
<td>(\mu m)</td>
<td>Index, length of the &quot;grid box 1&quot; in z - direction</td>
</tr>
<tr>
<td>Maxnat_g1</td>
<td>-</td>
<td>-</td>
<td>- Max agent number in 1 &quot;grid box 1&quot;</td>
</tr>
</tbody>
</table>
Table 23. Index 2/Grid 2

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Value</th>
<th>Units</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>(\text{inx}_g2)</td>
<td>20</td>
<td>-</td>
<td>Number of the parcel in x direction</td>
</tr>
<tr>
<td>(\text{iny}_g2)</td>
<td>-</td>
<td>-</td>
<td>Number of the parcel in y direction</td>
</tr>
<tr>
<td>(\text{inz}_g2)</td>
<td>-</td>
<td>-</td>
<td>Number of the parcel in z direction</td>
</tr>
<tr>
<td>(\text{id}_xg2)</td>
<td>-</td>
<td>(\mu m)</td>
<td>Size of the parcel in x direction</td>
</tr>
<tr>
<td>(\text{id}_yg2)</td>
<td>-</td>
<td>(\mu m)</td>
<td>Size of the parcel in y direction</td>
</tr>
<tr>
<td>(\text{id}_zg2)</td>
<td>-</td>
<td>(\mu m)</td>
<td>Size of the parcel in z direction</td>
</tr>
<tr>
<td>(\text{Maxnat}_g2)</td>
<td>-</td>
<td>-</td>
<td>Maximum agent number in one parcel, (grid 2)</td>
</tr>
<tr>
<td>(\text{SF-g2})</td>
<td>2</td>
<td>2</td>
<td>Safety Factor for computing the maximum agent number in one parcel, (grid 2)</td>
</tr>
<tr>
<td>(\text{iid}_g2)</td>
<td>-</td>
<td>-</td>
<td>Index ID Grid1, Index Table matching &quot;Grid 1&quot;, contains agents ID</td>
</tr>
<tr>
<td>(\text{ina}_g2)</td>
<td>-</td>
<td>-</td>
<td>Index Number Agents Index, contains quantity (number) of the agents in each parcel</td>
</tr>
<tr>
<td>(\text{iid}_p_g2)</td>
<td>-</td>
<td>-</td>
<td>Index ID Parcel, containing the Grid Address of the Agents: The number of the parcel in x, y, z direction, on which the agent is on</td>
</tr>
</tbody>
</table>

Table 24. States Variables Of The Parcels

<table>
<thead>
<tr>
<th>Variable</th>
<th>Units</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>(V_t)</td>
<td>(\mu m^3)</td>
<td>Total volume of the parcel, constant</td>
</tr>
<tr>
<td>(V_{e\text{New}})</td>
<td>(\mu m^3)</td>
<td>Empty volume of the parcel (e.g. the volume of the parcel without cells/ agents)</td>
</tr>
<tr>
<td>(dV_{e\text{New}})</td>
<td>(\mu m^3)</td>
<td>Derivatives of the empty parcel volume at the current time step</td>
</tr>
<tr>
<td>(dV_{e\text{Old}})</td>
<td>(\mu m^3)</td>
<td>Derivatives of the empty parcel volume at the previous time step</td>
</tr>
<tr>
<td>(M_{He})</td>
<td>(\mu g)</td>
<td>Mass of external AHL</td>
</tr>
<tr>
<td>(dM_{He})</td>
<td>(\mu g)</td>
<td>Derivative of the mass of external AHL</td>
</tr>
<tr>
<td>(H_e)</td>
<td>(\mu g/\mu m^3)</td>
<td>Concentration of external AHL</td>
</tr>
<tr>
<td>(M_{LuxIParcel})</td>
<td>(\mu g)</td>
<td>The sum of LuxI mass in the cells, for each parcel</td>
</tr>
</tbody>
</table>

Table 25. Quorum Sensing - Parameters

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Value</th>
<th>Units</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>(\mu)</td>
<td>0.5</td>
<td>0.5</td>
<td>1/min</td>
</tr>
<tr>
<td>(D_1)</td>
<td>4000</td>
<td>4000</td>
<td>(\mu m^2/\text{min})</td>
</tr>
<tr>
<td>(D_2)</td>
<td>2.5</td>
<td>2.5</td>
<td>(\mu m^2/\text{min})</td>
</tr>
<tr>
<td>(C_a)</td>
<td>1</td>
<td>1</td>
<td>-</td>
</tr>
<tr>
<td>(\text{Cl})</td>
<td>4</td>
<td>4</td>
<td>-</td>
</tr>
<tr>
<td>GammaA</td>
<td>15</td>
<td>15</td>
<td>1/min</td>
</tr>
<tr>
<td>Gammal</td>
<td>24</td>
<td>24</td>
<td>1/min</td>
</tr>
</tbody>
</table>
Table 26. Memory Matrix

<table>
<thead>
<tr>
<th>Variable</th>
<th>Units</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>dimHim</td>
<td>µm</td>
<td>Dimension of the ‘Hi Memory Record’</td>
</tr>
<tr>
<td>Him</td>
<td>µg/µm³</td>
<td>“Memory record” which consists of records of past AHL concentration for period of time</td>
</tr>
<tr>
<td>OldestRecord</td>
<td>-</td>
<td>Row number of the oldest record in the matrix containing past Hi concentrations</td>
</tr>
</tbody>
</table>

Table 27. Initial Coordinates Cells And Other Initial Conditions

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Value</th>
<th>Units</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Exp. 1</td>
<td>Exp. 2</td>
<td></td>
</tr>
<tr>
<td>Lti</td>
<td>1.2</td>
<td>2</td>
<td>Length total initial of the first cell</td>
</tr>
<tr>
<td>Lci</td>
<td>-</td>
<td>-</td>
<td>Length cylinder initial of the first cell</td>
</tr>
<tr>
<td>pa0</td>
<td>-</td>
<td>-</td>
<td>Initial coordinates of point a</td>
</tr>
<tr>
<td>pb0</td>
<td>-</td>
<td>-</td>
<td>Initial coordinates of point b</td>
</tr>
<tr>
<td>pc0</td>
<td>-</td>
<td>-</td>
<td>Initial coordinates of point c</td>
</tr>
<tr>
<td>pd0</td>
<td>-</td>
<td>-</td>
<td>Initial coordinates of point d</td>
</tr>
<tr>
<td>Vi</td>
<td>-</td>
<td>-</td>
<td>Initial cell volume</td>
</tr>
<tr>
<td>Si</td>
<td>-</td>
<td>-</td>
<td>Initial cell surface</td>
</tr>
<tr>
<td>HiInitial</td>
<td>50</td>
<td>400</td>
<td>Initial AHL internal concentration</td>
</tr>
<tr>
<td>MHi(1,1)</td>
<td>-</td>
<td>-</td>
<td>Computed AHL internal initial mass</td>
</tr>
<tr>
<td>IDofInitialCell</td>
<td>1</td>
<td>500</td>
<td>ID of the cell with initial Hi concentration</td>
</tr>
</tbody>
</table>
### 6.7. Tables Of Local Variables In ABM Of Bacteria

#### Table 28. Process: Growth

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Units</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ia1</td>
<td>-</td>
<td>Local variable, main counter in growth process, identifier of the cell ID</td>
</tr>
<tr>
<td>pa1</td>
<td>µm</td>
<td>Local variable, point A coordinates of the corresponding agent</td>
</tr>
<tr>
<td>pb1</td>
<td>µm</td>
<td>Local variable, point B coordinates of the corresponding agent</td>
</tr>
<tr>
<td>pc1</td>
<td>µm</td>
<td>Local variable, point C coordinates of the corresponding agent</td>
</tr>
<tr>
<td>pd</td>
<td>µm</td>
<td>Local variable, vector AB Coordinates of the corresponding agent</td>
</tr>
<tr>
<td>lc1</td>
<td>µm</td>
<td>Local variable, vector AB length of the corresponding agent</td>
</tr>
<tr>
<td>v1</td>
<td>µm³</td>
<td>Local variable, cell volume of the corresponding agent</td>
</tr>
<tr>
<td>sa1</td>
<td>µm²</td>
<td>Local variable, cell surface area of the corresponding agent</td>
</tr>
<tr>
<td>fin2</td>
<td>µg µm min⁻²</td>
<td>Local variable, force vector of the corresponding agent</td>
</tr>
<tr>
<td>p</td>
<td>µg µm⁻¹ min⁻²</td>
<td>Local variable, pressure exerted on cell surface of the corresponding agent</td>
</tr>
<tr>
<td>u</td>
<td>min⁻¹</td>
<td>Local variable, growth coefficient corrected with pressure limitation function of the corresponding agent</td>
</tr>
<tr>
<td>dv</td>
<td>µm³</td>
<td>Local variable, volume derivative, (volume increase) of the corresponding agent</td>
</tr>
<tr>
<td>v2</td>
<td>µm³</td>
<td>Local variable, the new volume of the corresponding agent</td>
</tr>
<tr>
<td>lc2</td>
<td>µm</td>
<td>Local variable, new cell length (center to center) of the corresponding agent</td>
</tr>
<tr>
<td>pd</td>
<td>µm</td>
<td>Local variable, half of the length of vector AB, AC vector of the corresponding agent</td>
</tr>
<tr>
<td>pe</td>
<td>µm</td>
<td>Local variable, extended AC vector of the corresponding agent</td>
</tr>
<tr>
<td>pa1</td>
<td>µm</td>
<td>Local variable, new coordinate of point A of the corresponding agent</td>
</tr>
<tr>
<td>pd</td>
<td>µm</td>
<td>Local variable, half of the length of vector AB, BC vector of the corresponding agent</td>
</tr>
<tr>
<td>pe</td>
<td>µm</td>
<td>Local variable, extended BC vector of the corresponding agent</td>
</tr>
<tr>
<td>pb1</td>
<td>µm</td>
<td>Local variable, new coordinate of point B of the corresponding agent</td>
</tr>
</tbody>
</table>

#### Table 29. Process: Division

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Units</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ia1</td>
<td>-</td>
<td>Local variable, second counter in division process, identifier of the cell ID, that will be born</td>
</tr>
<tr>
<td>ia2</td>
<td>-</td>
<td>Local variable, main counter in division process, identifier of the cell ID</td>
</tr>
<tr>
<td>v1</td>
<td>µm³</td>
<td>Local variable, cell volume of the corresponding agent</td>
</tr>
<tr>
<td>pa1</td>
<td>µm</td>
<td>Local variable, point A coordinates of the corresponding agent</td>
</tr>
<tr>
<td>pb1</td>
<td>µm</td>
<td>Local variable, point B coordinates of the corresponding agent</td>
</tr>
<tr>
<td>pc1</td>
<td>µm</td>
<td>Local variable, point C coordinates of the corresponding agent</td>
</tr>
<tr>
<td>pd</td>
<td>µm</td>
<td>Local variable, vector AB coordinates of the corresponding agent</td>
</tr>
<tr>
<td>lc1</td>
<td>µm</td>
<td>Local variable, vector AB length of the corresponding agent</td>
</tr>
<tr>
<td>pd1</td>
<td>µm</td>
<td>Local variable, point D coordinates of the corresponding agent</td>
</tr>
<tr>
<td>pg</td>
<td>µm</td>
<td>Local variable, vector CD coordinates of the corresponding agent, (vector perpendicular to AB vector)</td>
</tr>
<tr>
<td>sf0</td>
<td>-</td>
<td>Coefficient preventing the synchronization between the agents growth and division process</td>
</tr>
<tr>
<td>VolCell1</td>
<td>µm³</td>
<td>Local variable, the volume of the mother cell after division</td>
</tr>
<tr>
<td>VolCell2</td>
<td>µm³</td>
<td>Local variable, the volume of the daughter cell after division</td>
</tr>
<tr>
<td>LCell1</td>
<td>µm</td>
<td>Local variable, new cylinder length of the mother cell after division</td>
</tr>
<tr>
<td>Symbol</td>
<td>Value</td>
<td>Units</td>
</tr>
<tr>
<td>----------</td>
<td>-------</td>
<td>-------</td>
</tr>
<tr>
<td>ia1</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>pc1</td>
<td>µm</td>
<td>µm</td>
</tr>
<tr>
<td>iix</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>iiy</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>iiz</td>
<td>-</td>
<td>-</td>
</tr>
</tbody>
</table>

Table 31. Process: Shoving Cell To Cell

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Units</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ContactNumber</td>
<td>-</td>
<td>Local variable, contains number of the contacts in the whole population (sum of all contacts)</td>
</tr>
<tr>
<td>ia1</td>
<td>-</td>
<td>Local variable, main counter in shoving cell to cell process, identifier of the cell ID of the 1-st cell in “contact couple”</td>
</tr>
<tr>
<td>pa1</td>
<td>µm</td>
<td>Local variable, Contains Point A coordinates of the corresponding agent (the 1-st cell in “contact couple”)</td>
</tr>
<tr>
<td>pb1</td>
<td>µm</td>
<td>Local variable, Contains Point B coordinates of the corresponding agent (the 1-st cell in “contact couple”)</td>
</tr>
<tr>
<td>pc1</td>
<td>µm</td>
<td>Local variable, Contains Point C coordinates of the corresponding agent (the 1-st cell in “contact couple”)</td>
</tr>
<tr>
<td>A1B1</td>
<td>µm²</td>
<td>Local variable, contains vector AB coordinates of the corresponding agent (the 1-st cell in “contact couple”)</td>
</tr>
<tr>
<td>LsqA1B1</td>
<td>µm²</td>
<td>Local variable, contains vector AB length squared of the corresponding agent (the 1-st cell in “contact couple”)</td>
</tr>
<tr>
<td>iix</td>
<td>-</td>
<td>Local variable, the number of the box in X – direction, in which the corresponding agent is on</td>
</tr>
<tr>
<td>iiy</td>
<td>-</td>
<td>Local variable, the number of the box in Y – direction, in which the corresponding agent is on</td>
</tr>
<tr>
<td>iiz</td>
<td>-</td>
<td>Local variable, the number of the box in Z – direction, in which the corresponding agent is on</td>
</tr>
<tr>
<td>iijx</td>
<td>-</td>
<td>Local variable, ID of the adjacent boxes to the box where the agent is currently on, in x direction</td>
</tr>
<tr>
<td>iijy</td>
<td>-</td>
<td>Local variable, ID of the adjacent boxes to the box where the agent is currently on, in y direction</td>
</tr>
<tr>
<td>iijz</td>
<td>-</td>
<td>Local variable, ID of the adjacent boxes to the box where the agent is currently on, in z direction</td>
</tr>
<tr>
<td>iia</td>
<td></td>
<td>Local variable</td>
</tr>
<tr>
<td>ia2</td>
<td></td>
<td>Local variable, second counter in shoving cell to cell process, identifier of the cell ID of the 2-nd cell in “contact couple”</td>
</tr>
<tr>
<td>pc2</td>
<td></td>
<td>Local variable, contains point C coordinates of the corresponding agent (2-nd cell in “contact couple”)</td>
</tr>
</tbody>
</table>
A2B2 \( \mu m \) Local variable, vector AB coordinates of the corresponding agent (2-nd cell in “contact couple”)

LsqA2B2 \( \mu m^2 \) Local variable, vector AB length squared of the corresponding agent (2-nd cell in “contact couple”)

ac2c1 \( \mu m \) Local variable, vector C1C2 (between the centers of the cells)

pa2 \( \mu m \) Local variable, point A coordinates of the corresponding agent (2-nd cell in “contact couple”)

pb2 \( \mu m \) Local variable, point B coordinates of the corresponding agent (2-nd cell in “contact couple”)

A1A2 \( \mu m \) Vector, connecting first point of the first cell with the first point of the second cell in the possible contact couple

B1A2 \( \mu m \) Vector, connecting first point of the first cell with the second point of the second cell in the possible contact couple

B1A2 \( \mu m \) Vector, connecting second point of the first cell with the first point of the second cell in the possible contact couple

B1B2 \( \mu m \) Vector, connecting second point of the first cell with the second point of the second cell in the possible contact couple

LsqA1A2 \( \mu m^2 \) Squared length of vector A1A2

LsqB1A2 \( \mu m^2 \) Squared length of vector A1B2

LsqB1A2 \( \mu m^2 \) Squared length of vector B1A2

LsqB1B2 \( \mu m^2 \) Squared length of vector B1B2

MinDistance \( \mu m^2 \) Minimum length between the vectors A1A2, A1B2, B1A2 and B1B2

Contact - Boolean variable for contact existence, 1- contact exists , 0- no contact

ContactNumber - Current number of contacts between all agents

C1 \( \mu m \) Coordinates of the first virtual sphere

C2 \( \mu m \) Coordinates of the second virtual sphere

distance \( \mu m \) Length of the vector connecting virtual sphere centers

fm \( \mu g \mu m \text{ min}^{-2} \) magnitude of the force arising from cell-cell interaction in the first/second cell

pf \( \mu g \mu m \text{ min}^{-2} \) Force vector for first/second cell from local cell-cell interaction in the first/second cell

hr \( \mu m \) Coordinates of the point at middle between virtual sphere centers in the first/second cell

pr \( \mu m \) Vector CA, used to compute torque

pt \( \mu g \mu m^2 \text{ min}^{-2} \) Torque vector for first/second cell, local cell-cell interaction in the first/second cell

DataContact Vector, containing the results of the interaction, used for export. Has the following structure: [t ContactNumber ia1 ia2 C1 C2 distance]

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Units</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ia1</td>
<td>-</td>
<td>Local variable, main counter in shoving cell to cell process, identifier of the cell ID of the 1-st cell in “contact couple”</td>
</tr>
<tr>
<td>pa1</td>
<td>( \mu m )</td>
<td>Local variable, point A coordinates of the corresponding agent</td>
</tr>
<tr>
<td>pb1</td>
<td>( \mu m )</td>
<td>Local variable, contains point B coordinates of the corresponding agent</td>
</tr>
<tr>
<td>pc1</td>
<td>( \mu m )</td>
<td>Local variable, point C coordinates of the corresponding agent</td>
</tr>
<tr>
<td>Walli-PointA Interaction, i = 1 ( \ldots ) 6</td>
<td>d ( \mu m )</td>
<td>Distance between point A to Wall,</td>
</tr>
<tr>
<td></td>
<td>fm ( \mu g \mu m \text{ min}^{-2} )</td>
<td>Magnitude of the force arising from pole A/ wall1 interaction</td>
</tr>
<tr>
<td></td>
<td>pf ( \mu g \mu m \text{ min}^{-2} )</td>
<td>Force vector in interaction between point A and Wall,</td>
</tr>
<tr>
<td></td>
<td>pr ( \mu m )</td>
<td>Vector CA, used to compute torque</td>
</tr>
<tr>
<td></td>
<td>pt ( \mu g \mu m^2 \text{ min}^{-2} )</td>
<td>Torque vector for point A, local (Wall,) interaction</td>
</tr>
<tr>
<td>Walli-PointB Interaction, i = 1 ( \ldots ) 6</td>
<td>d ( \mu m )</td>
<td>Distance between point B to Wall,</td>
</tr>
<tr>
<td>Symbol</td>
<td>Units</td>
<td>Description</td>
</tr>
<tr>
<td>--------</td>
<td>-------</td>
<td>----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>fm</td>
<td>µg µm min⁻²</td>
<td>Magnitude of the force arising from pole B/ wall1 interaction</td>
</tr>
<tr>
<td>pf</td>
<td>µg µm min⁻²</td>
<td>Force vector in interaction between point B and Wall,</td>
</tr>
<tr>
<td>pr</td>
<td>µm</td>
<td>Vector CB, used to compute torque</td>
</tr>
<tr>
<td>pt</td>
<td>µg µm² min⁻²</td>
<td>Torque vector for point B, local (Wall,) interaction</td>
</tr>
</tbody>
</table>

### Table 33. Process: Moving

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Units</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ia1</td>
<td>-</td>
<td>Local variable, main counter in moving process, identifier of the cell ID</td>
</tr>
<tr>
<td>pa1</td>
<td>µm</td>
<td>Local variable, point A coordinates of the corresponding agent</td>
</tr>
<tr>
<td>pb1</td>
<td>µm</td>
<td>Local Variable, point B coordinates of the corresponding agent</td>
</tr>
<tr>
<td>pc1</td>
<td>µm</td>
<td>Local Variable, point C coordinates of the corresponding agent</td>
</tr>
<tr>
<td>pd1</td>
<td>µm</td>
<td>Local variable, point D coordinates of the corresponding agent</td>
</tr>
<tr>
<td>pd</td>
<td>µm</td>
<td>Local Variable, vector AB coordinates of the corresponding agent</td>
</tr>
<tr>
<td>lc1</td>
<td>µm</td>
<td>Local Variable, vector AB Length of the corresponding agent</td>
</tr>
<tr>
<td>lt1</td>
<td>µm</td>
<td>Local variable, total length of the corresponding agent</td>
</tr>
<tr>
<td>v1</td>
<td>µm³</td>
<td>Local variable, cell volume of the corresponding agent</td>
</tr>
<tr>
<td>m1</td>
<td>µg</td>
<td>Mass of the current agent</td>
</tr>
<tr>
<td>ix</td>
<td>µg µm²</td>
<td>X mass moment of inertia</td>
</tr>
<tr>
<td>iy</td>
<td>µg µm²</td>
<td>Y mass moment of inertia</td>
</tr>
<tr>
<td>iz</td>
<td>µg µm²</td>
<td>Z mass moment of inertia</td>
</tr>
<tr>
<td>pf</td>
<td>µg µm min⁻²</td>
<td>Local variable, force vector coordinates of the corresponding agent</td>
</tr>
<tr>
<td>a</td>
<td>µm min⁻²</td>
<td>Local variable, acceleration of the corresponding agent</td>
</tr>
<tr>
<td>d</td>
<td>µm</td>
<td>Local variable, displacement of the corresponding agent</td>
</tr>
<tr>
<td>pa1</td>
<td>µm</td>
<td>Local variable, point A coordinates of the corresponding agent after computed displacement</td>
</tr>
<tr>
<td>pb1</td>
<td>µm</td>
<td>Local variable, point B coordinates of the corresponding agent after computed displacement</td>
</tr>
<tr>
<td>pc1</td>
<td>µm</td>
<td>Local variable, point C coordinates of the corresponding agent after computed displacement</td>
</tr>
<tr>
<td>pd1</td>
<td>µm</td>
<td>Local variable, point D coordinates of the corresponding agent after computed displacement</td>
</tr>
<tr>
<td>pt</td>
<td>µg µm² min⁻²</td>
<td>Local variable, torque vector of the corresponding agent</td>
</tr>
<tr>
<td>tx</td>
<td>-</td>
<td></td>
</tr>
<tr>
<td>ty</td>
<td>-</td>
<td></td>
</tr>
<tr>
<td>tz</td>
<td>-</td>
<td></td>
</tr>
<tr>
<td>T</td>
<td>-</td>
<td></td>
</tr>
<tr>
<td>ptt1</td>
<td>µg µm² min⁻²</td>
<td></td>
</tr>
<tr>
<td>ptt2</td>
<td>µg µm² min⁻²</td>
<td></td>
</tr>
<tr>
<td>ptt3</td>
<td>µg µm² min⁻²</td>
<td></td>
</tr>
<tr>
<td>pat</td>
<td>µm</td>
<td></td>
</tr>
<tr>
<td>pbt</td>
<td>µm</td>
<td></td>
</tr>
<tr>
<td>pdt</td>
<td>µm</td>
<td></td>
</tr>
<tr>
<td>pdq1</td>
<td>min⁻²</td>
<td></td>
</tr>
<tr>
<td>pdq2</td>
<td>min⁻²</td>
<td></td>
</tr>
<tr>
<td>pdq3</td>
<td>min⁻²</td>
<td></td>
</tr>
<tr>
<td>mrX</td>
<td>-</td>
<td></td>
</tr>
<tr>
<td>Symbol</td>
<td>Units</td>
<td>Description</td>
</tr>
<tr>
<td>--------</td>
<td>-------</td>
<td>-------------</td>
</tr>
<tr>
<td>ia1</td>
<td></td>
<td>Local variable, main counter in “grid 2 build subprocess”, identifier of the cell ID</td>
</tr>
<tr>
<td>pc1</td>
<td>µm</td>
<td>Local variable, point C coordinates of the corresponding agent</td>
</tr>
<tr>
<td>iix</td>
<td></td>
<td>Local variable, the number of the box in X – direction, in which the corresponding agent is on</td>
</tr>
<tr>
<td>iiy</td>
<td></td>
<td>Local variable, the number of the box in Y – direction, in which the corresponding agent is on</td>
</tr>
<tr>
<td>iiiz</td>
<td></td>
<td>Local variable, the number of the box in Z – direction, in which the corresponding agent is on</td>
</tr>
</tbody>
</table>

Table 35. Process: Quorum Sensing - AHL Production And Degradation

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Units</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>i</td>
<td>-</td>
<td>Local variable, the number of the parcel in X – direction</td>
</tr>
<tr>
<td>j</td>
<td>-</td>
<td>Local variable, the number of the parcel in Y – direction</td>
</tr>
<tr>
<td>k</td>
<td>-</td>
<td>Local variable, the number of the parcel in Z – direction</td>
</tr>
<tr>
<td>HeCurrentCell</td>
<td>µg/µm³</td>
<td>The concentration of the AHL of the parcel, on which the corresponding agent is currently on</td>
</tr>
<tr>
<td>SummDiffConc</td>
<td>µg/µm³</td>
<td>Sum of differences between the agent intracellular and extracellular AHL concentration</td>
</tr>
<tr>
<td>SumNewVolOfCellsInParcel</td>
<td>µm³</td>
<td>Sum of agents new volume (after growth and division process), that are in the current parcel</td>
</tr>
<tr>
<td>SumOldVolOfCellsInParcel</td>
<td>µm³</td>
<td>Sum of agents OldVolume (before growth and division process), that are in the current parcel</td>
</tr>
<tr>
<td>NumberOfCellsInCurrentParcel</td>
<td>-</td>
<td>Number of the agents, that are in the current parcel</td>
</tr>
<tr>
<td>IDCurrentCell</td>
<td>-</td>
<td>Identifier if the agent</td>
</tr>
<tr>
<td>IDCurrentCell</td>
<td>-</td>
<td>Identifier if the agent</td>
</tr>
<tr>
<td>HiCurrentCell</td>
<td>µg/µm³</td>
<td>The concentration of the AHL of the agent</td>
</tr>
<tr>
<td>IDx</td>
<td>-</td>
<td>X ID of the parcel (grid2), on which the agent is currently on</td>
</tr>
<tr>
<td>IDy</td>
<td>-</td>
<td>Y ID of the parcel (grid2), on which the agent is currently on</td>
</tr>
<tr>
<td>IDz</td>
<td>-</td>
<td>Z ID of the parcel (grid2), on which the agent is currently on</td>
</tr>
<tr>
<td>HeCurrentParcelForThisCel</td>
<td>µg/µm³</td>
<td>The concentration of the AHL of the parcel, on which the corresponding agent is currently on</td>
</tr>
<tr>
<td>HillFunction</td>
<td>µg/µm³</td>
<td>Hill function</td>
</tr>
</tbody>
</table>
### Table 36. Process: Output

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>tPrint</td>
<td>Time at the print</td>
</tr>
<tr>
<td>DataRowStep1</td>
<td>Matrix for export in Output File ‘Step01’</td>
</tr>
<tr>
<td>DataRowCell</td>
<td>Matrix for export in Output File ‘Step04’ - ‘Step08’</td>
</tr>
<tr>
<td>DataRowParcel</td>
<td>Matrix for export in Output File ‘Step03’, ‘Step09’ – ‘Step11’</td>
</tr>
<tr>
<td>DataVariablesValues</td>
<td>Matrix for export in Output File ‘Step00’</td>
</tr>
</tbody>
</table>

### 6.8. Tables Of Variables In Subfunctions

#### Table 37. FC3.m, ‘Contact Function’, Input Variables

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Units</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>A1A2</td>
<td>µm</td>
<td>Vector, connecting first point of the first cell with the first point of the second cell in the possible contact couple</td>
</tr>
<tr>
<td>B1A2</td>
<td>µm</td>
<td>Vector, connecting first point of the first cell with the second point of the second cell in the possible contact couple</td>
</tr>
<tr>
<td>B1A2</td>
<td>µm</td>
<td>Vector, connecting second point of the first cell with the first point of the second cell in the possible contact couple</td>
</tr>
<tr>
<td>B1B2</td>
<td>µm</td>
<td>Vector, connecting second point of the first cell with the second point of the second cell in the possible contact couple</td>
</tr>
<tr>
<td>LsqA1A2</td>
<td>µm²</td>
<td>Squared length of vector A1A2</td>
</tr>
<tr>
<td>LsqB1A2</td>
<td>µm²</td>
<td>Squared length of vector A1B2</td>
</tr>
<tr>
<td>LsqB1A2</td>
<td>µm²</td>
<td>Squared length of vector B1A2</td>
</tr>
<tr>
<td>LsqB1B2</td>
<td>µm²</td>
<td>Squared length of vector B1B2</td>
</tr>
<tr>
<td>MinDistance</td>
<td>µm²</td>
<td>Minimum length between the vectors A1A2, A1B2, B1A2 and B1B2</td>
</tr>
</tbody>
</table>

#### Table 38. FC3.m, ‘Contact Function’, Output Variables

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Units</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Contact</td>
<td>-</td>
<td>Boolean variable for contact existence, 1- contact exists , 0- no contact</td>
</tr>
<tr>
<td>C1</td>
<td>µm</td>
<td>Coordinates of the first virtual sphere</td>
</tr>
<tr>
<td>C2</td>
<td>µm</td>
<td>Coordinates of the second virtual sphere</td>
</tr>
<tr>
<td>distance</td>
<td>µm</td>
<td>Length of the vector connecting virtual sphere centers</td>
</tr>
</tbody>
</table>

#### Table 39. FC3.m, ‘Contact function’, Local Variables

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Units</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>A2A1</td>
<td>µm</td>
<td>Vector equal to A1A2, but with opposite direction</td>
</tr>
<tr>
<td>A2B1</td>
<td>µm</td>
<td>Vector equal to B1A2, but with opposite direction</td>
</tr>
<tr>
<td>MinDistance</td>
<td>µm²</td>
<td>Current minimum length (minimum between the vectors A1A2, A1B2, B1A2 and B1B2 , A1K, B1K, A2K, B2K)</td>
</tr>
<tr>
<td>K</td>
<td>µm</td>
<td>Coordinates of point K, K is point at the heel of the perpendicular from one of the ends of the first agent to the axis of the second, or from one of the second agent to the axis of first agent (see PerpendicularLine3.m – function)</td>
</tr>
<tr>
<td>InnerPoint</td>
<td>-</td>
<td>K can be inner and external point in relation to the segments A1B1 or A2B2, Inner = 1, External = 0; (see PerpendicularLine3.m – function)</td>
</tr>
<tr>
<td>Symbol</td>
<td>Units</td>
<td>Description</td>
</tr>
<tr>
<td>--------</td>
<td>-------</td>
<td>-------------</td>
</tr>
<tr>
<td>A</td>
<td>μm</td>
<td>Coordinate of the point, that constitutes the line to which the perpendicular will be build (coordinate of agent’s Point A)</td>
</tr>
<tr>
<td>C</td>
<td>μm</td>
<td>Coordinate of the point, from which the perpendicular will be build, (coordinate of other agent’s point A or point B)</td>
</tr>
<tr>
<td>AB</td>
<td>μm</td>
<td>The vector, to which the perpendicular will be build, (the agent axis)</td>
</tr>
<tr>
<td>AC</td>
<td>μm</td>
<td>The vector of the perpendicular</td>
</tr>
<tr>
<td>LsqAB</td>
<td>μm²</td>
<td>The squared length of the line to which the perpendicular will be build (squared length of the axis of one of the cells)</td>
</tr>
</tbody>
</table>

**Table 41. PerpendicularLine3, Output Variables**

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Units</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>K</td>
<td>μm</td>
<td>Coordinates of the heel of the perpendicular, build from point C to line AB, CK ⊥ AB</td>
</tr>
<tr>
<td>LsqCK</td>
<td>μm²</td>
<td>Squared length of the perpendicular</td>
</tr>
<tr>
<td>InnerPoint</td>
<td>-</td>
<td>Boolean variable, indicating if point K is Inner or external, values 1/0</td>
</tr>
</tbody>
</table>

**Table 42. PerpendicularLine3, Local Variables**

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Units</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>m</td>
<td>-</td>
<td>Coefficient, equal to the ratio: ( \frac{\text{dot(AC,AB)}}{\text{dot(AB,AB)}} ); used to compute the coordinates of point K</td>
</tr>
<tr>
<td>CK</td>
<td>μm</td>
<td>Vector of the perpendicular, build from point C to AB line</td>
</tr>
</tbody>
</table>

**Table 43. Output_01.m, Output_02.m, Output_03.m, Output_04.m And Output_05.m, Input Variables**

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>FileName</td>
<td>string</td>
<td>Name of the text file, to which the data will be appended</td>
</tr>
<tr>
<td>FileNameLenght</td>
<td>scalar</td>
<td>Length of the filename string</td>
</tr>
<tr>
<td>OutputData</td>
<td>matrix</td>
<td>The matrix that will be exported to text file</td>
</tr>
<tr>
<td>---------------</td>
<td>--------------------</td>
<td>---------------------------------------------</td>
</tr>
<tr>
<td>OutputSize1</td>
<td>scalar</td>
<td>First dimension of the exported matrix</td>
</tr>
<tr>
<td>OutputSize2</td>
<td>scalar</td>
<td>Second dimension of the exported matrix</td>
</tr>
</tbody>
</table>
6.9. ABM Of Bacteria Code

6.9.1. ABM_2013_07_17_V_01_func.m

% Program: IBM/ABM of Bacteria
% Version: 10
% Date: 05/10/2013
% Author: Kameliya Z. Koleva
% Ferdi L. Hellweger
% Department of Civil and Environmental Engineering
% Northeastern University
% kamikoleva@gmail.com
% ferdi@coe.neu.edu

function []= ABM_2013_07_17_V_01_func
%#codegen
%% Matlab Version: to run the model in Matlab uncomment below
coder.inline('never')

%% GLOBAL DEFINITION: TIME CONSTANTS AND INITIAL PARAMETERS

 t = 0.001;
dt = 0.001;
tend = 100;
dtp = 1000*dt;
tps = t;
ntp = tps;
dimna = 1000;
nat = 1;

%% AUXILIARY DEFINITION: OUTPUT/INPUT OPTIONS

% RestartFile  = 0;
% Biomechanics = 1;
% OutputContacts = 1;
% OutputVcellOld = 0;
% OutputVcellNew = 0;
% OutputCellDenMem = 0;
% OutputdCoef = 0;

if OutputContacts ==1
    SF_Contact = 10;
    OutputContactMatrix = zeros(dimna^2*SF_Contact,11);
end

%% AUXILIARY DEFINITION: OUTPUT/INPUT

% Date = datestr(now, 'yyyy-mm-dd-HH-MM-SS' );
% Matlab Version: to run the model in Matlab uncomment above, comment below
Date = '2012-09-16-09-19-20';
coder.ceval('PrintTimeReferencedVariable', coder.wref(Date));

% NOTE: Put the name of the function HERE !!!
ModelName = 'ABM_2013_07_11_V_02_func';

% OutputName Creation and their Length

% OutputName_00 = [Date, '-Step-00-Variables-From-', ModelName,'.txt'];
% OutputName_00_Length = length(OutputName_00);
% OutputName_01 = [Date, '-Step-01-From-', ModelName,'.txt'];
% OutputName_01_Length = length(OutputName_01);

if OutputContacts ==2
    % OutputName_02 = [Date, '-Step-02-From-', ModelName,'.txt'];
    % OutputName_02_length = length(OutputName_02);
end
FILENAME_03 = [Date, '-Step-03-He-From-', ModelName, '.txt'];
FILENAME_03_LENGTH = length(FILENAME_03);
FILENAME_04 = [Date, '-Step-04-Hi-From-', ModelName, '.txt'];
FILENAME_04_LENGTH = length(FILENAME_04);
FILENAME_05 = [Date, '-Step-05-AiiA-From-', ModelName, '.txt'];
FILENAME_05_LENGTH = length(FILENAME_05);
FILENAME_06 = [Date, '-Step-06-LuxI-From-', ModelName, '.txt'];
FILENAME_06_LENGTH = length(FILENAME_06);

if OutputVcellOld ==1
    FILENAME_07 = [Date, '-Step-07-VcellOld-From-', ModelName, '.txt'];
    FILENAME_07_LENGTH = length(FILENAME_07);
end

if OutputVcellNew ==1
    FILENAME_08 = [Date, '-Step-08-VcellNew-From-', ModelName, '.txt'];
    FILENAME_08_LENGTH = length(FILENAME_08);
end

FILENAME_09 = [Date, '-Step-09-MLuxIPar-From-', ModelName, '.txt'];
FILENAME_09_LENGTH = length(FILENAME_09);

if OutputCellDenMem ==1
    FILENAME_10 = [Date, '-Step-10-CellDenMem-From-', ModelName, '.txt'];
    FILENAME_10_LENGTH = length(FILENAME_10);
end

if OutputdCoef ==1
    FILENAME_11 = [Date, '-Step-11-dCoef-From-', ModelName, '.txt'];
    FILENAME_11_LENGTH = length(FILENAME_11);
end

%% PROCESS: OUTPUT
%% OUTPUT variables initiation
 tPrint = zeros(dimna,1);
 IDPrint = zeros(dimna,1);
 IDPrint(1:dimna,1) = (1:dimna)';

%% AUXILIARY DEFINITION: CONTROL THE RANDOM SEED
 run('seed',15); % Old Function, change with the algorithm below, unmark

%% GLOBAL DEFINITION: PRE-ALLOCATION OF AGENT PARAMETERS
 apa = zeros(dimna,3);
 apb = zeros(dimna,3);
 apc = zeros(dimna,3);
 apd = zeros(dimna,3);

 aaa = zeros(dimna,1);
 aab = zeros(dimna,1);

 apf = zeros(dimna,3);
 asfm = zeros(dimna,1);
 apt = zeros(dimna,3);

 aAB = zeros(dimna,3);
 aABL = zeros(dimna,1);
 aABSq = zeros(dimna,1);

 VcellOld = zeros(dimna,1);
 VcellNew = zeros(dimna,1);
 CellSurfaceOld = zeros(dimna,1);
 CellSurfaceNew = zeros(dimna,1);
 DivisionCorrection = ones(dimna,1);

 MHi = zeros(dimna,1);
 dMHi = zeros(dimna,1);
 Hi = zeros(dimna,1);

 MAiiA = zeros(dimna,1);
 dMAiiA = zeros(dimna,1);
 AiiA = zeros(dimna,1);

 MLuxI = zeros(dimna,1);
 dMLuxI = zeros(dimna,1);
 LuxI = zeros(dimna,1);
GLOBAL DEFINITION: CELL BASIC GEOMETRIC PARAMETERS INITIALIZATION

\r0 = 0.25;
\d0 = 2*r0;

GLOBAL DEFINITION: CELL GROWTH PARAMETERS INITIALIZATION

\l0 = 1.2;
\l0 = l0 - 2*r0;
\v0 = \pi*r0^2*\l0 + 4/3*\pi*r0^3;
\rho = 1e-012;
\vd = 2*v0;
\lcd = (vd - 4/3*\pi*r0^3)/(\pi*r0^2);
\ltd = \lcd + 2*r0;

kmp = 1.00E-11;
np = 10;
\umax = \log(2)/5; \%log(2)/15; \%0.036; \%log(2)/20;

GLOBAL DEFINITION: CELL SHOVING PARAMETERS INITIALIZATION

kf = 10^{(-7)};
\rfx = 1.00E-11;
\rfy = 1.00E-11;
\rfz = 0;

\rtx = 0;
\rty = 0;
\rtz = 1.00E-11;

GLOBAL DEFINITION: WALL BASIC GEOMETRIC PARAMETERS INITIALIZATION - BOUNDARY

\xt = 200;
\yt = 200;
\zt = \d0;
\vchamber = \xt*\yt*\zt;

AUXILIARY DEFINITION: INDEX 1/GRID 1 |SHOVING| - INITIALIZATION

\inx_g1 = max(floor(\xt/\ltd),1);
\idx_g1 = \xt/\inx_g1;
\iny_g1 = max(floor(\yt/\ltd),1);
\idy_g1 = \yt/\iny_g1;
\inz_g1 = max(floor(\zt/\ltd),1);
\idz_g1 = \zt/\inz_g1;

SF_g1 = 2;
Maxnat_g1 = ceil(SF_g1*(\idx_g1*\idy_g1*\idz_g1)/v0);
\iid_g1 = zeros(\inx_g1,\iny_g1,\inz_g1,Maxnat_g1);
\ina_g1 = zeros(\inx_g1,\iny_g1,\inz_g1);

AUXILIARY DEFINITION: INDEX 2/GRID 2 |QUORUM SENSING| - INITIALIZATION

\idx_g2 = 20;
\idy_g2 = \yt;
\idz_g2 = \zt;

\vt = \idx_g2 * \idy_g2 * \idz_g2;
\inx_g2 = \xt / \idx_g2;
\iny_g2 = \yt / \idy_g2;
inz\_g2 = \( \frac{zt}{idz\_g2} \);
SF\_g2 = 2;
Maxnat\_g2 = ceil(Vt/v0\*SF\_g2);
ind\_g2 = zeros(inx\_g2,iny\_g2,inz\_g2,Maxnat\_g2);
indp\_g2 = zeros(dimna,3);

%% GLOBAL DEFINITION: QUORUM SENSING, VOLUME - INITIALIZATION

VeNew = zeros(inx\_g2, iny\_g2, inz\_g2);
dVeNew = zeros(inx\_g2, iny\_g2, inz\_g2);
dVeOld = zeros(inx\_g2, iny\_g2, inz\_g2);

%% GLOBAL DEFINITION: QUORUM SENSING - AHL PRODUCTION AND DEGRADATION - INITIALIZATION

MHe = zeros(inx\_g2, iny\_g2, inz\_g2);
dMHe = zeros(inx\_g2, iny\_g2, inz\_g2);
He = zeros(inx\_g2, iny\_g2, inz\_g2);
MLuxIParcel = zeros(inx\_g2, iny\_g2, inz\_g2);

%% GLOBAL DEFINITION: QUORUM SENSING - PARAMETERS INITIALIZATION

Mu = 0.1;
D1 = 200;
D2 = 2.5;
Ca = 3;
Ci = 4;
GammaA = 15;
GammaI = 24;
f = 0.3;
tau = 10;
delta = 10^-3;
alpha = 2500;
k1 = 0.1;
b = 0.06;
KCoef = 1;
GammaH1 = 0.01;
g = 0.01;
CellDensityCoefd0 = 0.88;
CellDensityMember = zeros(inx\_g2, iny\_g2, inz\_g2);

%% GLOBAL DEFINITION: MEMORY MATRIX - INITIALIZATION

dimHim = tau/dt;
Him = zeros(dimHim,dimna);
OldestRecord = 1;

%% AUXILIARY DEFINITION: RESTART FILE INPUT

if RestartFile == 1
% Implement in the next version !!!
else

%% INPUT: INITIAL COORDINATES OF THE 1-st CELL + VOLUME
%% Case1: 1 Initial Cell
Lti = 1.2;
Lci = Lti - 2*r0;
ap0 = [xt/2-idx\_g2/2 yt/2-Lci/2 0.25];
ape0 = [xt/2-idx\_g2/2 yt/2+Lci/2 0.25];
p0 = (ap0 + ape0)*0.5;
pd0 = [xt/2-idx\_g2/2+r0 yt/2 0.25];
V1 = p1 * r0^2 * Lci + 4/3*p1 * r0^3;
S1 = 2 * p1 * r0 * Lci + 4 * p1 * r0^2;
apa(1,1:3) = ap0;
apb(1,1:3) = ape0;
apc(1,1:3) = pc0;
apd(1,1:3) = pd0;

VcellOld(1,1) = V1;
CellSurfaceOld(1,1) = S1;
aABLength(1,1) = Lci;
% Case 2: 20,000 Initial Cells, Full Chamber
% pa0 = zeros(20000,3); % pb0 = zeros(20000,3);
yIncrease = 0.25;
AgLength = 2;
NumberOfAgInRow = Xt/AgLength;
NumberOfAgInColumn = Yt/d0;
for i = 1:NumberOfAgInRow:InitialNumberOfAgents-1
    pa0(i+1:i+NumberOfAgInRow,1) = [0.25:2:Xt]';
    pb0(i+1:i+NumberOfAgInRow,1) = [1.75:2:Xt]';
end
% pa0(:,2) = yIncrease;
pb0(:,2) = yIncrease;
yIncrease = yIncrease + 0.5;
% for i = 0:NumberOfAgInRow:InitialNumberOfAgents-1
%    pa0(i+1:i+NumberOfAgInRow,2) = yIncrease;
%    pb0(i+1:i+NumberOfAgInRow,2) = yIncrease;
%    yIncrease = yIncrease + 0.5;
% end
pa0(:,3) = 0.25;
pb0(:,3) = 0.25;
nat = size(pa0,1);
apa(1:nat,:) = pa0;
apb(1:nat,:) = pb0;
apc(1:nat,:) = 0.5*(apa(1:nat,:)+ pb0(1:nat,:));

% Compute AB Vector Matrix
aAB(1:nat,:) = apb(1:nat,:) - apa(1:nat,:); % (Length of AB vector)^2: [(xAb)^2 + (yAB)^2 + (zAB)^2]; Input parameter in "Contact Function"
aABsq(1:nat,:) = (aAB(1:nat,:) .^2) .* [1 1 1]';
% Compute AB Vector Length Matrix
aABLength(1:nat,:) = aABsq(1:nat,:) .^ 0.5;
% Compute the old cell volume and surface
VcellOld(1:nat,:) = pi*r0^2*aABLength(1:nat,:)+4/3*pi*r0^3;
CellSurfaceOld(1:nat,:) = 2*pi*r0*aABLength(1:nat,:)+4*pi*r0^2;

% Age of the pole/center in terms of number of divisions
aaa(1,1) = 0;
aab(1,1) = 0;
end;

%% INPUT: INITIAL CONDITIONS | QUORUM SENSING |
HiInitial = 50; % Initial AHL Internal Conc
IDofInitialCell = 1; % ID of the Cell with Initial Hi concentration
Hi(IDofInitialCell,:) = HiInitial; % Initial AHL Internal Conc.: 1-st Cell
Hi(1:nat,:) = HiInitial; % All present cells have Initial Hi Concentration

% Table of Initial Variables Values

DataVariablesValues = [ t dt tend dtpt tsp ntp dimna nat ...
    t0 lc0 v0 rho lcd ltd kmp np ...
    umax Sp ... MaxContactRange1 MaxContactRangeEq MaxContactRangeSq2 ...
    kf rfx rfy rfz rtx rty rtz ...
    Xt Yt Zt Xchamber ...
    inx g1 idx g1 iny g1idy g1 inz g1 idz g1 ... SF g1 Maxnat g1 ...
    lnx g2 idx g2 iny g2 idy g2 inz g2 idz g2 Vt 5f g2 Maxnat g2 ...
    Mu G1 G2 ... Ca CI GammaA GammaI f tau delta alpha ...]
Output_05(FileName_00, FileName_00_Length, DataVariablesValues, 1, size(DataVariablesValues,2))
while \( t \leq \text{tend} \)
if Biomechanics == 1;
% PROCESS: GROWTH
for \( i_{a1} = 1: \text{nat} \)
% Initialization
\[ \begin{align*}
\text{pa}_1 &= \text{apa}(i_{a1},1:3); \\
\text{pb}_1 &= \text{apb}(i_{a1},1:3); \\
\text{pc}_1 &= \text{apc}(i_{a1},1:3); \\
\text{lc}_1 &= \text{aABLength}(i_{a1},1); \\
\text{v}_1 &= \text{VcellOld}(i_{a1},1);
\end{align*} \]
% Pressure Limitation begin
\[ \begin{align*}
\text{p} &= \frac{\text{asfm}(i_{a1},1)}{\text{CellSurfaceOld}(i_{a1},1)};
\end{align*} \]
% Growth Coefficient
\[ \begin{align*}
\text{uc} &= \text{umax} \times \frac{\text{kmp}^\text{np}}{(\text{kmp}^\text{np} + \text{p}^\text{np})}; \\
\text{v}_2 &= \text{v}_1 + \text{dv}; \\
\text{lc}_2 &= \frac{(\text{v}_2 - 4/3\pi\text{r}_0^3)/(\pi\text{r}_0^2)}{\text{v}_1};
\end{align*} \]
% New Coordinates after Growth process
\[ \begin{align*}
\text{pd} &= \text{pa}_1 - \text{pc}_1; \\
\text{pe} &= \frac{\text{pd} \times (\text{lc}_2/\text{lc}_1)}{\text{lc}_2}; \\
\text{pa}_1 &= \text{pc}_1 + \text{pe}; \\
\text{pd} &= \frac{\text{pd} \times (\text{lc}_2/\text{lc}_1)}{\text{lc}_2}; \\
\text{pb}_1 &= \text{pc}_1 + \text{pe}; \\
\text{apa}(i_{a1},1:3) &= \text{pa}_1; \\
\text{apb}(i_{a1},1:3) &= \text{pb}_1;
\end{align*} \]
% Save the NEW Volume and NEW Cell Surface
\[ \begin{align*}
\text{VcellNew}(i_{a1},1) &= \text{v}_2; \\
\text{CellSurfaceNew}(i_{a1},1) &= 2\pi\text{r}_0\text{lc}_2 + 4\pi\text{r}_0^2; \\
\text{aABLength}(i_{a1},1) &= \text{lc}_2;
\end{align*} \]
end;

% SUBPROCESS: RESET DivisionCorrection MATRIX
DivisionCorrection = \text{ones(\text{dimna},1)};

% PROCESS: DIVISION
\( i_{a1} = 0; \)
for \( i_{a2} = 1: \text{nat} \)
% Initialization
\[ \begin{align*}
\text{v}_1 &= \text{VcellNew}(i_{a2},1);
\end{align*} \]
% Division REQUIREMENT check
if \( \text{v}_1 \gg \text{vd} \)
\[ \begin{align*}
\text{pa}_1 &= \text{apa}(i_{a2},1:3); \\
\text{pb}_1 &= \text{apb}(i_{a2},1:3); \\
\text{pc}_1 &= \text{apc}(i_{a2},1:3); \\
\text{pd}_1 &= \text{apd}(i_{a2},1:3); \\
\text{lc}_1 &= \text{aABLength}(i_{a2},1); \\
\text{pg} &= \frac{\text{pd}_1 - \text{pc}_1}{\text{pc}_1};
\end{align*} \]
% Give a ID of the New Cell
\( i_{a1} = i_{a1} + 1; \)
% Cell OVERPOPULATION Check
if (\text{nat}-i_{a1})> \text{dimna}
\[ \text{disp}('\text{Too many agents...'}); \]
sf0 = 0.5 + 0.1*rand(1,1);

VolCell1 = v1 * sf0;
VolCell2 = v1 - VolCell1;
LCell1 = (VolCell1 - 4/3*pi * r0^3)/(pi * r0^2);
LCell2 = (VolCell2 - 4/3*pi * r0^3)/(pi * r0^2);
fDL = 1;

% Daughter/Mother New Coordinates
pd = pa1 - pc1;
pb = pb1 - pc1;
pe = pd*((r0 + LCell2 - fDL*r0/3)/(lc1*0.5));
pe = pd*((r0 - fDL*r0/3)/(lc1*0.5));
pe = pd*((r0 + LCell1 - fDL*r0/3)/(lc1*0.5));

% Update Cell pol1/center Division Age
aaa(nat+ia1,:) = aaa(nat+ia1,:) + 1;

% Mother/Daughter New Coordinates
pd = pa1 - pc1;
pb = pb1 - pc1;
pe = pd*((r0 - fDL*r0/3)/(lc1*0.5));

% Update Cell Volume
VcellOld(nat+ia1,1) = VcellOld(ia2,1);
VcellNew(ia2,1) = VolCell1;
VcellNew(nat+ia1,1) = VolCell2;

% Update the Cell Surface
CellSurfaceOld(nat+ia1,1) = CellSurfaceOld(ia2,1);
CellSurfaceNew(ia2,1) = 4/3*pi*r0^2 + 2/r0*VolCell1;
CellSurfaceNew(nat+ia1,1) = 4/3*pi*r0^2 + 2/r0*VolCell2;

% Prepare to EQUALIZE the Concentrations of the Daughter to the Mother
DivisionCorrection(ia2,1) = VolCell1/v1;
DivisionCorrection(nat+ia1,1) = VolCell2/v1;

% Inherit the Hi-History
Him(:,nat+ia1) = Him(:,ia2);

% Update the Cell Number
nat = nat + ia1;

% Update the Cell Age
CellAge(1:nat) = CellAge(1:nat) + dt;
% SUB PROCESSES: INDEX BUILD 1 |SHOVING|

% Reset: Grid 1
iid_g1 = zeros(inx_g1, iny_g1, inz_g1, Maxnat_g1);

ina_g1 = zeros(inx_g1, iny_g1, inz_g1);

% Start Indexing: Grid 1
for iai = 1:nat
    pcl = apc(iai, 1:3);

    % X Direction
    iix = floor(pcl(1)/idx_g1) + 1;
    if pcl(1) < 0 || pcl(1) > Xt
        disp('Grid 1: Out Of Bounds - X');
    end

    % Y Direction
    iiy = floor(pcl(2)/idy_g1) + 1;
    if pcl(2) < 0 || pcl(2) > Yt
        disp('Grid 1: Out Of Bounds - Y');
    end

    % Z Direction
    iiz = floor(pcl(3)/idz_g1) + 1;
    if pcl(3) < 0 || pcl(3) > Zt
        disp('Grid 1: Out Of Bounds - Z');
    end

    % Rebuild Index Grid 1 [Put the cell in the BOX and tell to the Box which Cell it is containing]
    ina_g1(iix, iiy, iiz) = ina_g1(iix, iiy, iiz) + 1;
    iid_g1(iix, iiy, iiz, ina_g1(iix, iiy, iiz)) = iai;
end

% SUB PROCESSES: RANDOMIZATION and RESET |SHOVING|

% Reset: asfm
asfm = zeros(dimna, 1);

% Randomization: asf
apf(1:nat, 1:3) = (rand(nat, 3) - 0.5);
apf(1:nat, 1:3) = [apf(1:nat, 1)*rfx apf(1:nat, 2)*rfy apf(1:nat, 3)*rfz];

% Randomization: apt
apt(1:nat, 1:3) = (rand(nat, 3) - 0.5);
apt(1:nat, 1:3) = [apt(1:nat, 1)*rtx apt(1:nat, 2)*rty apt(1:nat, 3)*rtz];

% SUB PROCESSES: AB AND ABSQUARED COMPUTATION |SHOVING|
% Compute AB Vector Matrix
aAB(1:nat,:) = apb(1:nat,:) - apa(1:nat,:);

% (Length of AB vector)^2: [(xAb)^2 + (yAB)^2 + (zAB)^2]; Input parameter in "Contact Function"
aABsq(1:nat,:) = (aAB(1:nat,:)'.^2)*[1 1 1]';

% Compute AB Vector Length Matrix
aABLength(1:nat,:) = aABsq(1:nat,:).^0.5;

% PROCESSES: SHOVING CELL to CELL
ContactNumber = 0;
for iai = 1:nat-1

    % Initialization
    pa1 = apa(iai, 1:3);
    pb1 = apb(iai, 1:3);
    pcl = apc(iai, 1:3);

    A1B1 = aAB(iai, 1:3);
    LsqA1B1 = aABsq(iai, 1:3).

% Put the X, Y and Z-Coordinate of Point C in the right Box
    iix = floor(pcl(1)/idx_g1) + 1;
    iiy = floor(pcl(2)/idy_g1) + 1;
    iiz = floor(pcl(3)/idz_g1) + 1;
for iix = max(iix - 1, 1) : min(iix + 1, inx_g1)
for iiy = max(iiy - 1, 1) : min(iiy + 1, iny_g1)
for iiz = max(iiz - 1, 1) : min(iiz + 1, inz_g1)
    for iia = 1:ina_g1(ijx, iiy, iiz)
        ia2 = iid_g1(ijx,iiy,iiz,iia);
        if ia2 > ia1
            pc2 = apc(ia2,1:3);
            ac2c1 = pcl1 - pc2;

% Condition Check1 if the second cell is close to form eventually a "Contact"
if ((ac2c1(1))^2 + (ac2c1(2))^2 + (ac2c1(3))^2) <= MaxContactRangeSq1
    pa2 = apa(ia2,1:3);
    pb2 = apb(ia2,1:3);
    A2B2 = aAB(ia2,1:3);
    LsqA2B2 = aABsq(ia2,1);

% Compute the four vectors, connecting each Axis' End of the First Cell to the each Axis' End of the Second Cell
    A1A2 = pa2 - pa1;
    B1A2 = pa2 - pb1;
    B1B2 = pb2 - pb1;
    A1B2 = pb2 - pa1;

% Compute Their Length
    LsqA1A2 = A1A2(1)*A1A2(1) + A1A2(2)*A1A2(2) + A1A2(3)* A1A2(3);
    LsqB1A2 = B1A2(1)*B1A2(1) + B1A2(2)*B1A2(2) + B1A2(3)* B1A2(3);

    MinDistance = min([LsqA1A2, LsqA1B2, LsqB1A2, LsqB1B2]);

% Condition Check2 if the second cell is close to form eventually a "Contact"
if  MinDistance <= MaxContactRangeSq2
    % Call The "CONTACT FUNCTION" to check if the cells are in Contact and if yes,
    % Find distance between the Virtual Spheres, the closest distance between the two axes
    [Contact,C1,C2,distance]=FC4(pa1, pb1, A1B1, LsqA1B1, pa2, pb2, A2B2, LsqA2B2, ...  
      A1A2, B1A2, B1B2, A1B2, LsqA1A2, LsqB1A2, LsqB1B2, ...  
      LsqA1B2, MinDistance);
    if  Contact==1
        ContactNumber = ContactNumber + 1;
        fm = kf * (d0 - distance);
        pf = ((C1 - C2)/distance)*fm;
        apf(iia1,:,:) = apf(iia1,:,:) + pf;
        asfm(iia1,1,:) = asfm(iia1,1,:) + abs(fm);
        hr = (C1 + C2)/2;
        pr = hr - pcl1;
        pt = [(pr(2)*pf(3)-pr(3)*pf(2))... 
              -(pr(1)*pf(3)-pr(3)*pf(1))... 
              +(pr(1)*pf(2)-pr(2)*pf(1))];
        apt(iia1,:) = apt(iia1,:) + pt;

% Second Virtual Sphere
    pf = -pf;
    apf(iia2,:) = apf(iia2,:) + pf;
    asfm(iia2,1,:) = asfm(iia2,1,:) + abs(fm);
    pr = hr - pc2;
    pt = [(pr(2)*pf(3)-pr(3)*pf(2))... 
              -(pr(1)*pf(3)-pr(3)*pf(1))... 
              +(pr(1)*pf(2)-pr(2)*pf(1))];
    apt(iia2,:) = apt(iia2,:) + pt;

% The "contact" is present and it is time to Print, Output the Cent. of the Virtual Spheres
if OutputContacts ==1
    if t >= ntp
        % Prepare the output, by filling the OutputContactMatrix
        DataContact = [t ContactNumber ia1 ia2 c1 c2 distance];
        54
OutputContactMatrix(ContactNumber,:) = DataContact;
end

%% P R O C E S S: OUTPUT
if OutputContacts ==1
  if t >= ntp
    %% Export Contact Cell Parameters: time, Contact number, ID 1st agent, ID 2nd agent, p.C1, p.C2, distance between the virt.spheres
    Output_02(FileName_02, FileName_02_Length, OutputContactMatrix(1:ContactNumber,:), ContactNumber, 11)
  end
end

%% Reset the OutputContactMatrix
OutputContactMatrix = zeros(dimna*SF_Contact,11);

%% P R O C E S S: SHOVING WALL to CELL

for ia1 = 1:nat
  % Initialization
  pa1 = apa(ia1,1:3);
  pb1 = apb(ia1,1:3);
  pc1 = apc(ia1,1:3);

  % Wall1-PointA Interaction (1 x0 a)
  d = pa1(1) - 0;
  if  d < r0
    fm = kf*(r0-d);
    pf = [0 0 0];
    pf(1) = fm;
    apf(ia1,1:3) = apf(ia1,1:3) + pf;
    asfm(ia1) = asfm(ia1) + abs(fm);
    pr = pa1 - pc1;
    pt = [(pr(2)*pf(3)-pr(3)*pf(2))...
      -(pr(1)*pf(3)-pr(3)*pf(1))...
      +(pr(1)*pf(2)-pr(2)*pf(1))];
    apt(ia1,1:3) = apt(ia1,1:3) + pt;
  end

  % Wall1-PointB Interaction (2 x0 b)
  d = pb1(1) - 0;
  if  d < r0
    fm = kf*(r0-d);
    pf = [0 0 0];
    pf(1) = fm;
    apf(ia1,1:3) = apf(ia1,1:3) + pf;
    asfm(ia1) = asfm(ia1) + abs(fm);
    pr = pb1 - pc1;
    pt = [(pr(2)*pf(3)-pr(3)*pf(2))...
      -(pr(1)*pf(3)-pr(3)*pf(1))...
      +(pr(1)*pf(2)-pr(2)*pf(1))];
    apt(ia1,1:3) = apt(ia1,1:3) + pt;
  end

  % Wall2-PointA Interaction (3 Xt a)
  d = xt - pa1(1);
  if  d < r0
    fm = -kf*(r0-d);
    pf = [0 0 0];
    pf(1) = fm;
    apf(ia1,1:3) = apf(ia1,1:3) + pf;
    asfm(ia1) = asfm(ia1) + abs(fm);
    pr = pa1 - pc1;
    pt = [(pr(2)*pf(3)-pr(3)*pf(2))...
      -(pr(1)*pf(3)-pr(3)*pf(1))...
      +(pr(1)*pf(2)-pr(2)*pf(1))];
    apt(ia1,1:3) = apt(ia1,1:3) + pt;
  end

  % Wall2-PointB Interaction (4 Xt b)
  d = xt - pb1(1);
  if  d < r0
    fm = -kf*(r0-d);
    pf = [0 0 0];
    pf(1) = fm;
    apf(ia1,1:3) = apf(ia1,1:3) + pf;
    asfm(ia1) = asfm(ia1) + abs(fm);
    pr = pb1 - pc1;
    pt = [(pr(2)*pf(3)-pr(3)*pf(2))...
      -(pr(1)*pf(3)-pr(3)*pf(1))...
      +(pr(1)*pf(2)-pr(2)*pf(1))];
    apt(ia1,1:3) = apt(ia1,1:3) + pt;
  end
\[-(pr(1)*pf(3)-pr(3)*pf(1))\]...
\[+(pr(1)*pf(2)-pr(2)*pf(1))\];
apt(la1,1:3) = apt(la1,1:3) + pt;
end

% Wall 3-Point A Interaction (5 y0 a)
d = pa1(2) - 0;
if d < r0
fm = kf*(r0-d);
pf = [0 0 0];
apf(la1,1:3) = apf(la1,1:3) + pf;
asfm(la1) = asfm(la1) + abs(fm);
pr = pal - pcl1;
pt = [(pr(2)*pf(3)-pr(3)*pf(2))... 
-(pr(1)*pf(3)-pr(1))*pf(1))... 
-(pr(1)*pf(2)-pr(2))*pf(1)];
apt(la1,1:3) = apt(la1,1:3) + pt;
end

% Wall 3-Point B Interaction (6 y0 b)
d = pb1(2) - 0;
if d < r0
fm = kf*(r0-d);
pf = [0 0 0];
apf(la1,1:3) = apf(la1,1:3) + pf;
asfm(la1) = asfm(la1) + abs(fm);
pr = pb1 - pc1;
pt = [(pr(2)*pf(3)-pr(3)*pf(2))... 
-(pr(1)*pf(3)-pr(1))*pf(1))... 
-(pr(1)*pf(2)-pr(2))*pf(1)];
apt(la1,1:3) = apt(la1,1:3) + pt;
end

% Wall 4-Point A Interaction (7 yt a)
d = yt - pa1(2);
if d < r0
fm = -kf*(r0-d);
pf = [0 0 0];
apf(la1,1:3) = apf(la1,1:3) + pf;
asfm(la1) = asfm(la1) + abs(fm);
pr = pal - pcl1;
pt = [(pr(2)*pf(3)-pr(3)*pf(2))... 
-(pr(1)*pf(3)-pr(1))*pf(1))... 
-(pr(1)*pf(2)-pr(2))*pf(1)];
apt(la1,1:3) = apt(la1,1:3) + pt;
end

% Wall 4-Point B Interaction (8 yt b)
d = yt - pb1(2);
if d < r0
fm = -kf*(r0-d);
pf = [0 0 0];
apf(la1,1:3) = apf(la1,1:3) + pf;
asfm(la1) = asfm(la1) + abs(fm);
pr = pb1 - pc1;
pt = [(pr(2)*pf(3)-pr(3)*pf(2))... 
-(pr(1)*pf(3)-pr(1))*pf(1))... 
-(pr(1)*pf(2)-pr(2))*pf(1)];
apt(la1,1:3) = apt(la1,1:3) + pt;
end

% Wall 5-Point A Interaction (9 z0 a)
d = pa1(3) - 0;
if d < r0
fm = kf*(r0-d);
pf = [0 0 0];
apf(la1,1:3) = apf(la1,1:3) + pf;
asfm(la1) = asfm(la1) + abs(fm);
pr = pal - pcl1;
pt = [(pr(2)*pf(3)-pr(3)*pf(2))... 
-(pr(1)*pf(3)-pr(1))*pf(1))... 
-(pr(1)*pf(2)-pr(2))*pf(1)];
apt(la1,1:3) = apt(la1,1:3) + pt;
end

% Wall 5-Point B Interaction (10 z0 b)
d = pb1(3) - 0;
if d < r0
fm = kf*(r0-d);
pf = [0 0 0];
apf(la1,1:3) = apf(la1,1:3) + pf;
asfm(la1) = asfm(la1) + abs(fm);
pr = pb1 - pc1;
pt = [(pr(2) * pf(3) - pr(3) * pf(2)) ...
     - (pr(1) * pf(3) - pr(3) * pf(1)) ...
     (pr(1) * pf(2) - pr(2) * pf(1))];
apt(ia1,1:3) = apt(ia1,1:3) + pt;
end

% Wall6-PointA Interaction (11 Zt a)
d = X - p(3);
if d < r0
  fm = -k*(r0 - d);
  pf = [0 0 0];
  apf(ia1,1:3) = apf(ia1,1:3) + pf;
  asfm(ia1) = asfm(ia1) + abs(fm);
  pr = p - p(3);
  pt = [(pr(2) * pf(3) - pr(3) * pf(2)) ...
        - (pr(1) * pf(3) - pr(3) * pf(1)) ...
        (pr(1) * pf(2) - pr(2) * pf(1))];
  apt(ia1,1:3) = apt(ia1,1:3) + pt;
end

% Wall6-PointB Interaction (12 Zt b)
d = X - p(3);
if d < r0
  fm = -k*(r0 - d);
  pf = [0 0 0];
  apf(ia1,1:3) = apf(ia1,1:3) + pf;
  asfm(ia1) = asfm(ia1) + abs(fm);
  pr = p - p(3);
  pt = [(pr(2) * pf(3) - pr(3) * pf(2)) ...
        - (pr(1) * pf(3) - pr(3) * pf(1)) ...
        (pr(1) * pf(2) - pr(2) * pf(1))];
  apt(ia1,1:3) = apt(ia1,1:3) + pt;
end

%% PROCESSES: MOVING
for i = 1:nat
  % Initialization
  pa = ap(ia1,1:3);
  pb = apb(ia1,1:3);
  pc = apc(ia1,1:3);
  pd = apd(ia1,1:3);
  lc = aABLength(ia1,1);
  lt = lc + 2*r;
  v = VcellNew(ia1,1);
  m = v * rho;
  % Mass moment of inertia  % I = [ix iy iz];
  ix = pi * rho * (1/2 * r^4 * lt - 7/15 * r^5);
  iy = pi * rho * (1/12 * r^2 * lt^3 - 1/6 * r^3 * lt^2 + 5/12 * r^4 * lt - 3/10 * r^5);
  iz = iy;
  % Compute the displacement
  pf = apf(ia1,1:3);
  a = pf/m;
  d = a * dt^2 * 0.5;
  % Update the Coordinates with the shifting
  pa = pa + d;
  pb = pb + d;
  pc = pc + d;
  pd = pd + d;
  % The torque Initialization
  pt = apt(ia1,1:3);
  % Rotate torque vector
  tx = (pb - pa) / lc;
  ty = (pd - pc) / r;
  tz = [(tx^2)*ty(3) - tx(3)*ty(2)]...
       - (tx(1)*ty(3) - tx(3)*ty(1))...
       (tx(1)*ty(2) - tx(2)*ty(1));
  T = [tx; ty; tz];
\[ pt_{1} = tx(1) \times pt(1) + tx(2) \times pt(2) + tx(3) \times pt(3); \]
\[ pt_{2} = ty(1) \times pt(1) + ty(2) \times pt(2) + ty(3) \times pt(3); \]
\[ pt_{3} = tz(1) \times pt(1) + tz(2) \times pt(2) + tz(3) \times pt(3); \]

**% Rotated cell points**
\[ pat = [-l1*0.5 0 0]; \]
\[ pbt = [ l1*0.5 0 0]; \]
\[ pdt = [ 0 r0 0]; \]

**% Change in angle**
\[ pdq1 = pt_{1} / ix \times dt^2 / 2; \]
\[ pdq2 = pt_{2} / iy \times dt^2 / 2; \]
\[ pdq3 = pt_{3} / iz \times dt^2 / 2; \]

**% Construct rotation matrix**
\[
\begin{bmatrix}
1 & 0 & 0 \\
0 & \cos(pdq1) & -\sin(pdq1) \\
0 & \sin(pdq1) & \cos(pdq1)
\end{bmatrix};
\]
\[
\begin{bmatrix}
\cos(pdq2) & 0 & \sin(pdq2) \\
0 & 1 & 0 \\
-\sin(pdq2) & 0 & \cos(pdq2)
\end{bmatrix};
\]
\[
\begin{bmatrix}
\cos(pdq3) & -\sin(pdq3) & 0 \\
\sin(pdq3) & \cos(pdq3) & 0 \\
0 & 0 & 1
\end{bmatrix};
\]
\[ mr1 = mrx \times mry; \]
\[ mr2 = mr1 \times mrz; \]

**% Rotate cell points**
\[ pat2 = pat \times mr2T; \]
\[ pbt2 = pbt \times mr2T; \]
\[ pdt2 = pdt \times mr2T; \]

**% Rotate back**
\[ pat = pat2 \times T; \]
\[ pbt = pbt2 \times T; \]
\[ pdt = pdt2 \times T; \]

**% Translate points**
\[ pa1 = pat + pc1; \]
\[ pb1 = pbt + pc1; \]
\[ pd1 = pdt + pc1; \]

**% Update agent**
\[ apa(:,1:3) = pa1; \]
\[ apb(:,1:3) = pb1; \]
\[ apc(:,1:3) = (pa1 + pb1)*0.5; \]
\[ apd(:,1:3) = pd1; \]

end

%% SUBPROCESSES: aAB, aABLength, VcellNew and CellSurfaceNew COMPUTATION
\[ aAB(1:nat,:) = apb(1:nat,:) - apa(1:nat,:); \]
\[ aABLength(1:nat,:) = ((aAB(1:nat,:).^2)*[1 1 1]).^0.5; \]
\[ VcellNew(1:nat,:) = \frac{4}{3}\pi r0^3 + \pi r0^2 aABLength(1:nat,:); \]
\[ CellSurfaceNew(1:nat,:) = 2\pi r0 aABLength(1:nat,:) + 4\pi r0^2; \]

%% SUBPROCESSES: INDEX BUILD 2 |QUORUM SENSING |

**% RESET**
\[ iid_g2 = zeros(inx_g2, iny_g2, inz_g2, Maxnat_g2); \]
\[ ina_g2 = zeros(inx_g2, iny_g2, inz_g2); \]
\[ idp_g2 = zeros(dimna,3); \]

**% INDEXING 2 START**
\[ for iai = 1:nat \\
    pcl = apc(iai,:); \\

**% X Direction**
\[ ix = floor(pcl(1)/idx_g2) + 1; \]
\[ iidp_g2(iai,1) = ix; \]
\[ if pcl(1) < 0 || pcl(1) > Xt \]
** disp('Grid 2: Out Of Bounds - X!'); \]
end

**% Y Direction**
\[ iy = floor(pcl(2)/idy_g2) + 1; \]
\[ iidp_g2(iai,2) = iy; \]
\[ if pcl(2) < 0 || pcl(2) > Yt \]
% disp('Grid 2: Out Of Bounds - Y!');
end

% Z Direction
iiz = floor(pc1(3)/idz_g2) + 1;
iiidp_g2(ial,3) = iiz;
if pc1(3)< 0 || pc1(3)> Zt
% disp('Grid 2: Out Of Bounds - Z!');
end

% Put the cell in the BOX and tell to the Box which Cell it is containing
ina_g2(iix, iiy, iiz) = ina_g2(iix, iiy, iiz) + 1;
iiid_g2(iix, iiy, iiz, ina_g2(iix, iiy, iiz)) = ial;
end

%% PROCESS: QUORUM SENSING - AHL PRODUCTION AND DEGRADATION

%%% Reset: the MASS AHL, AiiA, LuxI Derivative
dMHe(:,:, :) = 0;
dMAiiA(:,:, :) = 0;
dMLuxI(:,:, :) = 0;

%%% ExtraCellular MASS AHL (MHe)
for i = 1:inx_g2
    for j = 1:iny_g2
        for k = 1:inz_g2
            % Reset: Local Variables
            HeCurrentCell = He(i,j,k);
            SummDiffConc = 0;
            SumNewVolOfCellsInParcel = 0;
            SumOldVolOfCellsInParcel = 0;
            NumberOfCellsInCurrentParcel = ina_g2(i,j,k);
            for ii = 1:NumberOfCellsInCurrentParcel
                IDCurrentCell = iid_g2(i,j,k,ii);
                HiCurrentCell = Hi(IDCurrentCell,1);
                VOldCurrentCell  =  VcellOld(IDCurrentCell,1);
                SurfaceOfCurrentCell =  CellSurfaceOld(IDCurrentCell,1);
                SummDiffConc = SummDiffConc + ...
                    DivisionCorection(IDCurrentCell,1)*D2*...
                    SurfaceOfCurrentCell^2*(HeCurrentCell - HiCurrentCell);
                VNewCurrentCell  = VcellNew(IDCurrentCell,1);
                dMHe(i,j,k)   = - SummDiffConc - Mu*He(i,j,k)*(Vt - SumOldVolOfCellsInParcel);
            end
        end
    end
end

% Include the diffusion Part in AHL External Mass Computation
% Row: Only 1-st
for i = 1
    for j = 1:iny_g2
        for k = 1:inz_g2
            dMHe(i,j,k) = dMHe(i,j,k) + ...
                D1*idx_g2*idy_g2*idz_g2/idx_g2^2.*(He(end,j,k)-2*He(i,j,k)+He(i+1,j,k));
        end
    end
end

% Row: from 2-st to end-1
for i = 2:inx_g2-1
    for j = 1:iny_g2
        for k = 1:inz_g2
            dMHe(i,j,k) = dMHe(i,j,k) + ...
                D1*idx_g2*idy_g2*idz_g2/idx_g2^2.*(He(i-1,j,k)-2*He(i,j,k)+He(i+1,j,k));
        end
    end
end

% Include the diffusion Part in AiiA External Mass Computation
% Row: Only 1-st
for i = 1
    for j = 1:iny_g2
        for k = 1:inz_g2
            dMAiiA(i,j,k) = dMAiiA(i,j,k) + ...
                D1*idx_g2*idy_g2*idz_g2/idx_g2^2.*(AiiA(end,j,k)-2*AiiA(i,j,k)+AiiA(i+1,j,k));
        end
    end
end

% Row: from 2-st to end-1
for i = 2:inx_g2-1
    for j = 1:iny_g2
        for k = 1:inz_g2
            dMAiiA(i,j,k) = dMAiiA(i,j,k) + ...
                D1*idx_g2*idy_g2*idz_g2/idx_g2^2.*(AiiA(i-1,j,k)-2*AiiA(i,j,k)+AiiA(i+1,j,k));
        end
    end
end

% Include the diffusion Part in LuxI External Mass Computation
% Row: Only 1-st
for i = 1
    for j = 1:iny_g2
        for k = 1:inz_g2
            dMLuxI(i,j,k) = dMLuxI(i,j,k) + ...
                D1*idx_g2*idy_g2*idz_g2/idx_g2^2.*(LuxI(end,j,k)-2*LuxI(i,j,k)+LuxI(i+1,j,k));
        end
    end
end

% Row: from 2-st to end-1
for i = 2:inx_g2-1
    for j = 1:iny_g2
        for k = 1:inz_g2
            dMLuxI(i,j,k) = dMLuxI(i,j,k) + ...
                D1*idx_g2*idy_g2*idz_g2/idx_g2^2.*(LuxI(i-1,j,k)-2*LuxI(i,j,k)+LuxI(i+1,j,k));
        end
    end
end
for i = inx_g2
  for j = 1:iny_g2
    for k = 1:inz_g2
      dMHe(i,j,k) = dMHe(i,j,k) + ...
        D1*idx_g2*idy_g2*idz_g2/idx_g2^2.*(He(i-1,j,k)-2*He(i,j,k)+He(1,j,k));
    end
  end
end

CellDensityMember = 1-(dVeOld./(Vt)/CellDensityCoefd0).^4;

for IDCurrentCell = 1:nat
  HiCurrentCell  = Hi(IDCurrentCell,1);
  IDx = iidp_g2(IDCurrentCell,1);
  IDy = iidp_g2(IDCurrentCell,2);
  IDz = iidp_g2(IDCurrentCell,3);
  HeCurrentParcelForThisCel = He(IDx,IDy,IDz);
  dMHi(IDCurrentCell,1) =  D2*CellSurfaceOld(IDCurrentCell,1)*...
    (HeCurrentParcelForThisCel - HiCurrentCell);
end

% DEGRADE BEGIN
  if t >= tau
    HillFunction = (delta + alpha*(Him(OldestRecord,IDCurrentCell))^2)/...
      (1 + k1 / (Him(OldestRecord,IDCurrentCell))^2);
    dMAiiA(IDCurrentCell,1) = Ca*CellDensityMember(IDx,IDy,IDz)*HillFunction...
      -GammaA*AiiA(IDCurrentCell,1)/...
      (1 + f*(AiiA(IDCurrentCell,1) + LuxI(IDCurrentCell,1)));
    dMLuxI(IDCurrentCell,1) = Ci*CellDensityMember(IDx,IDy,IDz)*HillFunction...
      -GammaI*LuxI(IDCurrentCell,1)/...
      (1 + f*(AiiA(IDCurrentCell,1) + LuxI(IDCurrentCell,1)));  
  end
end

MHe = MHe + dt*dMHe;
MHi(1:nat,:) =  DivisionCorrection(1:nat,:).*((MHi(1:nat,:)) + ...
    dt*VcellOld(1:nat,:).*b*LuxI(1:nat,:)/(1+kCoef*LuxI(1:nat,:)) - ...
    GammaHI*AiiA(1:nat,:).*Hi(1:nat,:)/...
    (1+g*AiiA(1:nat,:))));

MAiiA(1:nat,:) = DivisionCorrection(1:nat,:).*((MAiiA(1:nat,:)) + ...
    dt*VcellOld(1:nat,:).*dMAiiA(1:nat,:));
MLuxI(1:nat,:) = DivisionCorrection(1:nat,:).*((MLuxI(1:nat,:)) + ...
    dt*VcellOld(1:nat,:).*dMLuxI(1:nat,:));

% Update He, Hi, AiiA and LuxI Concentration
VeNew = Vt + dVeNew;
He = MHe ./ VeNew;
Hi(1:nat,:) = MHi(1:nat,:) ./ VcellNew(1:nat,:);
AiiA(1:nat,:) = MAiiA(1:nat,:) ./ VcellNew(1:nat,:);
LuxI(1:nat,:) = MLuxI(1:nat,:) ./ VcellNew(1:nat,:);

% Save the new volume as old
VcellOld = VcellNew;
CellSurfaceOld = CellSurfaceNew;

% Update The Memory Matrix with the current Hi value
if t < tau
  % Fill the HimemoryMatrix
  Him(round(t/dt),1:nat) = (Hi(1:nat,:))';
else
  % Write at the Oldest Record Position
  Him(OldestRecord,1:nat) = (Hi(1:nat,:))';
end
% Update the Index, Make Next Record the Oldest
OldestRecord = OldestRecord + 1;

% Reset the index
if OldestRecord > dimHim
    OldestRecord = 1;
end

%% P R O C E S S: OUTPUT

if t >= ntp %
    tPrint(:,1) = t;
    DataRowStep1 = [tPrint IDPrint apa apb apc apd aab aab H.I AiiA LuxI];
    Output_01(FileName_01, FileName_01_Length, DataRowStep1(1:nat,:), nat, 19);
    disp(DataRowStep1(1:nat,:))

    %% Export H.I Cell Concentration
    DataRowCell = [ t nat H.I'];
    Output_04(FileName_04, FileName_04_Length, DataRowCell, 1, dimna+2)

    %% Export AiiA Cell Concentration
    DataRowCell = [ t nat AiiA'];
    Output_04(FileName_05, FileName_05_Length, DataRowCell, 1, dimna+2)

    %% Export LuxI Cell Concentration
    DataRowCell = [ t nat LuxI'];
    Output_04(FileName_06, FileName_06_Length, DataRowCell, 1, dimna+2)

    %% Export He Parcel
    DataRowParcel = [ t nat He'];
    Output_03(FileName_03, FileName_03_Length, DataRowParcel, 1, inx_g2+2)

    %% Export Oldvolume Cell
    if OutputVcellOld == 1
        DataRowCell = [ t nat VcellOld'];
        Output_04(FileName_07, FileName_07_Length, DataRowCell, 1, dimna+2)
    end

    %% Export NewVolume Cell
    if OutputVcellNew == 1
        DataRowCell = [ t nat VcellNew'];
        Output_04(FileName_08, FileName_08_Length, DataRowCell, 1, dimna+2)
    end

    %% C O M P U T E THE Luxi MASS IN THE PARCEL AND OUTPUT
    MLuxIParcel = zeros(inx_g2,iny_g2,inz_g2);
    for i = 1:inx_g2
        for j = 1:iny_g2
            for k = 1:inz_g2

                % Reset: Local Variables
                SummMLuxI = 0;
                NumberOfCellsInCurrentParcel = ina_g2(i,j,k);
                for ii = 1:NumberOfCellsInCurrentParcel
                    IDCurrentCell = iid_g2(i,j,k,ii);
                    MLuxICurrentCell = MLuxI(IDCurrentCell,1);
                    SummMLuxI = SummMLuxI + MLuxICurrentCell;
                end

                MLuxIParcel(i,j,k) = SummMLuxI;
            end
        end
    end

    %% Export MassLuxI Parcel
    DataRowParcel = [ t nat MLuxIParcel'];
    Output_03(FileName_09, FileName_09_Length, DataRowParcel, 1, inx_g2+2)

    if OutputCellDenMem == 1
        DataRowCell = [ t nat CellDensityMember'];
        Output_03(FileName_10, FileName_10_Length, DataRowCell, 1, inx_g2+2)
    end

    if OutputdCoef == 1
        DataRowParcel = [ t nat (dVeOld./(Vt))'];
        Output_03(FileName_11, FileName_11_Length, DataRowParcel, 1, inx_g2+2)
    end

    %% Update Time Print Step
ntp = ntp + dtp;
end

%% PROCESS: TIMEUPDATE

t = t + dt;
end

%% MAIN LOOP - END

%% SAVE and OUTPUT: Final Variables Values

DataVariablesValues = [ t dt tend dtp ntp dimna nat ...
lt0 lc0 v0 rho ltc ltd kmp np ...
umax SF ... 
MaxContactRange1 MaxContactRangeSql MaxContactRangeSq2 ...
kf rfx rfy rfz rtx rty rtz ...
Xt Yt Zt Vchamber ...
inx_g1 idx_g1 iny_g1 idy_g1 inz_g1 idz_g1 SF_g1 Maxnat_g1 ...
inx_g2 idx_g2 iny_g2 idy_g2 inz_g2 idz_g2 Vt SF_g2 Maxnat_g2...
Mu D1 D2 ...
Ca C1 GammaA GammaI f tau delta alpha ...
k1 b kCoef GammaHi g CellDensityCoefD0 ...
OldestRecord...
HiInitial...
];
Output_05(FileName_00, FileName_00_Length, DataVariablesValues, 1, size(DataVariablesValues,2))
end
6.10. ABM Of Bacteria Code Subfunctions

6.10.1. FC4.m – Contact Function Code

---

% IBM/ABM of Bacteria Code
% ‘CONTACT FUNCTION’

% Program: ‘Contact Function’ for IBM/ABM of Bacteria
% Version: 4
% Date: 05/10/2013
% Author: Kameliya Z. Koleva
% Ferdi L. Hellweger
% Department of Civil and Environmental Engineering
% Northeastern University
% Boston
% kamikoleva@gmail.com
% ferdi@coe.neu.edu

function [Contact,C1,C2,distance] = FC4(A1,B1,...
A2B1,...
A2B2,...
LSQB2,...
A1A2, B1A2, B1B2, A1B2,...
LSQA1A2, LSQB1A2, LSQB1B2, LSQA1B2,...
MinDistance)

% WHAT THE FUNCTION NEEDS
% Taking the coordinates of the two agents, that can be potentially in contact
% their squared axis lengths (the squared segment lengths EX: |A1B1|^2 and |A2B2|^2)
% the squared lengths of the four segments connecting the agents Endsegmentpoints
% (EX: |A1A2|^2, |A1B2|^2, |B1A2|^2, |B1B2|^2)
% And the value of the variable MinDistance, that contains the minimum of
% the four segments connecting the agents Endsegmentpoints

% WHAT THE FUNCTION DOES AND WHAT IT RETURNS
% The function assess if the axes of the agents are in the same plane or not
% to the other agent axis, compares if the length is shortest, and if yes,
% assigns the position of the perpendicular heel point is inner or external point,
% depending of this computations it assigns the centers of the virtual spheres,
% gives the distance between them and
% gives if the agents are in contact (Yes/No)
% If this distance is less than or equal twice the radius (2*r0)

% MAIN VARIABLES
% A1 and B1 are end points of the first agent axis
% A2 and B2 are end points of the second agent axis
% When A1B1 and A2B2 are segments,
% A1B1 segment length: ((A1B1(1))^2 + (A1B1(2))^2 + (A1B1(3))^2)^0.5
% When A1B1 and A2B2 are lines, their length is infinity, but the
direction is the same as A1B1 and A2B2 segments respectively
% C1 and C2 are virtual Sphere Centers,
% C1C2 segment is the closest distance between A1B1 and A2B2 segments

% GENERAL COPLANAR CONFIGURATION
% K is point at the heel of the perpendicular from one of the ends of the
% first Agent to the axis of the other, or from one of the second Agent
to the axis of First Agent.
% Ex: From point A2, perpendicular to axis A1B1 is A2K (A2K ___ A1B1)
% Ex: From point A1, perpendicular to axis A2B2 is A1K (A1K ___ A2B2)
% Ex: From point B1, perpendicular to axis A2B2 is B1K (B1K ___ A2B2)
% K can be inner and external point in relation to the segments A1B1 or A2B2
% K is inner when is inside the segment A1B1 or A2B2
% K is external when is outside the segment A1B1 or A2B2,
% but still belongs to the line A1B1 or A2B2 !!!
% GENERAL SPACE CONFIGURATION ( NO COPLANAR lines)
% K1K2 is the shorstes transversal, perpendicular to both lines A1B1 and A2B2
% K1 can be inner and external point in relation to the segment A1B1
% K1 is inner when is inside the segment A1B1
% K1 is external when is outside the segment A1B1
K2 is inner when is inside the segment A2B2
K2 is external when is outside the segment A2B2

when K, K1 or K2 are inner points:
C1 = K, or C2 = K in GENERAL COPLANAR CONFIGURATION
(The Virtual Spheres Centers coincide with corresponding point K)
C1 = K1 and/or C2 = K2 in GENERAL SPACE CONFIGURATION (NO COPLANAR lines)
(The Virtual Spheres Centers coincide with corresponding point K)

when K, K1 or K2 are external points:
GENERAL COPLANAR CONFIGURATION
C1 = A1, or C1 = B1, depending on which point A1 or B1 is closest to point K
C2 = A2, or C2 = B2, depending on which point A1 or B1 is closest to point K
(The Virtual Spheres Centers coincide with the closest end of the A1B1 or A2B2 Segment)
GENERAL SPACE CONFIGURATION (NO COPLANAR lines)
C1 = A1, or C1 = B1, depending on which point A1 or B1 is closest to point K1
C2 = A2, or C2 = B2, depending on which point A1 or B1 is closest to point K2
(The Virtual Spheres Centers coincide with the closest end of the A1B1 or A2B2 Segment)

The function used here "PerpendicularLine4" is taking:
- the one of coordinates of one agent (EX: point A1)
- the length of its axis (EX: |A1B1| - A1B1 segment length)
- and the coordinates of one of the other agent Endsegmentpoints
- The function is building the perpendicular from one of the other agent Endsegmentpoints to the axis of the other agent
- The output is:
  1) The coordinates of the heel of the perpendicular,
  2) The squared length of the perpendicular
  3) Inner/ External position (Yes/No)

function [Contact, C1, C2, distance] = FC4(A1,B1,A1B1,LsqA1B1,...
A1A2,B1A2,B1B2,A1B2,...
LsqA1A2, LsqB1A2, LsqB1B2, LsqA1B2, ...
MinDistance)

% BASIC VECTORS AND POINTS
A2A1 = -A1A2;
A2B1 = -B1A2;

% SPACE CONFIGURATION CONDITION: abs(det([A1 1;A2 1;B1 1;B2 1])) <= 0.05
if abs(A1(1)*A2(2)*B1(3) - A1(1)*A2(3)*B1(2) - A1(2)*A2(1)*B1(3) + A1(2)*A2(3)*B1(1) + ...
A1(3)*A2(1)*B1(2) - A1(3)*A2(2)*B1(1) - A2(1)*A1(2)*B2(3) + A2(1)*A1(3)*B2(2) + ...
== 0.05
% CASE I - PLANE CONFIGURATION <<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<
% Assign the Virtual Sphere Centers depending which segment, connecting the Agent Endsegmentpoints is shortest
if MinDistance == LsqA1A2
  C1 = A1; C2 = A2;
elseif MinDistance == LsqA1B2
  C1 = A1; C2 = B2;
elseif MinDistance == LsqB1A2
  C1 = B1; C2 = A2;
elseif MinDistance == LsqB1B2
  C1 = B1; C2 = B2;
end

% COLLINEAR CONFIGURATION CHECK: (dot(A1B1, A1A2))^2 == |A1B1|^2*|A1A2|
if (A1B1(1)*A1A2(1)+A1B1(2)*A1A2(2)+A1B1(3)*A1A2(3))^2 == (LsqA1B1)*(LsqA1A2)
% Find point K, heel of the perpendicular from point A2 to A1B1 line.
% Assign Virtual sphere centers, if point k is inner and shorter than MinDistance
[X, LsqCK, InnerPoint] = PerpendicularLine4(A1,A2,A1B1,A1A2,LsqA1B1);
if InnerPoint
  if LsqCK < MinDistance
    MinDistance = LsqCK;
    C1 = K;
  end
end
\[ C_2 = A_2; \]
\[ \text{end;} \]
\[ \text{end;} \]
\[% Find point } K_1, \text{ heel of the perpendicular from point } B_2 \text{ to } A_1B_1 \text{ line, } \%
\% Assign Virtual sphere Centers, if point } K \text{ is inner and shorter than } MinDistance \\
[k, LsqCK, InnerPoint] = PerpendicularLine4(A1, B2, A1B1, A1B2, LsqA1B1);
\% If InnerPoint
\% If LsqCK < MinDistance
\% MinDistance = LsqCK;
\% C1 = K;
\% C2 = B2;
\% end;
\%
\% Find point } K_2, \text{ heel of the perpendicular from point } A_1 \text{ to } A_2B_2 \text{ line, } \%
\% Assign Virtual Sphere Centers, if point } K \text{ is inner and shorter than } MinDistance \\
\% If InnerPoint
\% If LsqCK < MinDistance
\% MinDistance = LsqCK;
\% C2 = K;
\% C1 = A1;
\% end;
\%
\% Find point } K_3, \text{ heel of the perpendicular from point } B_1 \text{ to } A_2B_2 \text{ line, } \%
\% Assign Virtual Sphere Centers, if point } K \text{ is inner and shorter than } MinDistance \\
[k, LsqCK, InnerPoint] = PerpendicularLine4(A2, B1, A2B2, A2B1, LsqA2B2);
\% If InnerPoint
\% If LsqCK < MinDistance
\% MinDistance = LsqCK;
\% C2 = K;
\% C1 = B1;
\% end;
\%
\% Compute distance between centers of the virtual spheres, contact - Yes/No \\
\text{distance = MinDistance}^{0.5}; \]
\else \]
\%
\text{CASE II - GENERAL CONFIGURATION} <<<<<<<<<<<<<<<<<<<<<<<<<<<<< \\
\%
\text{Find } K_1K_2, \text{ shortest transversal, perpendicular to both } A_1B_1 \text{ and } A_2B_2 \text{ Lines}
\]
\text{U1 = } A1B1(1)^2 + A1B1(2)^2 + A1B1(3)^2; \text{ U2 = } A2B2(1)^2 + A2B2(2)^2 + A2B2(3)^2; \\
\text{V1 = -dot((A1B1),(A1B1)); V2 = -U1; W1 = -dot((A1B1),(A1B1)); W2 = -U2; q = det([W1 V1; W2 V2])/det([U1 V1; U2 V2]); p = det([U1 W1; U2 W2])/det([U1 V1; U2 V2]); K1 = p*A1 + (1-p)*B1; K2 = q*A2 + (1-q)*B2; K1A1 = A1 - K1; K1B1 = B1 - K1; K2A2 = A2 - K2; K2B2 = B2 - K2; \]
\text{Find First Virtual Sphere Center} \%
\text{K1 Inner/External Check, (inside or outside of the segment } A2B2): \text{ dot(k1A1,k1B1) <= 0} \%
\text{if } (k1A1(1)*k1B1(1) + k1A1(2)*k1B1(2) + k1A1(3)*k1B1(3)) <= 0 \%
\text{K1 is inner point, } C1 = K1; \%
\text{else } \%
\text{K1 is external point, if min(|k1A1|,|k1B1|) == |k1A1|, then } C1 = A1, \text{ if not } C1 = B1 \%
\%
\%
\%
\text{Find Second Virtual Sphere Center} \%
\text{K2 Inner/External Check, (inside or outside of the segment } A2B2): \text{ dot(k2A2,k2B2) <= 0} \%
\text{if } (k2A2(1)*k2B2(1) + k2A2(2)*k2B2(2) + k2A2(3)*k2B2(3)) <= 0
% K2 is inner point, C2 = K2
    C2 = K2;
else
  % K2 is external point, if min(|K2A2|,|K2B2|) == |K2A2|, then C2 = A2, if not C2 = B2
  if min((K2A2(1)^2 + K2A2(2)^2 + K2A2(3)^2), (K2B2(1)^2 + K2B2(2)^2 + K2B2(3)^2))
    C2 = A2;
  else
    C2 = B2;
  end
end;

%% COMPUTE DISTANCE BETWEEN CENTERS OF THE VIRTUAL SPHERES
C1c2 = C2 - C1;
distance = ((C1c2(1))^2 + (C1c2(2))^2 + (C1c2(3))^2)^0.5;
end;

%% CONTACT - YES / NO
if distance <= 0.5;
  Contact = 1;                % Contact - YES
else
  Contact = 0;                % Contact - NO
end;
end
% Program: 'Perpendicular Line Function' for IBM/ABM of Bacteria
% Version: 4
% Date: 05/10/2013
% Author: Kameliya Z. Koleva
% Ferdi L. Hellweger
% Department of Civil and Environmental Engineering
% Northeastern University
% Boston
% kamikoleva@gmail.com
% ferdi@coe.neu.edu
%
% The function used here "PerpendicularLine3" is taking:
% the one of coordinates of one agent (EX: point A1)
% the length of its axis (EX: |A1B1| - A1B1 segment length)
% and the coordinates of one of the other agent endsegmentpoints
% The function is building the perpendicular from one of the other agent
% endsegmentpoints to the axis of the other agent,
% The output is:
% 1) The coordinates of the heel of the perpendicular,
% 2) The squared length of the perpendicular
% 3) Inner/ External position (Yes/No)

%#codegen
coder.inline('never')

% Definitions of the output variables ----------------
K = rand(1,3);
LsqCK = rand(1,1);
InnerPoint = rand(1,1);
% ----------------------------------------------------
% COMPUTE m = dot(AC,AB)/dot(AB,AB);
% C O M P U T E coordinates of point K: K = A + mAB;
% C O M P U T E CK = K-C ;
% COMPUTE LsqCK = ((CK(1))^2 + (CK(2))^2 + (CK(3))^2);

m = (AC(1)*AB(1) + AC(2)*AB(2) + AC(3)*AB(3))/(LsqAB);
K = [A(1) + m*AB(1)  A(2) + m*AB(2)  A(3) + m*AB(3)];
CK = K - C;
LsqCK = ((CK(1))^2 + (CK(2))^2 + (CK(3))^2);

if m >= 0 && m <= 1
% K is INSIDE point for the AB segment;
InnerPoint = 1;
else
% K is OUTSIDE point for the AB segment;
InnerPoint = 0;
end
end
6.10.3. Output_01.m

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% IBM/ABM of Bacteria CODE
% 'Output_01 FUNCTION'
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% Program: 'Perpendicular Line Function' for IBM/ABM of Bacteria
% Version: 2
% Date: 05/10/2013
% Author: Kameliya Z. Koleva
% Ferdi L. Hellweger
% Department of Civil and Environmental Engineering
% Northeastern University
% Boston
% kamikoleva@gmail.com
% ferdi@coe.neu.edu
%

function [] = Output_01(FileName, FileNameLenght, OutputData, OutputSize1, OutputSize2)
coder.inline('never')
coder.ceval( 'myoutput', FileName, FileNameLenght, OutputData, OutputSize1, OutputSize2);
end

% Matlab Version: to run the model in Matlab uncomment below, comment above
% function [] = Output_01(FileName, FileNameLenght, OutputData, OutputSize1, OutputSize2)
% dlmwrite(FileName, OutputData,'-append','coffset',0,'delimiter','\t','precision','%6.6G')
% precision = 4;
% disp( num2str( OutputData,precision))
% end

6.10.4. Output_02.m

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% IBM/ABM of Bacteria CODE
% 'Output_02 FUNCTION'
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% Program: 'Perpendicular Line Function' for IBM/ABM of Bacteria
% Version: 2
% Date: 05/10/2013
% Author: Kameliya Z. Koleva
% Ferdi L. Hellweger
% Department of Civil and Environmental Engineering
% Northeastern University
% Boston
% kamikoleva@gmail.com
% ferdi@coe.neu.edu
%

function [] = Output_02(FileName, FileNameLenght, OutputData, OutputSize1, OutputSize2)
coder.inline('never')
coder.ceval( 'myoutput', FileName, FileNameLenght, OutputData, OutputSize1, OutputSize2);
end

% Matlab Version: to run the model in Matlab uncomment below, comment above
% function [] = Output_02(FileName, FileNameLenght, OutputData, OutputSize1, OutputSize2)
% dlmwrite(FileName, OutputData,'-append','coffset',0,'delimiter','\t','precision','%6.6G')
% end
function [] = Output_03(FileName, FileNameLenght, OutputData, OutputSize1, OutputSize2)
coder.inlin('never')
coder.ceval('myoutput', FileName, FileNameLenght, OutputData, OutputSize1, OutputSize2);
end

function [] = Output_04(FileName, FileNameLenght, OutputData, OutputSize1, OutputSize2)
coder.inlin('never')
coder.ceval('myoutput', FileName, FileNameLenght, OutputData, OutputSize1, OutputSize2);
end
function [] = Output_05(FileName, FileNameLength, OutputData, OutputSize1, OutputSize2)
coder.inline('never')
coder.ceval( 'myoutput', FileName, FileNameLength, OutputData, OutputSize1, OutputSize2);
end

% ## Matlab Version: to run the model in Matlab uncomment below, comment above
% function [] = Output_05(FileName, FileNameLength, OutputData, OutputSize1, OutputSize2)
% dlmwrite(FileName, OutputData,'-append','coffset',0,'delimiter','\t','precision','%6.6G')
% end
6.11. ABM Of Bacteria Animation Code

6.11.1. ABM_Animation_2013_07_17_V1.m

clear all; clc

%% INPUT FILES
FilenameStep0 = '2013-07-16-19-58-03-Step-00-Variables-From-ABM_2013_07_11_V_02_func.txt';  % Coordinates of points, age and concentrations
FilenameStep1 = '2013-07-16-19-58-03-Step-01-From-ABM_2013_07_11_V_02_func.txt';            % Virtual spheres coordinates
FilenameStep2 = '2013-07-16-19-58-03-Step-02-From-ABM_2013_07_11_V_02_func.txt';            % External Concentration of the parcel
FilenameStep3 = '2013-07-16-19-58-03-Step-03-He-From-ABM_2013_07_11_V_02_func.txt';         % contains parameters for the environment

%% Options of the animation
Recording = 1;  % 1, 4, 24, 25, 36
NumberOfFrames = 1;   % 1, 4, 24, 25, 36
Manual = 0;  % pres any key to go on the next frame
WindowSize = [0 0 1920 1080];        % or [0 0 3750 1500];
% windowSize = get(0,'ScreenSize');
ShowContacts = 0;  % Mirtual Spheres ON/OFF --> 1/0
ShowExternalConcentration = 0;       % External Concentration Visualization ON/OFF --> 1/0
ShowParcelVerticalBorderwalls = 0;   % Show walls separating the parcels
CellColorGradient = 0;               % Internal AHL Cell Concentration Visualization ON/OFF --> 1/0
ColorMode = 4;                       % Choose color sheme for cells See the example below; Possible Values from 1 to 6
ID = 1;                              % ID of the agent that will be in red color in case ColorMode 3 is choosen ; Possible values from 1 to MaxNumberOfBacteria
vrsetpref DefaultViewer internalv4

%%

Datstep1 = load(FilenameStep1, '-ascii');
DataStep0 = (load(FilenameStep0, '-ascii'))';

%% Max Number Of Bacteria
MaxNumberOfBacteria = DataStep1(end,2);
if ShowContacts == 1
DataStep2 = load(FilenameStep2, '-ascii'));
MaxNumberOfContacts = max(DataStep2(:,2));
end

%% Environment: Wall Parameters
Lx = DataStep0(29);             % length of the environment in X direction
Ly = DataStep0(30);             % length of the environment in Y direction
Lz = DataStep0(31);             % length of the environment in Z direction
Ldx = DataStep0(42);            % length of 1 parcel in X direction
ParcelNumber = DataStep0(41);   % Number of Parcels in x direction
t0 = 0.1;                       % Thickness Of The wall

SHIFTING = [DataStep1(1,3) DataStep1(1,4) 50]; % the last number controls the zoom

%% CellColors
if CellColorGradient == 1
DefaultCellColor = [0 24, 25, 36];
else
CellColor = CellColorsMode(ColorMode, MaxNumberOfBacteria, ID);  % ColorMode for the cells
CellColor = CellColorsMode(1,MaxNumberOfBacteria);  % random colors for all
CellColor = CellColorsMode(2,MaxNumberOfBacteria);  % White Color for all
CellColor = CellColorsMode(3,MaxNumberOfBacteria, ID);  % White Color for all, but one agent
with ID = ?
CellColor = CellColorsMode(4,MaxNumberOfBacteria);  % Blue Color for all
CellColor = CellColorsMode(5,MaxNumberOfBacteria);  % Grey Color for all
CellColor = CellColorsMode(6,MaxNumberOfBacteria);  % First 28 colors predefined, then
random colors

71
if ShowContacts == 1
    VirtualSpheresColors = CellColorsMode(ColorMode, MaxNumberOfContacts, ID);
    CellColor = CellColorsMode($, MaxNumberOfBacteria, ID);
end
end
DefaultParcelColor = [0 0.80 0.85];

wallColors
VerticalMainWallColor = [0.17 0.17 0.17]; % Main walls color
BottomMainWallColor = [0.15 0.12 0.12]; % Main Bottom walls color
MainWallsNumber = 6;
% wall Options and Properties
wallTransparency = 0; % main transparency of the walls
wallTransparency2 = 1; % transparency that need to be different
ParcelVerticalBorderwallsColor = [0.75 0 0]; % Walls separating the parcels
ParcelBottomBorderwallsColor = [0 1 0]; % Bottom walls of the parcels for external Concentration visualization
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%
%% OUTPUT FILES NAMES
Date = datestr(now,'yyyy-mm-dd-HH-MM-SS' );
FileNameVideoOutput = [Date, '-Video-Generated-From-', FilenameStep1, '.avi'];
%% Create and open the virtual world object
VirtualWorld = vrworld('ProtoEnvironment.wrl', 'new');
open(VirtualWorld);
%% Create the figure
Figure = vrfigure(VirtualWorld);
if Recording == 1
    set(Figure,'NavPanel','none','Position', WindowSize);
    VideoObject = VideoWriter(FileNameVideoOutput, 'Uncompressed AVI');
    open(VideoObject);
end
%% get the path to the wrl file with bacterium PROTOs
ProtoSphere   = which('ProtoSphere.wrl');
ProtoCylinder = which('ProtoCylinder.wrl');
ProtoWall     = which('ProtoWall.wrl');
%% create an EXTERNPROTO with specified marker
addexternproto(VirtualWorld, ProtoSphere, 'Sph');
addexternproto(VirtualWorld, ProtoCylinder, 'Cyl');
addexternproto(VirtualWorld, ProtoWall, 'Wall');
%% create the DefaultCylinder coordinates
DefaultCylinder = [0 1 0];
% AgentColor: control the color of agents in case of virtual spheres,
% make the agents grey, and give color only to the virtual sphere
if ShowContacts == 1;
    [VirtualSpheresColors] = CellColorsMode(ColorMode, MaxNumberOfContacts, ID);
    CellColors = ones(MaxNumberOfBacteria, 3);
else
    [CellColors] = CellColorsMode(ColorMode, MaxNumberOfBacteria, ID);
end
% Objects creation
% walls build
% Left wall
wall1 = vrnode(VirtualWorld,sprintf('wall_%d',1),'Wall');
wall1.wallTrans = wallTransparency;
wall1.wallColor = VerticalMainWallColor;
wall1.wallScale = [t0 Ly Lz];
wall1.wallPosition = [(0-t0/2) Ly/2 Lz/2] - SHIFTING;
% Front wall
wall2 = vrnode(VirtualWorld,sprintf('wall_%d',2),'Wall');
wall2.wallTrans = wallTransparency;
wall2.wallColor = VerticalMainWallColor;
wall2.wallScale = [Lx t0 Lz];
wall2.wallPosition = [Lx/2 (0-t0/2) Lz/2] - SHIFTING;
% Bottom wall
wall3 = vrnode(VirtualWorld,sprintf('wall_%d',3),'Wall');
wall3.wallTrans = wallTransparency;
wall3.wallColor = BottomMainWallColor;
wall3.wallScale = [Lx Ly t0 ];
wall3.wallPosition = [Lx/2 Ly/2 (0-t0/2)] - SHIFTING;
% Right Wall
wall14 = vrnode(VirtualWorld,sprintf('Wall_%d',4),'Wall');
wall14.WallTrans = WallTransparency;
wall14.WallColor = VerticalMainWallColor;
wall14.WallScale = [t0 Ly Lz];
wall14.WallPosition = [(t0/2+Lx) Ly/2 Lz/2] - SHIFTING;

% Back Wall
wall15 = vrnode(VirtualWorld,sprintf('Wall_%d',5),'Wall');
wall15.WallTrans = WallTransparency;
wall15.WallColor = VerticalMainWallColor;
wall15.WallScale = [Lx t0 Lz];
wall15.WallPosition = [Lx/2 (t0/2+Ly) Lz/2] - SHIFTING;

% Top Wall
wall16 = vrnode(VirtualWorld,sprintf('Wall_%d',6),'Wall');
wall16.WallTrans = 1;
wall16.WallColor = VerticalMainWallColor;
wall16.WallScale = [Lx Ly t0];
wall16.WallPosition = [Lx/2 Ly/2 (t0/2+Lz)] - SHIFTING;

% HelpVerticalWalls Creation
MaxNumberVertWalls = ParcelNumber + MainWallsNumber;
if ShowParcelVerticalBorderWalls == 1
for WallNumber = 1:ParcelNumber-1
ParcelVerticalBorderWalls = vrnode(VirtualWorld,sprintf('Wall_%d',(WallNumber+MainWallsNumber)),'Wall');
ParcelVerticalBorderWalls.WallTrans = WallTransparency; % transparent
ParcelVerticalBorderWalls.WallColor = ParcelVerticalBorderWallsColor;
ParcelVerticalBorderWalls.WallScale = [t0 Ly Lz];
ParcelVerticalBorderWalls.WallPosition = [(t0/2+WallNumber*Ldx) Ly/2 Lz/2] - SHIFTING;
end
end

% HelpBottomWalls Creation
if ShowExternalConcentration == 1
ParcelNumberBottom = ParcelNumber;
for WallNumber = 1:ParcelNumberBottom
ParcelBottomBorderWalls = vrnode(VirtualWorld,sprintf('Wall_%d',(WallNumber+ParcelNumber+MainWallsNumber-1)),'Wall');
ParcelBottomBorderWalls.WallTrans = 1; % transparent
ParcelBottomBorderWalls.WallScale = [Lx Ly t0 ];
ParcelBottomBorderWalls.WallPosition = [(t0/2+(WallNumber-0.5)*Ldx) Ly/2 Lz/2] - SHIFTING;
end
end

% Create All Cells
for i = 1 : MaxNumberOfBacteria

%% Sphere A
SphA = vrnode(VirtualWorld,sprintf('SphA_%d',i),'Sph');
SphA.SphTrans = 1;

%% Sphere B
SphB = vrnode(VirtualWorld,sprintf('SphB_%d',i),'Sph');
SphB.SphTrans = 1;

%% Cylinder
Cylinder = vrnode(VirtualWorld,sprintf('Cylinder_%d',i),'Cyl');
Cylinder.CylTrans = 1;
end;

% Create All Virtual Spheres
if ShowContacts == 1;
clear DataStep2
for i = 1 : MaxNumberOfContacts

%% Sphere A
SphV1 = vrnode(VirtualWorld,sprintf('SphV1_%d',i),'Sph');
SphV1.SphTrans = 1;

SphV2 = vrnode(VirtualWorld,sprintf('SphV2_%d',i),'Sph');
SphV2.SphTrans = 1;
end;
end

clear DataStep1 DataStep0
% Start preparing for the main loop
tPrev = 0;
i2 = 1;
i3 = 1;
FileID  = fopen(FilenameStep1);
if ShowExternalConcentration == 1
FileID3 = fopen(FilenameStep3);
end

73
%% Open Input File Step2 if Virt. Sph on
if ShowContacts == 1;
    FileID2 = fopen(FilenameStep2);
    DataV = fscanf(FileID2, '%f %f %f %f %f %f %f %f %f %f %f',11);
    DataV = DataV';
    tvirtSph = DataV(1);
end
Data = fscanf(FileID, '%f %f %f %f %f %f %f %f %f %f %f %f %f %f %f %f %f %f %f',19);
Data = Data';
t = Data(1);

%% MAIN LOOP Start
while ~feof(FileID) == 1
    % Read from file and get the coordinates for one cell
    Data = fscanf(FileID, '%f %f %f %f %f %f %f %f %f %f %f %f %f %f %f %f %f %f %f',19);
    Data = Data';
    if numel(Data)==0

    t = Data(1);                % time
    n = Data(2);                % ID of Agent
    A = Data(3:5) - SHIFTING;   % p. A coordinates
    B = Data(6:8) - SHIFTING;   % p. B Coordinates
    AB = B - A;                 % Axis for the capsule

    % READ THE DATA FOR LUXI AND GIVE CELLCOLOR TO THE CELL
    if CellColorGradient == 1
        a1 = 1; b1 = 1000;
        CellColor = a1*(Data(end)/b1)*DefaultCellColor;
    end

    % Hide everything if the time is different, or show Virt Sph and external concentration and then hide all
    if t > tPrev
        % Read the external concentration Data for the next time step
        if ShowExternalConcentration ==1
            Data2 = fscanf(FileID3, '%f %f %f %f %f %f %f %f %f %f %f %f ',12);
            Data2 = Data2';
            ConcentrationAllParcels = Data2(3:end);
            a = 10; b = 1;
            for WallNumber = 1:ParcelNumberBottom
                ParcelBottomBorderWalls = vrnode(VirtualWorld,sprintf('Wall_%d',WallNumber+ParcelNumber+6-1));
                ParcelBottomBorderWalls.WallColor    = a*(ConcentrationAllParcels(WallNumber))/b * DefaultParcelColor;
                ParcelBottomBorderWalls.WallTrans    = 0;
            end
        end

    % Draw the Virt Sphere
    if ShowContacts == 1;
        if numel(Data)==0
            while tvirtSph==tPrev && ~feof(FileID2) && ShowContacts
                SphV1 = vrnode(VirtualWorld,sprintf('SphV1_%d',i3));
                SphV2 = vrnode(VirtualWorld,sprintf('SphV2_%d',i3));
                i3=i3+1;
                IDContact = DataV(2);
                SphV1.SphColor = VirtualSpheresColors(IDContact,:);
                SphV2.SphColor = SphV1.SphColor;
                SphV1.SphTrans = 0;
                SphV1.SphPosition = [DataV(5) DataV(6) DataV(7)] - SHIFTING;
                SphV2.SphPosition = [DataV(8) DataV(9) DataV(10)] - SHIFTING;
            end
        end
    end
end

end

%% MAIN LOOP Start
Read from File and get the coordinates for Virt. Sphere
```
dataV = fscanf(FileID2, '%f %f %f %f %f %f %f %f %f %f %f',11);
dataV = DataV';
tVirtSph = DataV(1);
end
```

%% External Concentration
```
if ShowExternalConcentration == 1
   for WallNumber = 1:ParcelNumberBottom
      ParcelBottomBorderWalls = vrnode(VirtualWorld,sprintf('Wall_%d',WallNumber+ParcelNumber+6-1));
      ParcelBottomBorderWalls.WallTrans = 1;
   end
end
```

%% PAUSE IF MANUAL MODE
```
if Manual == 1
   pause
else
   pause(0.1)
end
```

%% DRAW THE OBJECTS
```
vrdrawnow;
```

%% RECORD THE FRAME
```
if Recording == 1
   image_capture = capture(Figure);
   for k = 1:NumberOfFrames
      writeVideo(VideoObject,image_capture);
   end
end
```

%% HIDE ALL OBJECTS (PREPARE FOR THE NEXT FRAME)
```
if ShowContacts == 1;
   if numel(DataV)==0
      for i = 1:MaxNumberOfBacteria
         % Sphere A
         SphA = vrnode(VirtualWorld,sprintf('SphA_%d',i));
         SphA.SphTrans = 1;
         % Sphere B
         SphB = vrnode(VirtualWorld,sprintf('SphB_%d',i));
         SphB.SphTrans = 1;
         % Cylender
         Cylinder = vrnode(VirtualWorld,sprintf('Cylinder_%d',i));
         Cylinder.CylTrans = 1;
      end;
   end
end
```

```
i2 = 1;
```

```
if ShowContacts == 1;
   for i = 1:MaxNumberOfContacts
      % Sphere C1
      SphV1 = vrnode(VirtualWorld,sprintf('SphV1_%d',i));
      SphV1.SphTrans = 1;
      % Sphere C2
      SphV2 = vrnode(VirtualWorld,sprintf('SphV2_%d',i));
      SphV2.SphTrans = 1;
   end;
end
```

```
i3 = 1;
```

```
elseif t == tPrev && numel(Data)==0
   % Read the external concentration Data for the next time step
   if ShowExternalConcentration == 1
      Data2 = fscanf(FileID3, '%f %f %f %f %f %f %f %f %f %f %f %f ',12);
      Data2 = Data2';
      ConcentrationAllParcels = Data2(3:end);
      a = 10; b = 1;
   end
```

```
for WallNumber = 1:ParcelNumberBottom
   ParcelBottomBorderWalls = vrnode(VirtualWorld,sprintf('Wall_%d',WallNumber+ParcelNumber+6-1));
   ParcelBottomBorderWalls.WallColor = a*(ConcentrationAllParcels(WallNumber))/b * DefaultParcelColor;
end
```

75
ParcelBottomBorderWalls.wallTrans  = 0;
end

%% Draw the Virt Sphere
if ShowContacts == 1;
  if numel(DataV)==0
    while tVirtSph==tPrev  && ~feof(FileID2) && ShowContacts
      pause(0.1)
      SphV1 = vrnode(VirtualWorld,sprintf('SphV1_%d',i3));
      SphV2 = vrnode(VirtualWorld,sprintf('SphV2_%d',i3));
      i3=i3+1;
      IDContact = DataV(2);
      SphV1.SphColor = VirtualSpheresColors(IDContact,:);
      SphV2.SphColor = SphV1.SphColor;
      SphV1.SphTrans = 0;
      SphV1.SphPosition = [DataV(5) DataV(6) DataV(7)] - SHIFTING;
      SphV2.SphPosition = [DataV(8) DataV(9) DataV(10)]- SHIFTING;

      % Read from file and get the coordinates for Virt. Sphere
      DataV = fscanf(FileID2, '%f %f %f %f %f %f %f %f %f %f %f',11);
      DataV = DataV';
      if numel(DataV) ~= 0
        tVirtSph = DataV(1);
      end
    end
    end
  end
  end
end

%% External Concentration
if ShowExternalConcentration == 1
  for WallNumber = 1:ParcelNumberBottom
    ParcelBottomBorderWalls = vrnode(VirtualWorld,sprintf('Wall_%d',WallNumber+ParcelNumber+6-1));
    ParcelBottomBorderWalls.wallTrans  = 1;
  end
end

%% PAUSE IF MANUAL MODE
if Manual == 1
  pause
else
  pause(0.1)
end

%% DRAW THE OBJECTS
vrdrawnow;

%% RECORD THE FRAME
if Recording == 1
  image_capture = capture(Figure);
  for k = 1:NumberOfFrames
    writeVideo(VideoObject,image_capture);
  end
end

%% DRAW THE CELLS
if numel(Data)~=0
  % Ajust cell Transparency to have more vivid Virtual spheres if ShowContacts Mode ON
  if ShowContacts == 1;
    Transparency = 0.85;
  else
    Transparency = 0;
  end

  % Sphere 1
  SphA = vrnode(VirtualWorld,sprintf('SphA_%d',i2));
  SphA.SphTrans = Transparency;
  SphA.SphPosition = A;
  if CellColorGradient == 1
    SphA.SphColor = CellColor;
  else
    SphA.SphColor = CellColor(n,:);
  end
end
% Sphere 2
SphB = vrnode(VirtualWorld,sprintf('SphB_%d',i2));
SphB.SphColor = SphA.SphColor;
SphB.SphTrans = Transparency;
SphB.SphPosition = B;

% Cylinder
Cylinder = vrnode(VirtualWorld,sprintf('Cylinder_%d',i2));
Cylinder.CylScale = [1 norm(AB) 1];
Cylinder.CylColor = SphA.SphColor;
Cylinder.CylTrans = Transparency-0.1;
Cylinder.CylPosition = (A + B)/2;

if any(cross(DefaultCylinder,AB))
    Cylinder.CylRotation = [cross(DefaultCylinder,AB) acos((dot(DefaultCylinder,
AB))/(norm(DefaultCylinder)* norm(AB))));
else
    Cylinder.CylRotation = [0 1 0 1];
end;

% Display time and number of agents:
VirtualWorld.TEXT.string = {sprintf('Time%4.0f min/ Number of Agents%4u \n', t, i2)};
fprintf('----------------------------------- \n');
fprintf('Timeframe = %1.4f, n = %1.0f \n', t, i2);

% Update the time and the counter
i2 = i2 + 1;
tPrev = t;
end;

% end:
VirtualWorld.TEXT.string = {sprintf('Time%4.0f min/ Number of Agents%4u \n', t, i2-1)};
fprintf('----------------------------------- \n');
fprintf('Timeframe = %1.4f, n = %1.0f \n', t, i2-1);

if ShowContacts == 0;
    pause(1)
    disp('Last Record')
    if Recording == 1
        image_capture = capture(Figure);
        for k = 1:NumberOfFrames
            writeVideo(VideoObject,image_capture);
        end
    end
end

% Close Input and Output FILES
fclose(VideoObject);
close(FileID);
if ShowExternalConcentration == 1
    fclose(FileID3);
end
if ShowContacts == 1;
    fclose(FileID2);
end
disp('Animation recording OFF')
function [ColorMatrix] = CellColorsMode(ColorMode, MaxNumberOfBacteria, ID)

    if ColorMode == 1
        s = RandStream('mcg16807', 'Seed', 3); RandStream.setDefaultStream(s);
        ColorMatrix = rand(MaxNumberOfBacteria, 3);
    end

    if ColorMode == 2
        ColorMatrix = ones(MaxNumberOfBacteria, 3);
    end

    if ColorMode == 3
        ColorMatrix = ones(MaxNumberOfBacteria, 3);
        ColorMatrix(ID,:) = [1 0 0]; % for debug, change the number of the row
    end

    if ColorMode == 4
        MainColor = [0 0.5 0.85];
        ColorMatrix = repmat(MainColor, MaxNumberOfBacteria,1);
    end

    if ColorMode == 5
        MainColor = [0.85 0.85 0.85];
        ColorMatrix = repmat(MainColor, MaxNumberOfBacteria,1);
    end

    if ColorMode == 6
        s = RandStream('mcg16807','Seed',1); RandStream.setDefaultStream(s);
        ColorMatrix = rand(MaxNumberOfBacteria,3);
        PredefinedColors = [
            1.0000 0.0000 0.0000 % red
            0.0000 0.5000 0.0000 % green dark 1
            1.0000 0.5000 0.0000 % orange
            0.0000 0.5000 1.0000 % blue 3
            0.9950 0.9950 0.0000 % yellow 4
            0.7500 0.1000 0.8500 %
            1.0000 0.5000 0.1000 % white
            0.1378 0.3002 0.4785 % red dark
            1.0000 0.0000 0.7000 % pink 8
            0.2815 0.1126 0.8009 % light orange 9
            0.1000 0.5000 0.0000 % dark orange 10
            0.0000 0.1000 1.0000 % dark blue 11
            0.7240 0.8520 0.0000 % very light green 12
            0.3500 0.1000 0.4500 % Eggplant 5
            0.0000 0.1000 1.0000 % dark blue 11
            0.7240 0.8520 0.0000 % very light green 12
            0.3500 0.1000 0.4500 % Eggplant 5
            0.9000 0.9800 0.9800];
        ColorMatrix(1:size(PredefinedColors),:) = PredefinedColors;
    end
6.11.3. ProtoEnvironment.wrl

#VRML V2.0 utf8
#Created with V-Realm Builder v2.0
#Integrated Data Systems Inc.
#www.ids-net.com

Background {
  groundAngle[ 0.9, 1.5, 1.57 ]
 groundColor[ 0 0 0,
              0 0 0,
              0 0 0,
              0 0 0 ]
  skyAngle[ 0.1, 1.2, 1.57 ]
  skyColor[ 0 0 0,
             0 0 0,
             0 0 0,
             0 0 0 ]
}

WorldInfo {
  info ""
  title "Escherichia coli"
}

Transform {
  translation 0 -4 0
  children Billboard {
    axisOfRotation 0 0 0
    children Shape {
      geometry DEF TEXT Text {
        string "Time"
        fontStyle FontStyle {
          family "SANS"
          size 0.25
        }
      }
    }
  }
}


6.11.4. ProtoSphere.wrl

#VRML V2.0 utf8

#Created with V-Realm Builder v2.0
#Integrated Data Systems Inc.
#www.ids-net.com

WorldInfo {}

PROTO Sph []
   exposedField SFVec3f SphScale 1 1 1
   exposedField SFVec3f SphPosition 0 0 0
   exposedField SFRotation SphRotation 0 0 1 0
   exposedField SFCOLOR SphColor 0.5 0.5 0.5
   exposedField SFFloat SphTrans 0

{}  
DEF Body Transform {
   translation IS SphPosition
   rotation IS SphRotation
   scale IS SphScale

   children DEF Sph Transform {
      translation 0 0 0
      children Shape {
         appearance Appearance {
            material Material {
               ambientIntensity 0.2
               diffuseColor IS SphColor
               emissiveColor 0 0 0
               shininess 0.8
               specularColor 0.444226 0.723576 0.84
               transparency IS SphTrans
            }
         }
      }
      geometry Sphere {
      }
   }
}

}

}
6.11.5. ProtoCylinder.wrl

#VRML V2.0 utf8

#Created with V-Realm Builder v2.0
#Integrated Data Systems Inc.
#www.ids-net.com

WorldInfo {
}
PROTO Cyl [
  exposedField SFVec3f CylScale 1 1 1
  exposedField SFVec3f CylPosition 0 0 0
  exposedField SFRotation CylRotation 0 0 1 0
  exposedField SFCColor CylColor 0.5 0.5 0.5
  exposedField SFFloat CylTrans 0
]

{ DEF Body Transform {
    translation IS CylPosition
    rotation IS CylRotation
    scale IS CylScale
    children DEF Cyl Transform {
      translation 0 0 0
      children Shape {
        appearance Appearance {
          material Material {
            ambientIntensity 0.2
            diffuseColor IS CylColor
            emissiveColor 0 0 0
            shininess 0.8
            specularColor 0.444226 0.723576 0.84
            transparency IS CylTrans
          }
        }
        geometry Cylinder {
          height 1
          radius 0.25
          side TRUE
          top FALSE
          bottom FALSE
        }
      }
    }
  }
}

81
#VRML V2.0 utf8

#Created with V-Realm Builder v2.0
#Integrated Data Systems Inc.
#www.ids-net.com

WorldInfo {
}

PROTO Wall [
    exposedField SFVec3f WallScale 1 1 1
    exposedField SFVec3f WallPosition 0 0 0
    exposedField SFRotation WallRotation 0 0 1 0
    exposedField SFCOLOR WallColor 0.84 0.84 0.84
    exposedField SFFloat WallTrans 0
]

{ DEF Body Transform {
    bboxSize 0 0 0
    translation IS WallPosition
    rotation IS WallRotation
    scale IS WallScale
    children DEF Wall Transform {
        translation 0 0 0
        children Shape {
            appearance Appearance {
                material Material {
                    ambientIntensity 0.2
                    diffuseColor IS CylColor
                    emissiveColor 0 0 0
                    shininess 0.8
                    diffuseColor IS WallColor
                    transparency IS WallTrans
                }
            }
        }
    }
}

geometry Box {
    size 1 1 1
}
]