CellSim3D: GPU accelerated software for simulations of cellular growth and division in three dimensions

Madhikar, Pranav; Åström, Jan; Westerholm, Jan; Karttunen, Mikko

Published in:
Computer Physics Communications

DOI:
10.1016/j.cpc.2018.05.024

Published: 01/01/2018

Document Version
Accepted author manuscript

Document License
CC BY-NC-ND

CellSim3D: GPU Accelerated Software for Simulations of Cellular Growth and Division in Three Dimensions

Pranav Madhikar[a], Jan Åström[b], Jan Westerholm[c], Mikko Karttunen[d]

[a]*Department of Mathematics and Computer Science & Institute for Complex Molecular Systems, Eindhoven University of Technology, 5600 MB, Eindhoven, Netherlands*
[b]*CSC Scientific Computing Ltd, Kägelstranden 14, 02150 Esbo, Finland*
[c]*Faculty of Science and Engineering, Åbo Akademi University, Vattenborgsvägen 3, FI-20500, Åbo, Finland*
[d]*Department of Chemistry and Department of Applied Mathematics, Western University, 1151 Richmond Street, London, Ontario N6A 5B7, Canada*

Abstract

We present a new open source software package *CellSim3D* for computer simulations of mechanical aspects (that is, biochemical details are not accounted for) of cell division in three dimensions. It is also possible to use the software in the mode with cell division and growth turned off which allows for simulations of soft colloidal matter. The code is based on a previously introduced two dimensional mechanical model for cell division which is extended to full 3D. *CellSim3D* is written in C/C++ and CUDA and allows for simulations of 100,000 cells using standard desktop computers.

Keywords: Molecular dynamics, GPU, CUDA, cell division, soft colloids

PROGRAM SUMMARY

Program Title: CellSim3D version 1.0
Licensing provisions: GPLv2
Programming language: C/C++, CUDA, Python
Nature of problem: Mechanical 3-dimensional model for cell division and soft colloidal matter.
Solution method: Representation of cells as elastic three dimensional spheres with elastic forces, friction, repulsion, attraction and osmotic pressure. Integration of the equations of motion using the velocity-Verlet method from dissipative particle dynamics. Cells with volumes higher than a threshold can divide. Cell division can also be turned off thus allowing for simulations of soft colloidal matter.

Additional comments: Software web site: https://github.com/SoftSimu/CellSim3D

1. Introduction

The behavior of cells, their influence on their own environments, and the feedback of the environment onto the cells themselves have been of extreme interest and importance in biology and biophysics. In addition, there is strong evidence that suggests that the *mechanical* properties such as texture and stiffness can affect cell function and development[1–11]. Cellular migration, a vital part of many biological processes[12,13] such as embryonic morphogenesis[14–16], is yet another example of a mechanically affected aspect of cell behavior. Another beautiful example of the importance of mechanical forces in cell division is provided by Burton and Taylor who measured traction forces and showed how they localize during cell division in an elegant experiment[17].

Many experimental techniques have been used to study the cellular response to mechanical stimuli[18–20]. One needs to understand how the mechanism of mechanical stimulus is transformed into biochemical responses and vice versa[18] which requires the measurement of the distribution of forces at the molecular level. This is not possible to do with current experimental techniques. Furthermore, simulations allow testing many different conditions at a much lower cost. Finally, it is difficult to separate mechanical stimuli from biochemical stimuli in the lab. Computational methods eliminate the biochemical noise that exists in these types of systems. There has been a considerable effort to develop computational models[21–25] that can focus on the mechanical behavior of cells without adressing any of the complex biochemical processes that occur during cell division and growth. This is also the case with the current model: it focuses purely on mechanical behavior. We would also like to note that there are mathematical and computational models that focus on the biochemical aspects (and ignore mechanical aspects). Such models include phase field-based approaches such as the one by Howard et al.[26] and coupled partial differential equations, e.g., the recent one by Shtylla[27].

In general, mechanical models can be at the continuum level, where interaction functions are used to approximate bulk properties, or discrete cell based (or agent based) two and three-dimensional models. Continuum models use local interaction functions to simulate tissue, rather than collections

of cells. In these kinds of models, cell behavior is abstracted into functional forms containing expressions for density, growth rate, death rate, and inter-cellular interaction strength[28,29]. Continuum models have been used to successfully study tissue growth and topological rearrangements[28–31]. Unfortunately, due to their macroscopic nature, information regarding inter-cellular interactions is lost.

Numerous agent based methods, based on particle and lattice models, have also been proposed to address the mechanical aspects of cell division at a more detailed level. Examples include Delaunay Object Dynamics (DOD)[32–35], Vertex models[36–40], and lattice based models such as the cellular Potts Model[41–45], which itself can be extended to phase-field models[24,46].

It is surprising that despite the fundamental importance of cell division, and the influence of mechanics on it, there are very few software packages available to study it. This is drastically different from fields such as materials and biomaterials research. To the best of our knowledge, there are no freely or commercially available packages comparable to the one presented in this paper. Current free software packages are mostly based on the cellular Potts model, such as CompuCell3D[47] and CompuCell[48]. There is also the two dimensional Cellular Potts Model Library called Tissue Simulation Toolkit based on the work of Graner and Glazier[42] (available at https://sourceforge.net/projects/tst/), and cellGPU by Sussman[49]. Other methods are also available, for example LBICell that uses elastic polygons and the immersed boundary method[50], and the agent-based package CellSys by Hoehme and Drasdo[51] that is available as a free binary executable for non-commercial use. Commercial software includes the so-called Cell Division Program based on the work of Pyshnov[52]. Interested readers are referred to several recent reviews of computational models[53–55].

In this work, we present a software package named *CellSim3D* for simulating the mechanical aspects of cell division computationally. Our fully three dimensional approach is based on the 2D models originally described in[22,56]. The mechanical properties of cells can be related to the model parameters as discussed in Section 2. In our previous work using the 2D cell division model[22], the details of the parameter mapping between the model and real experimental systems were shown and verified. In particular, we showed that the model can accurately and spontaneously reproduce cell packing topologies similar to that in the *Drosophila*[57] wing disc, thus providing a proof of correctness of the general approach. Importantly, the user can easily change these parameters if a different mapping is required. In addition to

the simulation code, the software package includes a number of analysis tools as described in Section 3.1.5.

CellSim3D is able to simulate the organic growth of tissue from a single cell (starting from more than one cell is supported as well) into tissue with up to approximately 10^6 cells in 3D. Since the focus be on the mechanical aspects of cell behavior, the *CellSim3D* force-field, detailed below, assumes that all biochemical parameters are kept the same for all cells. Other terms that account for biochemical differences can be added later.

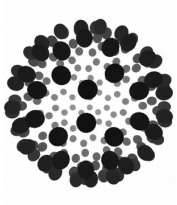
In brief, the *CellSim3D* code enables the simulation of the following, purely mechanical aspects of cell behavior and cell division:

1. The manner in which, and the strength with which, cells interact with each other and their environment mechanically. The physical origin of this is their membranes with inter-membrane interactions.
2. The mechanical effects of a cell's growth and division on its neighbors.
3. The mechanical response of a cell to external stimuli (the external stimulus itself need not be mechanical).
4. Cell migration: the mechanical basis within the cell, and the mechanism used to move on a substrate or in an external matrix.
5. The interplay between all of the above, including between cells of different type with different mechanical parameters.

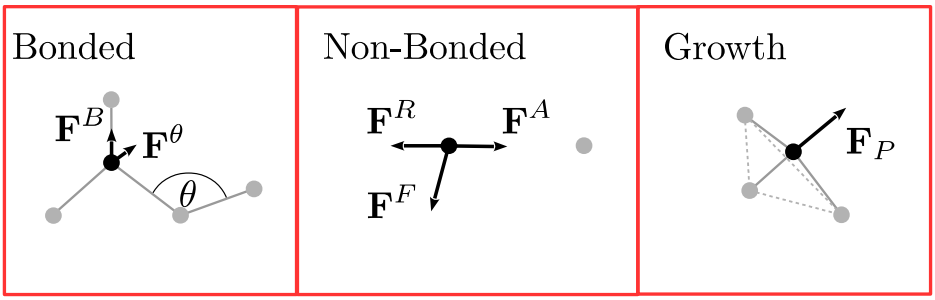
2. Model and Methods

The model presented here extends the model introduced in Ref.[22] from 2D to 3D. Each cell is modeled as a connected network of nodes, see Figure 1(A). This model was proposed to simulate a variety of cell behaviors. Intracellular forces are modeled with simple damped Hookean springs, each node being connected to three nearest neighbors, and an internal pressure. Inside the tissue, each node experiences forces that arise from intercellular interactions: adhesion and repulsion between cells, friction between cells, and friction between cells and the intercellular medium. The shapes of the simulated cells evolve according to the interplay between internal pressure, contractile forces along cell boundaries, and inter-cellular interactions. This allows for studying the dynamics of cell membranes, and their interactions, with higher spatial and temporal resolution.

With the above, the force field used in the simulations is given by



(A) C180 fullerene structure used in *CellSim3D*.



(B) *CellSim3D* force-field.

(C) Cell growth and division

Figure 1: (A) The geometry of the cell, which is taken from the C180 fullerene. (B) Schematic showing the parts of the force-field used in this model. Bonded forces are defined between nearest neighbor nodes within the cell (fullerene), the black dot is the node being considered and the gray are its neighbors. Non-bonded interactions are between nodes belonging to different cells, the black dot represents a node in the cell being considered and the gray dot a node belonging to another nearby cell. Cell growth is modeled by an internal pressure normal to the surface of the cell $\mathbf{F}_P$. (C) The various stages of division. The figures show snapshots from a simulation of a single cell. Cell division is done with a division plane oriented randomly and such that the cell is divided symmetrically, to model mitotic division[58].

$$\mathbf{F} = m\ddot{\mathbf{r}} = \mathbf{F}^B + \mathbf{F}^\theta + \mathbf{F}^P + \mathbf{F}^R + \mathbf{F}^A + \mathbf{F}^F, \tag{1}$$

where $\mathbf{F}^B$ is the damped Hookean bonding force between neighboring nodes and is related to the elasticity of cellular membranes. $\mathbf{F}^\theta$ is the angle force which preserves curvature, $\mathbf{F}^P$ is the force due to the cell's internal pressure coming from the osmotic pressure within cells. $\mathbf{F}^R$ is the repulsive force between different cells, $\mathbf{F}^A$ is the attractive force between different cells, together they approximate the inter-membrane interaction between membranes. $\mathbf{F}^F$ is the friction term, which itself is decomposed into viscous drag due to the medium (e.g. water) and inter-membrane friction.

For simplicity, harmonic potentials are used to approximate both the bonded interactions between nodes of the same cell and non-bonded interactions between nodes of different cells. This leads to simple definitions of the various forces in Equation 1. If necessary, these may be changed to other terms in a straightforward manner. Each node is bonded to three neighboring nodes with a bonding force given by

$$\mathbf{F}_i^B = \sum_{j=1}^{3} k^B \hat{\mathbf{b}}_{ij} \left(R_{ij} - R_o\right) - \gamma_{int}\mathbf{v}_{ij},$$

where k^B is the bonding spring constant, R_o is the equilibrium bond length, and γ_{int} is a term that damps the oscillation of the bonds. The angle force is given by

$$\mathbf{F}^\theta = -\nabla\left(\frac{1}{2}k_\theta(\theta - \theta_o)^2\right), \tag{2}$$

which is summed over the contributions due to all of the angles that the node is a part of. k^θ is the angle spring constant and θ_o is the equilibrium angle (taken from a C180 fullerene structure) formed by the bond vectors between bonded neighbors.

The interactions between different cells are governed by short-range non-bonded interactions. The repulsive part is given by

$$\mathbf{F}^R = \begin{cases} k^R\left(R - R_o^R\right)\hat{\mathbf{r}}_j & \text{if } R < R_o^R, \\ 0 & \text{if } R \geq R_o^R \end{cases} \tag{3}$$

and

$$\mathbf{F}^A = \begin{cases} -k^A\left(R - R_o^A\right)\hat{\mathbf{r}}_j & \text{if } R < R_o^A, \\ 0 & \text{if } R \geq R_o^A \end{cases} \tag{4}$$

defines the attractive component of intermembrane interactions. R_o^R and R_o^A are cutoff lengths. Together, $\mathbf{F}^A$ and $\mathbf{F}^R$ approximate inter-membrane interactions. K^R, K^A are the repulsive and attractive spring constants respectively, and R^R, R^A are the attractive and repulsive spring equilibrium lengths. Normally, $R^R < R^A$ and $K^R \gg K^A$. Cell-cell repulsion forces prevent a cell from occupying the space occupied by another cell, while adhesion forces maintain the integrity of tissues. In real cells, adhesion forces originate from adhesive bonds formed between Cell Adhesion Molecules (CAMs) on the surface of neighboring cells[59–62]. There is a wide variety of such adhesion molecules including Integrins, Selectins, Cadherins, and other proteins categorized in the IgSuperfamily[61,62]. In the simulated cells, each node on the boundary is an adhesive site, meant to approximate average CAM behavior. We would also like to add that it would be possible to make adhesion dynamic. For that, the functional form for adhesion needs to be determined. This is likely to be included in a future update.

Finally, the growth force is defined as

$$\mathbf{F}^P = PS\hat{\mathbf{n}}, \tag{5}$$

$\hat{\mathbf{n}}$ is the normal to the surface of the cell at the node in question. The surface is defined as the plane in which the three bonded neighbors lie. P is the internal pressure of the cell and S is a unit element of the surface area, PS is the magnitude of the force causing growth in the cell volume, while $\Delta(PS)$ is the growth rate.

Living cells are known to regulate their shape, growth and movement by modulating their internal pressure[63]; movement is regulated in combination with modifying cell cortex properties[64,65]. Internal hydrostatic pressure is balanced by inward forces generated by contractile forces in the actomyosin cortex on the periphery of cells[63]. Stewart et al.[63] showed that this pressure may be a result of changes in osmolyte concentrations in the cytoplasm and the contractility of the cell cortex. Animal cells are known to have a rounded shape in mitosis due to this pressure[57,66]. It is theorized that roundness of mitotic cells is a necessary geometric requirement for division[67–69]. In the model presented, each model cell is subjected to the same internal pressure P (Figure 1).

The internal pressure force $\mathbf{F}^P = PS\hat{\mathbf{n}}$ is balanced by spring forces $\mathbf{F}^B$ and angle forces $\mathbf{F}^\theta$. Combined, the spring-like forces represent contractile forces in the cell's cortex. The simulated cell is homogeneous in the sense

that all nodes and corresponding spring constants are identical, k^{θ}. This makes cells roughly spherical at equilibrium, which agrees with experimental characterizations of cells during mitosis[63,66–71].

The friction forces $\mathbf{F}^{F}$ consist of two components:

$$\mathbf{F}^{F} = \mathbf{F}^{F,m} + \mathbf{F}^{F,e}.$$

Cells in the current model move past each other while the tissue develops. As they move, they experience friction forces due to the medium and resistance of the cytoplasm. It is defined as

$$\mathbf{F}^{F,m} = -\gamma_m \mathbf{v}.$$

Cells also experience an effective friction due to the interactions with other cells. Specifically, when cells move past each other, the ith node of one cell slides along jth node of another with the relative velocity $\mathbf{v}_{ij} = \mathbf{v}_i - \mathbf{v}_j$. If $\mathbf{v}_{ij}^{\tau}$ is the component of the relative velocity tangential to the surface of the cell containing i, then the force on i is approximated as a damping force acting on the node i given as

$$\mathbf{F}_{ij}^{F,e} = -\gamma_{ext} \mathbf{v}_{ij}^{\tau} \tag{6}$$

for $R_{ij} < R_o^A$. The magnitude of the friction coefficient γ_{ext} can be related to the extent of cellular rearrangement during tissue growth. This term was borrowed from the DOD model[32–35].

2.1. Demonstration: Simulation of tissue growth

Simulated tissue starts as a single cell, or a collection of cells, which grow due to an increase in cellular pressure. Energy enters the system this way. See Figure 2 for simulation snapshots showing the formation of 3D tissue and an epithelium. A gradual increase in the internal pressure $\Delta(PS)$, leads to an increase in cell volume $V = V_o + \Delta V$. Once a cell exceeds the threshold value, V^{div}, it is divided into two new cells by a division plane and new nodes are introduced on both sides of the plane such that they form two deformed new cells. Thus, mass is input into the system. As an initial approximation, cytokinesis is assumed to be instantaneous and always such that symmetric daughters are formed to simulate mitosis. Figure 3 shows how new cells are created.

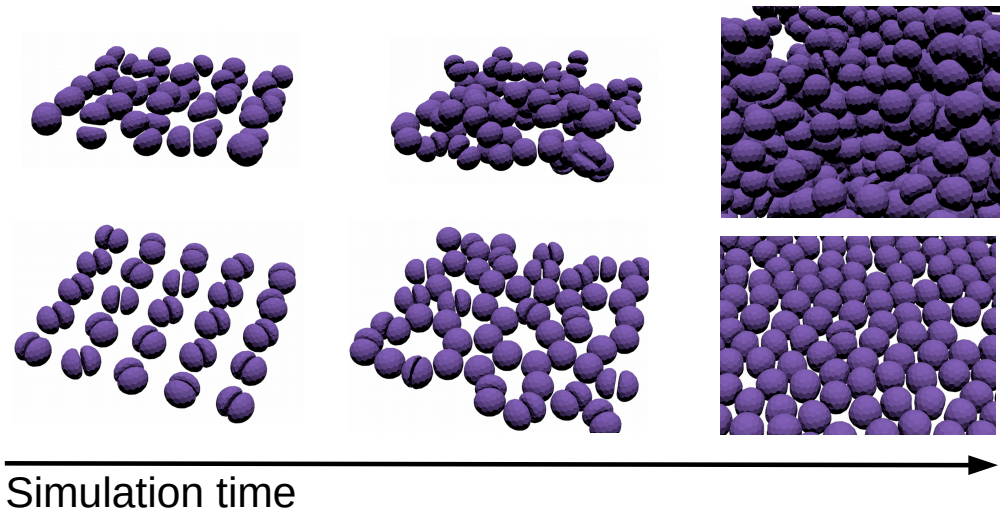


Figure 2: Snapshots from a simulation of tissue growth. The top panel shows the growth of three dimensional tissue and the bottom panel is the growth of an epithelium. The epithelium is created by adding confining walls above and below the system and making the cells divide in a plane perpendicular to the confining walls.

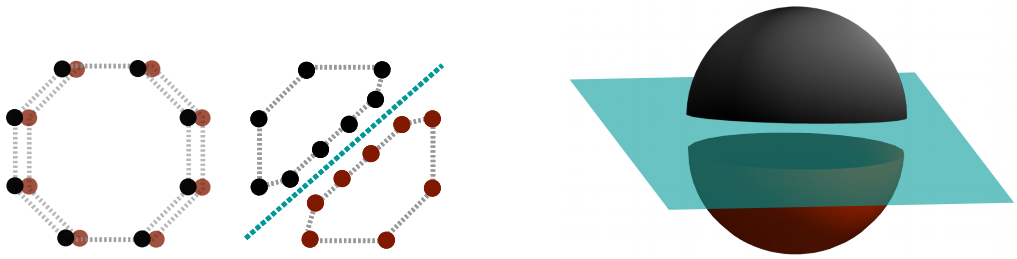


Figure 3: Sketch of the cell division algorithm. Left: The cell division algorithm shown in 2D for simplicity. First a cell large enough for division is chosen, and its nodes are copied (black and red). Then a randomly oriented division line is chosen (cyan) such that it divides the cell in half symmetrically. The corresponding nodes of the two new cells are deformed such that the nodes that would be in contact are laid along the division line. This leaves two new cells. Right: The same procedure is carried out in 3D, except using a division plane instead of a line. New cells are shown in dark red and black, the division plane is in cyan.

3. *CellSim3D* Software

The *CellSim3D* full source code is available under the GNU General Public License version 2 (GPLv2)[72] at https://github.com/SoftSimu/CellSim3D. The software is written in standard C/C++ and is accelerated with CUDA[73] to run on a single NVIDIA GPU. The algorithms are easily portable to OpenCL and/or MPI. The current limitation of a single GPU is not a problem since modern GPUs have enough memory to simulate up to 100,000 cells easily. Purely from a memory use standpoint, most modern mid-range GPUs can store up to 10^5 cells (such as the GTX 970 or the GTX 980) and high end GPUs (such as the GTX 1080Ti, GTX TITAN, or TESLA devices) can easily handle 10^6. As more and more global memory becomes available on hardware, even larger systems can be simulated in the future. 100,000 cells can be stored in the memory of a computer configured with the a GTX980 GPU (4GB of RAM); simulating a system of up to approximately 12,000 cells required approximately 2.5 hours.

CellSim3D has been tested on the following GPUs: GTX 760, GTX 780, GTX 980, GTX 1080Ti, GTX Titan Xp, GTX 960M, on systems configured with Intel CORE i7 and i5 CPUs (CPU architecture is of no consequence). The software has been successfully compiled with CUDA versions 5.5 to 9.1 — with the corresponding supported gcc versions (see the CUDA Toolkit Documentation[73]).

Communication and synchronization are known to be bottlenecks when accelerating any kind of computations with GPUs due to the higher latency of communication between host RAM (Random Access Memory) and GPU RAM. Therefore, as much of the computations as possible are done entirely on the GPU. The *CellSim3D* force field (Eq. 1) contains only short-range potentials. Thus, the entire potential and force calculations can be done on the GPU only, greatly minimizing the need for communication. Neighbor node list generation and most of the cell division algorithm are also carried out entirely on the GPU. Neighbor lists are generated with a simple algorithm that subdivides the simulation box into sub-boxes in parallel. Which cells are in which sub-boxes are also calculated in parallel. This information is then used to calculate a per-node neighbor list on the fly. The latter is done in parallel on the GPU.

Thanks to the above optimizations, most of the computations are done in CUDA only, with minor host code in C/C++ that controls execution and handles data input and output. Therefore, normal workstations with modest

mid-range CPUs may be used with *CellSim3D*.

3.1. Implementation

3.1.1. Core Simulator and system requirements

Since most of the heavy computations are carried out on the GPU, the requirements on the rest of the hardware configuration is not stringent. These modest hardware requirements allow the study of interesting systems with relative ease on a single node with a single GPU. Support for multiple nodes or multiple GPUs is not implemented currently. Multiple GPU support is planned for future releases.

The memory use is constant over the simulation and can be configured with the input JSON file, see Fig. 4. This is to avoid repetitive allocation/deallocation of memory on the host and on the GPU. Memory is allocated for a maximum number of cells for the simulator. This number is typically much bigger than needed in the actual simulation. One can allocate enough memory for 100,000 cells on a single modern GPU such as the GTX 980, even low-end GPUs such as the GTX 960M can hold 50,000 cells. 100,000 cells take about 4GB of memory. As an example, systems of 10,000 cells can be safely simulated on GPUs with less than 1GB of memory and typically take less than 45–60 minutes of wall clock time.

The simulator code supports any version of CUDA newer than 5.0. That is, it can take advantage of optimizations in newer versions of CUDA (up to CUDA 9.1 is supported) while still being more-or-less compatible with most NVIDIA GPUs today. *CellSim3D* can be easily compiled with the `makefile` provided. Only Linux is supported. Any corresponding gcc compiler that is required by the CUDA version may be used. For example, CUDA 9.1 requires gcc 5.3.1 on Ubuntu 16.04 running on x86_64 systems. Refer to the CUDA Toolkit Documentation for details[73].

CellSim3D only depends on the `jsoncpp` library, which is pre-packaged with *CellSim3D* for ease of use. No other libraries are needed by the simulator. Some Python libraries are required for the analysis tools, which are outlined in Section 3.1.5.

Minimum System Requirements:

- NVIDIA GPU of compute capability of 3.5 or higher
- 1 GB of GPU memory (4GB recommended)
- CUDA 5.5 or higher (later versions recommended)

- Python 3.5, with required libraries
- Blender 2.7 or higher (only needed for visualization)

3.1.2. Software Structure and Use

3.1.3. Source files

There are two main types of files that are included in the package. The first, and most vital, are the source files that contain all the code. The second are data files that contain information such as cell geometry and bonding.

Below are short summaries of some of the source files.

- `GPUBounce.cu` is the main source file that outlines the flow the simulation.
- `Propagate.cu` contains all the code pertaining to calculating and integrating the forces in the system.
- `Volume.cu` calculates cell volumes in parallel on the GPU, and flags them for division if needed.
- `CellDivision.cu` performs the actual cell division. A simple geometrical argument is made when dividing the cells, see Figure 3.

Figure 4 shows a flowchart of the basic algorithm implemented in the simulator. Once compiled, the simulator is configured with a input JSON file, see Fig. 4. This file follows the standard JSON format, with some additions for C-style comments. It contains all the force field parameters. A sample input file named `inp.json`, which contains many informative comments, is included with the software package.

The bounding box of the simulated cells is subdivided into a number of smaller sub-boxes on the GPU, which are used to quickly calculate the cells that are within the same sub-box. This is done efficiently and in parallel on the GPU. Then, during force calculation, the nodes from other cells within range of each node are calculated in parallel for the inter-membrane interactions.

3.1.4. Integration of the equations of motion

The force field outlined in Section 2 contains a friction term. After testing different approaches, a modified velocity-Verlet integrator, the so-called

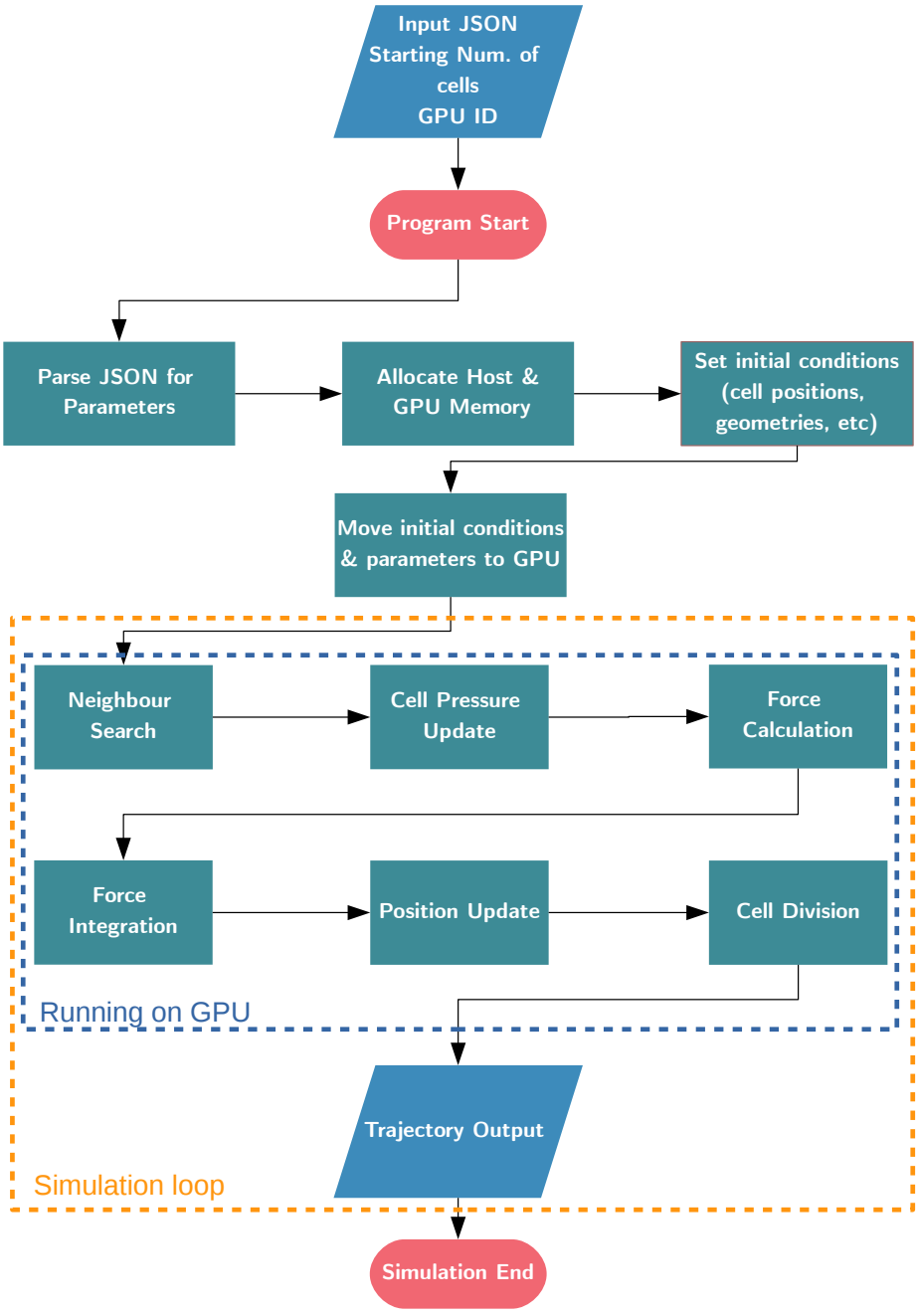


Figure 4: Simulation flowchart. All of the parameters used in the simulation are set in a JSON (JavaScript Object Notation) file that is given as an argument to the simulator (the program distribution includes a sample file). The GPU id (assigned by the system) and the initial number of cells are also program arguments.

DPD-VV (Dissipative Particle Dynamics velocity-Verlet) of Besold et al.[74] turned out to be stable and fast. DPD-VV has been described and tested in detail in Refs.[74,75]. The algorithm is listed in Table 1.

$$
\begin{array}{ll}
(1) & \mathbf{v}_i \leftarrow \mathbf{v}_i + \frac{1}{2m}\left(\mathbf{F}_i^C \Delta t + \mathbf{F}_i^D \Delta t + \mathbf{F}^R \sqrt{\Delta t}\right) \\
(2) & \mathbf{r}_i \leftarrow \mathbf{r}_i + \mathbf{v}_i \Delta t \\
(3) & \text{Calculate } \mathbf{F}_i^C\{\mathbf{r}\},\ \mathbf{F}_i^D\{\mathbf{r},\mathbf{v}\},\ \mathbf{F}_i^R\{\mathbf{r}\} \\
(4a) & \mathbf{v}_i^o \leftarrow \mathbf{v}_i \frac{1}{2m}\left(\mathbf{F}_i^C \Delta t + \mathbf{F}_i^R \sqrt{\Delta t}\right) \\
(4b) & \mathbf{v}_i \leftarrow \mathbf{v}_i^o + \frac{1}{2m}\mathbf{F}_i^D \Delta t \\
(5) & \text{Calculate } \mathbf{F}_i^D\{\mathbf{r},\mathbf{v}\}
\end{array}
$$

Table 1: DPD-VV integration algorithm[74]. $\mathbf{F}^C$ are the conservative parts of the force-field, these include the bonded and non-bonded interactions, $\mathbf{F}^D$ is the dissipative part — the friction part, and $\mathbf{F}^R$ is the random component of the force-field, which is zero in the current *CellSim3D* force-field (Eq. 1). The positions and velocities are given by $\mathbf{r}_i$ and $\mathbf{v}_i$, respectively, m is the particle mass and Δt the time step.

3.1.5. Input/Output and Analysis Tools

The only input required for the simulation is a simple JSON[76] file with extensions allowing for C-style comments. The input file is parsed with the `jsoncpp` library[77]. This file sets the initial conditions for the system.

The simulator writes the output of the simulation to ASCII files, and the trajectories of the nodes are written to a custom binary format that is flexible enough to allow for a variable number of nodes in a single time step. The binary file can be read and processed in python with a packaged module named `celldiv.TrajHandle`. This is a simple interface between the trajectory output from the simulator and NumPy[78] arrays in python3. In this way, all the power of the numerical libraries available through Python can be applied easily on any of the data. All the tools, including a movie renderer, are wrappers around this interface. Some other data, such as cell volumes and counts, is output in ASCII format for ease of use. Future releases will be using the HDF5[79] format for efficient, consistent Input/Output.

This software also comes with other tools, in the form of python3 scripts, that can be used to easily analyze the data produced. python2 is not supported.These tools require the latest versions of the following python libraries: NumPy[78], Scipy[80], matplotlib[81], tqdm[82], and pandas[83]. A brief summary file called `requirements.txt` is provided for easily building a suitable python environment.

3.1.6. Scaling

The scaling of this simulator is complicated to measure as the number of nodes in the simulation increases non-linearly in time. However, we can measure the time consumption as a function of growth rate $\Delta(PS)$. Figure 5 shows a measurement of the time it took to run a simulation of 150,000 time steps.

3.2. Limitations

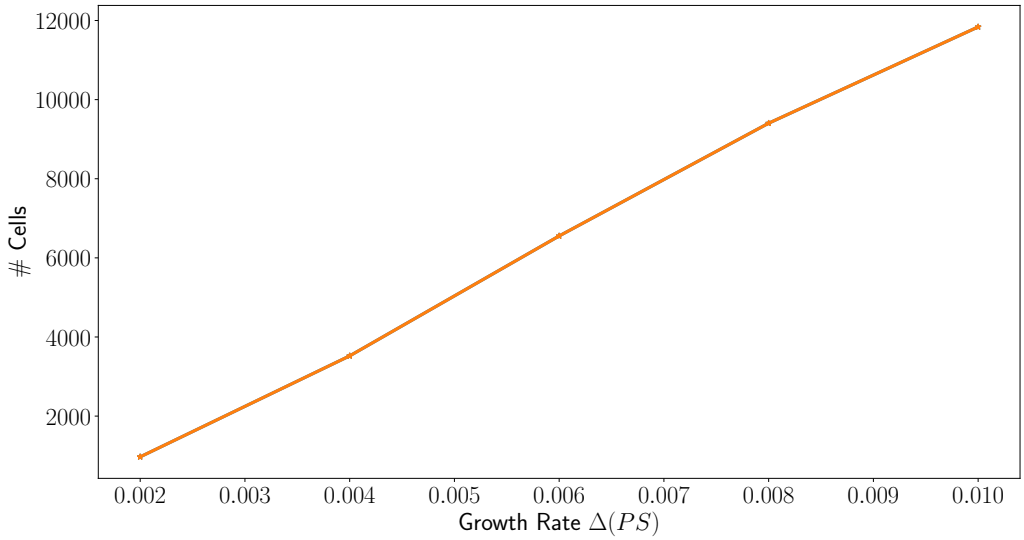
The simulator only supports a currently hard-coded cell geometry, shown in Figure 1. This geometry is that of the C180 fullerene. It was chosen for ease of use and programming. However, this geometry may be changed to suit the needs of the system of interest. More geometries can trivially be added, however we find the default structure used for the moment is adequate for most needs. The equilibrium shape of C180 is spherical.

The focus of this simulator is to study the symmetrical division of cells. In biological terms, this corresponds to the production of two daughter cells identical to each other and the parent cell. Therefore, only this kind of division is currently supported. This is not the only kind of division that occurs in living tissue[84]. Asymmetric division can occur as well. Other modes of division are planned for future releases.

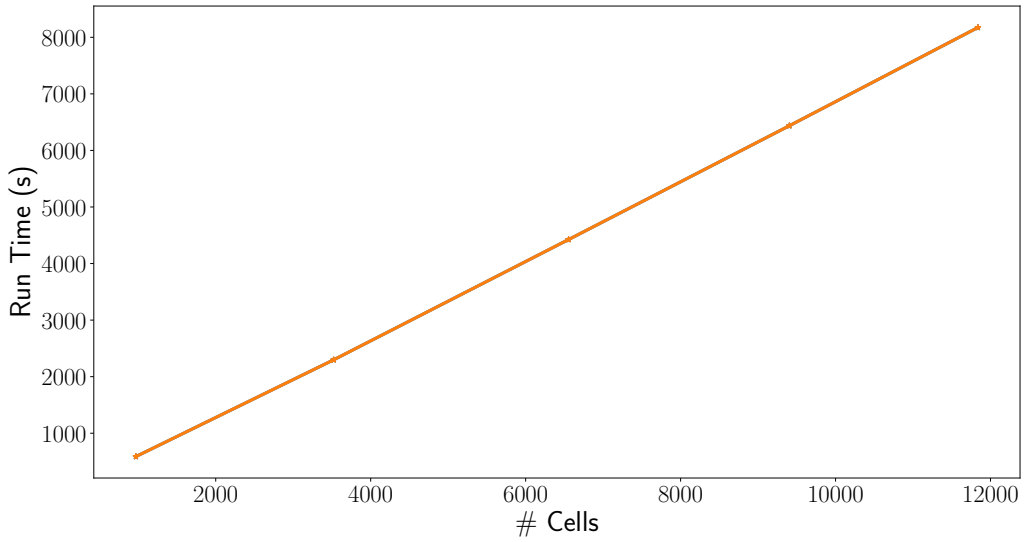
4. Summary

In this work, we have presented a new fully three dimensional software package named *CellSim3D* for computer simulations of cell division. The code is released under the GNU General Public License version 2 (GPLv2)[72] at https://github.com/SoftSimu/CellSim3D. It is written in standard C/C++ and accelerated with CUDA[73] to run simulations on a single NVIDIA GPU. Systems sizes of 100,000 cells are easily reachable with standard desktop hardware. The cell division model itself is an extension of the 2D mechanical model originally described in[22,56].

We would like to comment the formation of structures in terms of cell deformations. Cells in real *in vivo* systems display large, often asymmetric, deformations. This is also the case with the current model and was demonstrated already in connection with both the two-dimensional model with[22] and without[56] cell division. In particular, in Ref.[22] the polygonal distributions and mitotic index from simulations using the two dimensional model



(A)



(B)

Figure 5: Scaling at different growth rates. Simulations were run for 150,000 time steps in total, 100,000 with cell division enabled and 50,000 with cell division disabled (simulates growth, then equilibration). (A) Final number of cells as a function of growth rate ($\Delta(PS)$). (B) Simultion run time to reach the final number of cells shown in (A).

were compared with experimental data[57] and found to be in excellent agreement. The default structure for a cell in the current implementation is the fullerene C180, but this is can be easily changed by the user. This default structure was chosen since it is robust with respect to both physical and structural properties. In addition, it is possible to implement asymmetric division and this will be included in a future update.

It is also possible to turn off cell division. By doing that, it can be used to simulate three dimensional soft colloidal matter in spirit similar to what was done in two dimensions with the predecessor of this model[56]. Both open boundary conditions as well as confinement by walls can be used. A future update will include generalized fullerene structures[85] allowing for simulations of tubular and other structures which may be useful in simulations of colloidal matter. Growth of colonies of tubular bacteria is another example of such systems.

Acknowledgments

We would like to thank Björn Baumeier for many helpful discussions. MK would like to thank Natural Sciences and Engineering Research Council of Canada (NSERC) for financial resources. We also gratefully acknowledge the support of the NVIDIA Corporation for the GTX Titan Xp GPU used for developing and testing CellSim3D.

[1] C.J. Jen, S.J. Jhiang, H.I. Chen, Invited review: effects of flow on vascular endothelial intracellular calcium signaling of rat aortas ex vivo., J. Appl. Physiol. 89 (2000) 1657–1662.

[2] M. Sato, M.J. Levesque, R.M. Nerem, Micropipette aspiration of cultured bovine aortic endothelial cells exposed to shear stress, Arterioscler. Thromb. Vasc. Biol. 7 (1987) 276–286.

[3] M.J. Kuchan, J.A. Frangos, Shear stress regulates endothelin-1 release via protein kinase c and cgmp in cultured endothelial cells., Am. J. Physiol. 264 (1993) H150–H156.

[4] D.T. Scadden, The stem-cell niche as an entity of action, Nature 441 (2006) 1075–1079.

[5] D.A. Lee et al, Stem cell mechanobiology, J. Cell. Biochem. 112 (2010) 1–9.

[6] S. Battista et al, The effect of matrix composition of 3D constructs on embryonic stem cell differentiation, Biomaterials 26 (2005) 6194–6207.

[7] M. Akhmanova et al, Physical, spatial, and molecular aspects of extracellular matrix ofin vivoniches and artificial scaffolds relevant to stem cells research, Stem Cells International (2015) 1–35.

[8] D.A. Fletcher, R.D. Mullins, Cell mechanics and the cytoskeleton, Nature 463 (2010) 485–492.

[9] F. Gattazzo, A. Urciuolo, P. Bonaldo, Extracellular matrix: A dynamic microenvironment for stem cell niche, Biochim. Biophys. Acta - General Subjects 1840 (2014) 2506–2519.

[10] M. Votteler et al, Stem cell microenvironments - unveiling the secret of how stem cell fate is defined, Macromol. Biosci. 10 (2010) 1302–1315.

[11] A. Wade, A. McKinney, J.J. Phillips, Matrix regulators in neural stem cell functions, Biochim. Biophys. Acta - General Subjects 1840 (2014) 2520–2525.

[12] D.A. Lauffenburger, A.F. Horwitz, Cell migration: A physically integrated molecular process, Cell 84 (1996) 359–369.

[13] D. Hanahan, R.A. Weinberg, Hallmarks of cancer: the next generation., Cell 144 (2011) 646–674.

[14] R. Keller, M. Danilchik, Regional expression, pattern and timing of convergence and extension during gastrulation of xenopus laevis., Development 103 (1988) 193–209.

[15] M. Chuai et al, Cell movement during chick primitive streak formation, Dev. Biol. 296 (2006) 137–149.

[16] L.C. Butler et al, Cell shape changes indicate a role for extrinsic tensile forces in drosophila germ-band extension, Nat. Cell Biol. 11 (2009) 859–864.

[17] K. Burton, D.L. Taylor, Traction forces of cytokinesis measured with optically modified elastic substrata, Nature, 385 (1997) 450—454.

[18] H. Huang, Cell mechanics and mechanotransduction: pathways, probes, and physiology, AJP: Cell Physiology 287 (2004) C1–C11.

[19] P.A. Janmey, C.A. McCulloch, Cell mechanics: Integrating cell responses to mechanical stimuli, Annu. Rev. Biomed. Eng. 9 (2007) 1–34.

[20] M.M. Saunders, Mechanical testing for the biomechanics engineer: A practical guide, Synthesis Lectures on Biomedical Engineering 9 (2015) 1–276.

[21] A. Verdier, C. Chaviere, L. Preziosi, Cell Mechanics: From Single Scale-Based Models to Multiscale Modeling, Mathematical and Computationl Biology Series, CRC Press, Boca Raton, FL, 2010.

[22] A. Mkrtchyan, J.A. Åström, M. Karttunen, A new model for cell division and migration with spontaneous topology changes, Soft Matter 10 (2014) 4332–4339.

[23] D. Drasdo, S. Hoehme, M. Block, On the role of physics in the growth and pattern formation of multi-cellular systems: What can we learn from individual-cell based models?, J. Stat. Phys. 128 (2007) 287–345.

[24] M. Nonomura, Study on multicellular systems using a phase field model, PLoS ONE 7 (2012) e33501.

[25] B. Aigouy et al, Quantitative methods to study epithelial morphogenesis and polarity., Methods in cell biology 139 (2017) 121–152.

[26] M. Howard, A.D. Rutenberg, S. de Vet, Dynamic Compartmentalization of Bacteria: Accurate Division in *E. Coli*, Phys. Rev. Lett., 87 (2001) 278102.

[27] B. Shtylla, Mathematical modeling of spatiotemporal protein localization patterns in *C. crescentus* bacteria: A mechanism for asymmetric FtsZ ring positioning, J. Theor. Biol., 433 (2017) 8–20.

[28] J. Ranft et al, Fluidization of tissues by cell division and apoptosis, Proc. Natl. Acad. Sci. U.S.A. 107 (2010) 20863–20868.

[29] T. Bittig et al, Dynamics of anisotropic tissue growth, New J. Phys. 10 (2008) 063001.

[30] M. Ben Amar, C. Chatelain, P. Ciarletta, Contour instabilities in early tumor growth models, Phys. Rev. Lett. 106 (2011) 148101.

[31] S. Turner, Using cell potential energy to model the dynamics of adhesive biological cells, Phys. Rev. E 71 (2005) 041903.

[32] G. Schaller, M. Meyer-Hermann, Kinetic and dynamic delaunay tetrahedralizations in three dimensions, Comput. Phys. Commun. 162 (2004) 9–23.

[33] G. Schaller, M. Meyer-Hermann, Multicellular tumor spheroid in an off-lattice voronoi-delaunay cell model, Phys. Rev. E 71 (2005) 051910.

[34] M. Meyer-Hermann, Delaunay-object-dynamics: Cell mechanics with a 3D kinetic and dynamic weighted delaunay-triangulation, Curr. Top. Dev. Biol. (2008) 373–399.

[35] T. Beyer, M. Meyer-Hermann, Multiscale modeling of cell mechanics and tissue organization, IEEE Eng. Med. Biol. Mag. 28 (2009) 38–45.

[36] H. Honda, M. Tanemura, T. Nagai, A three-dimensional vertex dynamics cell model of space-filling polyhedra simulating cell behavior in a cell aggregate, J. Theor. Biol. 226 (2004) 439–453.

[37] R. Farhadifar et al, The influence of cell mechanics, cell-cell interactions, and proliferation on epithelial packing, Curr. Biol. 17 (2007) 2095–2104.

[38] L. Hufnagel et al, On the mechanism of wing size determination in fly development, Proc. Natl. Acad. Sci. U.S.A. 104 (2007) 3835–3840.

[39] A. Fletcher et al, Vertex models of epithelial morphogenesis, Biophys. J. 106 (2014) 2291–2304.

[40] D.M. Sussman et al, Soft yet sharp interfaces in a vertex model of confluent tissue, Phys. Rev. Lett. 120 (2018) 058001.

[41] J. Glazier, F. Graner, Simulation of the differential adhesion driven rearrangement of biological cells, Phys. Rev. E 47 (1993) 2128–2154.

[42] F. Graner, J. Glazier, Simulation of biological cell sorting using a two-dimensional extended potts model, Phys. Rev. Lett. 69 (1992) 2013–2016.

[43] A. Szabó, R.M.H. Merks, Cellular potts modeling of tumor growth, tumor invasion, and tumor evolution, Front. Oncol. 3 (2013) 87.

[44] A. Shirinifard et al, 3D multi-cell simulation of tumor growth and angiogenesis, PLoS One 4 (2009) e7190.

[45] R.M.H. Merks, J.A. Glazier, A cell-centered approach to developmental biology, Physica A 352 (2005) 113–130.

[46] B. Palmieri et al, Multiple scale model for cell migration in monolayers: Elastic mismatch between cells enhances motility, Sci. Rep. 5 (2015) 11745.

[47] M.H. Swat et al, Multi-scale modeling of tissues using compucell3d, in: Methods in cell biology, Vol. 110, Elsevier, Cambridge, MA, 2012, pp. 325–366.

[48] J.A. Izaguirre et al, Compucell, a multi-model framework for simulation of morphogenesis, Bioinformatics 20 (2004) 1129–1137.

[49] D.M. Sussman, cellGPU: Massively parallel simulations of dynamic vertex models, Comput. Phys. Commun. 219 (2017) 400–406.

[50] S. Tanaka, D. Sichau, D. Iber, LBIBCell: a cell-based simulation environment for morphogenetic problems, Bioinformatics 31 (2015) 2340–2347.

[51] S. Hoehme, D. Drasdo, A cell-based simulation software for multicellular systems, Bioinformatics 26 (2010) 2641–2642.

[52] M.B. Pyshnov, Topological solution for cell proliferation in intestinal crypt. i. elastic growth without cell loss, J. Theor. Biol. 87 (1980) 189–200.

[53] G.W. Jones, S.J. Chapman, Modeling growth in biological materials, SIAM Rev. 54 (2012) 52–118.

[54] P.V. Liedekerke et al, Simulating tissue mechanics with agent-based models: concepts, perspectives and some novel results, Computational Particle Mechanics 2 (2015) 401–444.

[55] F. Ziebert, I.S. Aranson, Computational approaches to substrate-based cell motility, npj Computational Materials 2 (2016) 16019.

[56] J.A. Åström, M. Karttunen, Cell aggregation: Packing soft grains, Phys. Rev. E 73 (2006) 062301.

[57] M.C. Gibson et al, The emergence of geometric order in proliferating metazoan epithelia, Nature 442 (2006) 1038–1041.

[58] Y. Imoto et al, The cell cycle, including the mitotic cycle and organelle division cycles, as revealed by cytological observations, Microscopy 60 (2011) S117–S136.

[59] F. van Roy, G. Berx, The cell-cell adhesion molecule e-cadherin., Cell. Mol. Life Sci. 65 (2008) 3756–3788.

[60] M.P. Stemmler, Cadherins in development and cancer., Mol. Biosyst. 4 (2008) 835–850.

[61] C.D. Buckley et al, Cell adhesion: More than just glue (review), Mol. Membr. Biol. 15 (1998) 167–176.

[62] G.M. Edelman, K.L. Crossin, Cell adhesion molecules: Implications for a molecular histology, Annu. Rev. Biochem. 60 (1991) 155–190.

[63] M.P. Stewart et al, Hydrostatic pressure and the actomyosin cortex drive mitotic cell rounding, Nature 469 (2011) 226–230.

[64] C. Roubinet, P.T. Tran, M. Piel, Common mechanisms regulating cell cortex properties during cell division and cell migration., Cytoskeleton 69 (2012) 957–972.

[65] T.S.P. Strangeways, Observations on the changes seen in living cells during growth and division, Proc. Roy. Soc. London 94 (1922) 137–141.

[66] L.P. Cramer, T.J. Mitchison, Investigation of the mechanism of retraction of the cell margin and rearward flow of nodules during mitotic cell rounding., Mol. Biol. Cell 8 (1997) 109–119.

[67] P. Kunda et al, Moesin controls cortical rigidity, cell rounding, and spindle morphogenesis during mitosis., Curr. Biol. 18 (2008) 91–101.

[68] S. Carreno et al, Moesin and its activating kinase slik are required for cortical stability and microtubule organization in mitotic cells., J. Cell Biol. 180 (2008) 739–746.

[69] P. Kunda, B. Baum, The actin cytoskeleton in spindle assembly and positioning., Trends Cell Biol. 19 (2009) 174–179.

[70] T. Lecuit, P.F. Lenne, Cell surface mechanics and the control of cell shape, tissue patterns and morphogenesis, Nat. Rev. Mol. Cell Biol. 8 (2007) 633–644.

[71] T. Lecuit, L. Le Goff, Orchestrating size and shape during morphogenesis, Nature 450 (2007) 189–192.

[72] Gnu general public license, version 2.
URL `https://www.gnu.org/licenses/old-licenses/gpl-2.0.en.html`

[73] NVIDIA, CUDA Toolkit Documentation.
URL `http://docs.nvidia.com/cuda/index.html`

[74] G. Besold et al, Towards better integrators for dissipative particle dynamics simulations, Phys. Rev. E 62 (2000) R7611–R7614.

[75] P. Nikunen, M. Karttunen, I. Vattulainen, How would you integrate the equations of motion in dissipative particle dynamics simulations?, Comput. Phys. Commun. 153 (2003) 407–423.

[76] Introducing json (javascript object notation).
URL `https://www.json.org/`

[77] jsoncpp - a c++ library for interacting with json.
URL `https://github.com/open-source-parsers/jsoncpp`

[78] S. van der Walt, S.C. Colbert, G. Varoquaux, The numpy array: A structure for efficient numerical computation, Comput. Sci. Eng. 13 (2011) 22—30.

[79] T.H.D.F. Group, Hierarchical Data Format, version 5, http://www.hdfgroup.org/HDF5/ (1997-2018).

[80] E. Jones et al, SciPy: Open source scientific tools for python (2001).

[81] J.D. Hunter, Matplotlib: A 2d graphics environment, Computing In Science & Engineering 9 (2007) 90–95.

[82] C. da Costa-Luis et al, tqdm/tqdm: tqdm v4.19.5 stable, http://dx.doi.org/10.5281/zenodo.1251290 (Dec. 2017).

[83] W. McKinney, Data structures for statistical computing in python, in: S. van der Walt, J. Millman (Eds.), Proceedings of the 9th Python in Science Conference, 2010, pp. 51–56.

[84] J.A. Knoblich, Asymmetric cell division: recent developments and their implications for tumour biology, Nat. Rev. Mol. Cell Biol. 11 (2010) 849–860.

[85] Y. Wang, S. Diaz-Tendero, M.A.F. Mart, Generalized structural motif model for studying the thermodynamic stability of fullerenes: from C60 to graphene passing through giant fullerenes, Phys. Chem. Chem. Phys., 19 (2017) 19646–19655.