Computer Physics Communications 185 (2014) 826–835

Contents lists available at ScienceDirect

Computer Physics Communications

journal homepage: www.elsevier.com/locate/cpc

Efficient and extendible class scheme for the combined reaction–diffusion of multiple molecular species

Sabrina Stella [a,*], Roberto Chignola [b,c], Edoardo Milotti [a,c]

[a] Department of Physics, University of Trieste – Via Valerio, 2 – 34127 Trieste, Italy
[b] Department of Biotechnology, University of Verona, Strada Le Grazie, 15, I-37134 Verona, Italy
[c] Istituto Nazionale di Fisica Nucleare – Sezione di Trieste, Italy

ARTICLE INFO

Article history:
Received 5 July 2013
Received in revised form
6 November 2013
Accepted 19 November 2013
Available online 25 November 2013

Keywords:
Biophysics
C++
Reaction–diffusion equation
Simulation

ABSTRACT

When dealing with large numbers of cells in biophysical simulations, it is important to properly manage the different substances that diffuse and react in and around cells. Although in an object-oriented programming environment it seems more natural to define cells as the basic objects, it turns out that individual substances are better suited to take this role. Here we describe the biophysical problem and our computational solution, and display the results obtained with a toy model. We find that the new implementation does not decrease performance and yet it leads to a much better structured and modular code. This will make more realistic programs with many molecular pathways much more modular and readily extendible.

1. Introduction

Simulations of the biochemistry of large aggregates of cells, like the one described in Refs. [1–4], require the solution of large reaction–diffusion problems. The individual cells – and their surrounding environments – act like small spatial compartments, with many chemicals flowing in and out [5]. Some of these chemicals are confined within cells, while others can pass the cell membranes only by the action of facilitated diffusion processes, mediated by membrane proteins. These spatial compartments act as small bioreactors, where chemicals react and determine cells' life and death. The diffusion equations for each chemical are naturally discretized on the geometrically and topologically disordered network defined by individual cells, as shown in Fig. 1. This disordered network of cells is an essential element of the simulation if we wish to include the biomechanics of cells as well as their biochemistry, and it brings with it important structural problems. As the cluster of cells grows and the neighbors of an individual cell change, so does the structure of the corresponding system of reaction–diffusion equations—the number of equations becomes larger and larger, and terms in each equation change (again, see Figs. 1 and 2). Moreover, the chemicals involved in the simulation are strongly coupled in those small, natural bioreactors that are the individual cells. Therefore in such an object-oriented code it is straightforward to define the cells themselves as the basic objects, and the masses of chemicals as attributes (internal data) of cells. This is in stark contrast to simulations like those that utilize cellular automata [6], or recast the whole process of tumor growth into fixed systems of equations [7], and which more closely resemble classical reaction–diffusion problems (see, e.g., [8]). Our interest in this problem is very practical, as we seek computational solutions in our project that aims to simulate small avascular solid tumors [1–5]. We have started this complex computational task with a basic description of the internal machinery of cells, however we aim at a gradual increase of the biological complexity, with the inclusion of many additional molecular circuits that are still missing from the present version of the program (Fig. 3).

In the C++ code that implements the model [3], cell objects are stored in turn in structures like vectors, or in STL containers like STL vectors or lists (or equivalently in Boost containers [9]), and the solution of the reaction–diffusion equations requires access to the internal data of the elements of the vectors or lists of cells—and possibly their storage into temporary vectors. Thus, although cells seem to be the most natural objects in this context, they also lead to a very cumbersome and ineffective solution of the reaction–diffusion equations—and increasingly so, as the number of molecular species, and thus of temporary vectors, grows in more detailed versions of the simulation programs.

Another important adverse effect on the code is the absence of modularity in the treatment of the different chemicals, that comes from the scalar nature of the data inside the cell objects.

* Corresponding author. Tel.: +39 0405583388.
E-mail address: sabrina.stella@ts.infn.it (S. Stella).

0010-4655/$ – see front matter © 2013 Elsevier B.V. All rights reserved.
http://dx.doi.org/10.1016/j.cpc.2013.11.014

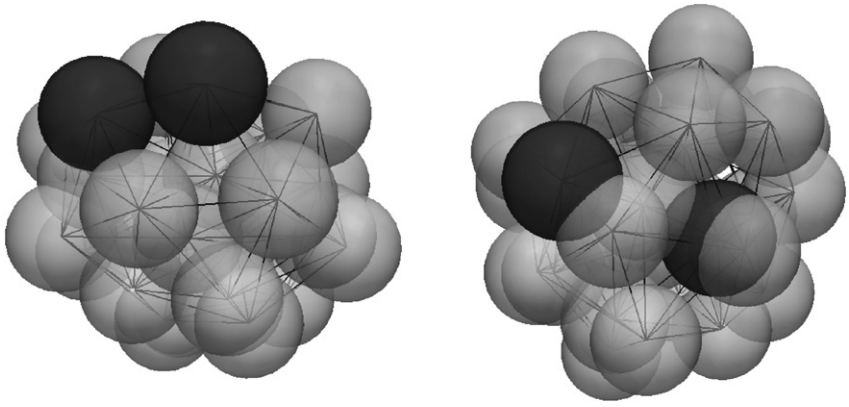

Fig. 1. Detailed view of cells, early in one simulation run of the program described in [3]. In these figures the cells are represented by spheres, and the run started with a single cell. These snapshots show the same aggregate after 100 hs of simulated time (left panel), and ten hours later (right panel). The cells are actively proliferating and there are 29 cells in the left panel, and 32 cells in the right panel. The straight segments represent the edges of the Delaunay triangulation of the cell centers, and they connect the center of each cell to its nearest neighbors; cell radii have been slightly reduced to better show the edges of the triangulation. As the aggregate of cells grows, cells migrate under the push of neighboring cells, and the network of connections changes. The two marked, darker cells are originally in contact (left panel), but they move to different positions and are no longer in contact after ten hours of simulated time. The different number of cells and the different connectivity mean that the differential system that describes reaction–diffusion in the cell cluster changes from one snapshot to the next.

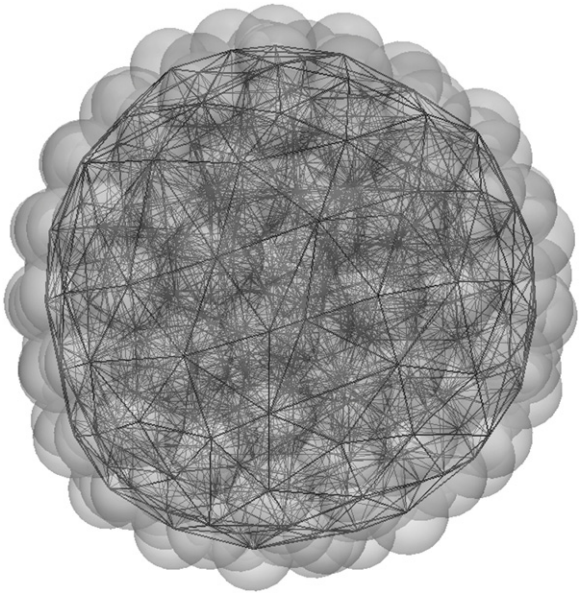

Fig. 2. The network of interconnected cells in a larger spheroid. This figure shows the same cluster of cells of Fig. 1, after 211 h of simulated time, and the edges of the Delaunay triangulation of the cells' centers. There are 575 cells in this cluster, and the network of connections has become much more complex. This only hints at the complexity of the structure which eventually contains several hundred thousands of cells.

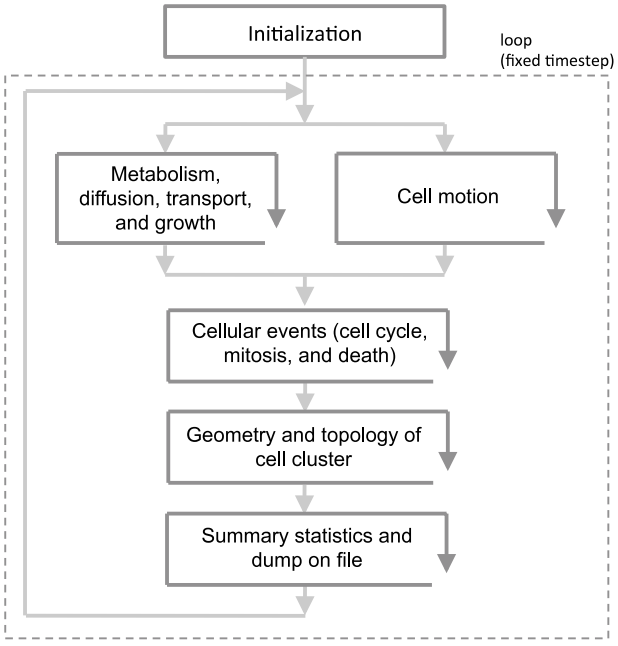

Fig. 3. Schematic layout of program described in [3] that simulate the growth of small avascular solid tumors. After initialization the main loop starts (enclosed in dashed rectangle). The main loop includes several inner loops over cells (denoted by rectangles with arrows); metabolism and mechanical motion do not interact directly and can be calculated by different OpenMP threads. This scheme shows that cell data are repeatedly retrieved from the structure in a cell-based abstraction, leading to an awkward code.

These considerations suggest a new representation of the different chemicals as STL vectors – i.e., expandable vectors, which can accommodate a growing number of proliferating cells – where the vector index corresponds to a given cell. In this "reversed logic" the chemicals themselves become the basic objects, while the aggregate of cells is a mere spatial scaffolding. Here we show how to set up a modular framework to manipulate them.

In this paper we describe the implementation of the new computational design with the aid of a 2-dimensional toy model, which is illustrated in Fig. 4. It consists of a set of cells (represented by circles in Fig. 4) arranged on a 2-dimensional grid on which the diffusion of two different molecular species, named A and B takes place. We wish to stress that here we choose a 2-dimensional lattice only as a practical testbed, while the actual application shall utilize a disordered lattice in a much more complex setting.

The initial concentrations of both chemicals are set to 1 (arbitrary units) in the peripheral cells and remain constant over time, whereas the inner cells have vanishing initial concentrations. Each chemical diffuses through cells and can take part to reaction processes; the dynamics of concentrations is described by second order partial differential equations that are solved numerically. In this example we take a reaction part which is sufficiently simple to avoid an unnecessary emphasis on implementation details, and yet sufficiently close to actual biology to make it realistic. In particular we take a simple Michaelis–Menten dynamics for the first substrate (A), and a double-substrate Michaelis–Menten dynamics that couples A and B. Finally, we remark that while the 2-dimensional lattice is easy to visualize, it is also close to real *in vitro* biological systems.