Journal of Physics: Conference Series **490** (2014) 012129 doi:10.1088/1742-6596/490/1/012129

Competing computational approaches to reaction-diffusion equations in clusters of cells

Sabrina Stella[1], **Roberto Chignola** [2] **and Edoardo Milotti**[1,3]

[1]Department of Physics, University of Trieste, Italy
[2]Department of Biotechnology, University of Verona, Italy
[3]Istituto Nazionale di Fisica Nucleare, Sezione di Trieste, Italy

E-mail: sabrina.stella@ts.infn.it

Abstract. We have developed a numerical model that simulates the growth of small avascular solid tumors. At its core lies a set of partial differential equations that describe diffusion processes as well as transport and reaction mechanisms of a selected number of nutrients. Although the model relies on a restricted subset of molecular pathways, it compares well with experiments, and its emergent properties have recently led us to uncover a metabolic scaling law that stresses the common mechanisms that drive tumor growth. Now we plan to expand the biochemical model at the basis of the simulator to extend its reach. However, the introduction of additional molecular pathways requires an extensive revision of the reaction-diffusion part of the C++ code to make it more modular and to boost performance. To this end, we developed a novel computational abstract model where the individual molecular species represent the basic computational building blocks. Using a simple two-dimensional toy model to benchmark the new code, we find that the new implementation produces a more modular code without affecting performance. Preliminary results also show that a factor 2 speedup can be achieved with OpenMP multithreading, and other very preliminary results indicate that at least an order-of-magnitude speedup can be obtained using an NVidia Fermi GPU with CUDA code.

1. Introduction

Biophysical models of cancer are widely believed to be important tools in the understanding of the disease, and currently there are two main – and often competing – approaches: analytical models [1], and numerical models [2, 3]. We think that both approaches are useful and important, and we have recently given an overview of a synergetic approach that combines both methods [4]. While the difficulties of the analytical methods are well understood, the actual feasibility of computational modeling depends on a plethora of details, and here we discuss an interesting numerical implementation of the reaction-diffusion part of our program for the simulation of the growth of small avascular solid tumors [5]. The program includes the definition of single cells with their phenotype, and of their metabolic activity, growth, and proliferation. The simulations produce 3-D lattice free cell aggregates with nearly spherical shape which can eventually contain more than one million cells [6]. The diffusion and the interaction of a limited, but accurately chosen set of molecular species, within cells and the environment is described by a system of partial differential equation, and the numerical integration of this system is one of the important aspect of the simulation process. The diffusion problem is naturally discretized over the network of cells' centers. In this context it is natural to implement an object-oriented code where single

cells are the basic computational objects, and the masses of metabolites inside cells are object data. Here we describe a new abstract model with a reversed logic, where the molecular species are the basic objects while cells are a mere spatial scaffolding; this approach makes the code more reliable, thanks to a better encapsulation, and more easily expandable.

2. Material and Methods

We explain the inner workings of the new code with the aid of the 2-dimensional toy model which is illustrated in fig.1. It consists of a grid of cells (represented by circles) in which the diffusion

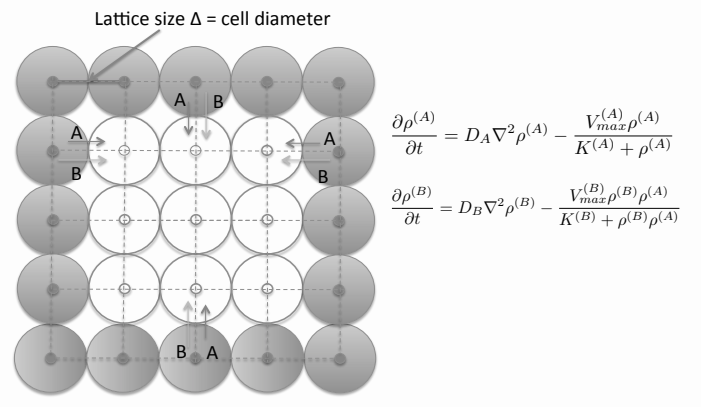

Figure 1. Geometry of the toy model. Cells are arranged in a square two-dimensional lattice. Two different chemical species diffuse throughout the cell aggregate, and the initial concentration values are set to 1 (arbitrary units) along the edge of the lattice and vanish in the remaining cells.

of two different molecular species, A and B, takes place. The initial concentration values are set to 1 (arbitrary unit) along the edge and remain constant over the time, whereas the inner cells has a vanishing concentration value. The molecular species can diffuse throughout the lattice and, at the same time, they take part to reaction processes; the evolution of their concentrations in the lattice can be described by second order partial differential equations

$$\frac{\partial \rho^{(k)}(x,y,t)}{\partial t} = D^{(k)} \nabla^2 \rho^{(k)}(x,y,t) - f^{(k)}[\boldsymbol{\rho}(x,y,t)] \tag{1}$$

where the two terms on the r.h.s. are respectively the diffusion and the reaction terms, and $\boldsymbol{\rho} = (\rho^{(A)}, \rho^{(B)})$ is the vector of the concentrations of two interacting chemicals. The parameters $D^{(k)}$ $(k = A, B)$ are the diffusion constants, whereas the functions $f^{(k)}$ corresponds to a single and double substrate Michaelis-Menten terms respectively for the molecular species A and B as illustrated in fig.1. The equations are solved numerically by discretizing the spatial domain with a step size equal to the diameter of the cells; a $N \times N$ sets of ordinary differential equation is obtained, where N is the lattice size. A stable solution can be obtained using the implicit Euler method [5] which then yields a system of coupled nonlinear equations. These equations are then solved with the Newton-Raphson method; the iterations are stopped when a predetermined accuracy value is finally achieved over the whole lattice.

3. C++ implementation and test runs

The new abstract model has been implemented using two classes that we call `System` and `Chemical`. The class `System` stores information about the environment, in this case the square lattice, and it manages the collective diffusion of all the initially defined molecular species. The class `Chemical` represents a single diffusing molecular species and it stores its spatial concentration map in a single STL vector where the n-th component corresponds to the n-th cell. `Chemical` manages the time evolution of the concentration using the method `Interact()` that calculates the concentration of the molecular species in the whole lattice at time $t + \Delta t$ from the

value known at time t. `Interact()` is specific to each different molecular species, and we use the constructor to define a pointer to external functions that will be successively dereferenced in the body of `Interact()`. In particular two external functions are used to determine the numerical integration method (which might eventually be different from the straightforward implicit Euler method) and to identify the specific metabolic function. Using these classes the toy model is implemented according to the following pseudocode:

1 create the environment where diffusion takes place: define lattice size, lattice spacing, and time steps;
2 define the biochemical system: create `Chemical` objects representing the molecular species A and B;
3 iterate the diffusion steps by calling the `System::CollectiveDiffusion()` method.

Figure 2 shows a concentration map for the molecular species A (roughly corresponding to glucose), obtained with the toy model, with a 100×100 lattice, with lattice spacing (i.e., cell diameter) 5 μm, and with the parameters specified in the cell legend.

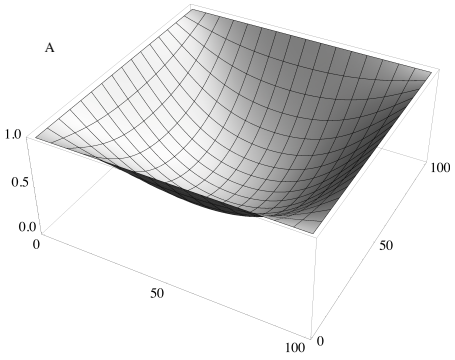

Figure 2. Concentration map of the molecular species A for simulated time of 3600 s, $D = 7 \cdot 10^{-6}\mathrm{cm}^2 s^{-1}$, $V_{max} = 1.2 \cdot 10^{-4}\mathrm{g\,cm}^{-3} s^{-1}$; $K = 0.27 \cdot 10^{-3}\mathrm{g\,cm}^{-3}$. The general shape follows the expected behavior.

Both to check the correctness of the results and to compare the respective performances and structural properties, we implemented the toy model also with a conventional scheme in which the cells are the basic objects (hereafter we call this the *cell-based* model). In the *cell-based* model the concentration values of A and B are data of the `Cell` class, and the corresponding parameters are data of an additional class `CellType` that represents the cell's phenotype. Finally a `CellSystem` class is the equivalent of `System` in previous abstract scheme. The *chemical-based* approach (that is the previously describe one) has a strikingly better encapsulation, since the introduction of a new molecular species does not require any change in the already defined classes but only the introduction of a new metabolic function in a header file. No other additional code is required. On the contrary, in the *cell-based* model the introduction of a new molecular species requires the modification of the whole set of classes.Table 1 shows that the new scheme does not decrease the performance either. Another set of runs with variable lattice size shows that for both abstract models the total execution time T has a faster-than-linear growth with respect to the total number of lattice sites N, $T \approx 2.15 \cdot 10^{-5} N + 2.16 \cdot 10^{-7} N^{1.5}$, where the $N^{1.5}$ term is due to the global convergence condition in the Newton algorithm, since the number of loops required to reach convergence is proportional to the number of lattice sites and to the linear lattice size, i.e., to $N \times \sqrt{N} = N^{1.5}$. These considerations show that the total execution time is dominated by the numerical solution algorithm, and not by memory transfers (more efficient in the *chemical-based* version).

Table 1. Execution times [s] of the diffusion process using the cell-based and chemical-based method, for different simulated diffusion times in a 100×100 lattice. The simulations were run on a quad core MacBook Pro using the compilers Intel C++ XE 13.0 and GCC 4.7.

	Intel XE 13.0		GCC 4.7		GCC 4.7	(opt -O)
Diff.time (s)	Cell-b	Chem-b	Cell-b	Chem-b	Cell-b	Chem-b
3600	0.43	0.47	1.25	1.24	0.44	0.46
5400	0.55	0.58	1.56	1.56	0.56	0.58
7200	0.67	0.70	1.90	1.91	0.67	0.70
9000	0.78	0.83	2.25	2.25	0.78	0.82

4. Code parallelization

The new code has high intrinsic parallelism: using straightforward OpenMP preprocessor directives we obtain a speed-up of about 1.95x in a simulation with 4 different molecular species on a 100x100 lattice, however, this can be pushed much further using a GPU, and the code was slightly modified to exploit the power of a Quadro 4000 NVidia GPU. In this preliminary implementation that does not use any optimization procedure, such as exploiting texture or shared memory [7] we obtain a speedup of about 5x for a lattice size of 1000×1000 and total simulation time equal to 3600 s. A comparison of the execution times for the diffusion-reaction processes implemented on CPU and GPU is shown in table 2.

Table 2. Execution times [s] of the diffusion-reaction process implemented on GPU and CPU with or without optimization flag for different lattice size and simulation time = 3600s.

Lattice size	GPU	CPU (no opt)	speed-up	CPU (-O3)	speed-up
96×96	0.73	1.2	1.6	0.46	<1
296×296	3.04	20.8	6.8	7.8	2.6
600×600	13.53	144.8	10.6	55.2	4.8
1000×1000	46.02	624.8	13.5	239.5	5.2

5. Conclusions

The new computational scheme shall be used in the numerical simulator of tumor spheroids. It is more easier extendible, allowing for an easier introduction of new chemicals, and more robust, providing a better encapsulation of the code, without loss in terms of performance. Moreover, the code running on GPU can be further optimized exploiting its hierarchical memory, to significantly increase the execution speed.

References

[1] Araujo R and McElwain D 2004 *Bulletin of mathematical biology* **66** 1039–1091
[2] Chignola R, FABBRO A D, Farina M and Milotti E 2011 *Journal of Bioinformatics and Computational Biology* **9** 559–577
[3] Chignola R and Milotti E 2012 *AIP Advances* **2** 011204–011204
[4] Milotti E, Vyshemirsky V, Sega M, Stella S, Dogo F and Chignola R 2013 *IEEE/ACM TCBB*
[5] Milotti E, Del Fabbro A and Chignola R 2009 *Computer Physics Communications* **180** 2166–2174
[6] Milotti E and Chignola R 2010 *PLoS One* **5** e13942
[7] Jr F M, Izsákb F, Mészárosc R and Lagzic I 2011 *Chemometrics and Intelligent Laboratory Systems* **108** 76 – 85