Parallel numerical solution of a 2D Chemotaxis-Stokes system on GPUs technology*

Raffaele D'Ambrosio, Stefano Di Giovacchino, and Donato Pera

Department of Information Engineering and Computer Science and Mathematics, University of L'Aquila, Italy
raffaele.dambrosio@univaq.it, stefano.digiovacchino@graduate.univaq.it, donato.pera@univaq.it

Abstract. The aim of this paper is the numerical solution of a 2D chemotaxis model by a parallel numerical scheme, implemented on a GPU technology. The numerical discretization relies on the employ of a finite difference scheme for the spatial part and the explicit Euler method for the time integration. Accuracy and stability properties are provided. The effectiveness of the approach, as well as the coherence of the results with respect to the modeled phenomenon, is provided through a numerical evidence, also giving a performance analysis of the serial and the parallel implementations.

Keywords: Chemotaxis, GPU computing, parallel numerical method.

1 Introduction

Chemotaxis [3, 9, 10, 20, 21, 13, 15] is a very common phenomenon consisting in the movement of an organism in response to a chemical stimulus. For example, in order to find food, bacteria swim toward highest concentration of food molecules [9]. An other example is given by the motion of sperm towards the egg during fertilization in which chemotaxis phenomena is very crucial. Sometimes, as we can read in [20], the mechanism that allows chemotaxis phenomena in animals can be subverted; this is the case, for example, of cancer metastasis.

The model we deal with was first derived in [21] in order to describe the swimming of bacteria and oxygen transport near contact lines. Subsequently, this model was modified and completed by Cao in [3], where he describes the motion of oxygen consumed by bacteria in a drop of water. The model is given by the chemotaxis-Stokes system with a rotational flux term, in a three-dimensional domain. The equations for an incompressible Navier-Stokes fluid are coupled with two parabolic equations, in which the first one presents a chemotactic term. In [3],

* This work is supported by GNCS-INDAM project and PRIN2017-MIUR project. The first author (RD) is member of the INdAM Research group GNCS. The author are thankful to the Department of Information Engineering and Computer Science and Mathematics of the University of L'Aquila for the usage of the HPC Caliban Cluster (https://caliband.disim.univaq.it).

it is proved in the two and three dimensional case the existence and uniqueness of classical solution under a smallness assumption in the initial concentration. We will recall these results and we will give a wider description of this model in Section 2.

Numerical analysis plays a crucial role in computing solutions for PDEs system especially when it is quite difficult to find the analytical one or it is proved under some restriction assumptions on the data of the problem. For this reason, our goal is to develop a numerical scheme to compute the solution of this system and to simulate it. In particular, we expect that there exists a time t after that the bacteria start the chemotaxis and move toward the oxygen.

Numerical solutions often require high spatial resolution to capture the detailed biophysical phenomena. As a consequence, long computational times are often required when using a serial implementation of a numerical scheme. Parallel computation can strongly improve the time efficiency of some numerical methods such as finite differences algorithms, which are relatively simple to implement and apply to a generic PDEs system. The Graphics Processing Units (GPUs) are perfect to use when we want to execute a numerical code based of a very large number of grid points, since the larger is the number of the grid points, the higher is the accuracy of the our numerical solution.

The codes used to study the performance of GPUs presented in this article were programmed using CUDA. The CUDA platform (Compute Unified Device Architecture), introduced by NVIDIA in 2007, was designed to support GPU execution of programs and focuses on data parallelism [11]. With CUDA, graphics cards can be programmed with a medium-level language, that can be seen as an extension to C/C++, without requiring a great deal of hardware expertise. We refer to [14, 16] for a comprehensive introduction to GPU-based parallel computing, including details about the CUDA programming model and the architecture of current generation NVIDIA GPUs. As regards the application of GPU computing to partial differential equations, see [1, 5, 12] and references therein.

It is important to point out that, although the model is set in the three dimensional case we will perform our numerical analysis in a two dimensional setting. This assumption is not too restrictive since this is the most treated case in the literature concerning chemotaxis models (see [10] and reference therein). Indeed in many models, because of the microscopic third dimension, without loss of generality, cells are considered bidimensional.

This paper is organized as follows. In the next section, Section 2, a short description of the biological phenomenon and the equations of the model are presented. We present the numerical scheme in Section 3.

In the Section 4, the analysis of consistency and stability, for our numerical scheme, is given. Moreover, a set of numerical experiments are presented in Section 5. Section 6 contains the comparative performance evaluation between GPUs and CPUs implementations of the numerical scheme. We summarize our work and we give some possible future developments in the final section, Section 7.

2 Mathematical model

In this paper, we study the motion of oxygen consumed by bacteria in a drop of water. In particular, the model describes the motion of bacteria towards the zone of highest concentration of oxygen. However, the bacteria don't move directly toward these area but with some rotations that can be completely random. The following initial boundary problem model, has been introduced in [3] and it is given by the following set of equations,

$$\begin{cases} n_t = \Delta n - \nabla \cdot (nS(x,n,c) \cdot \nabla c) - u \cdot \nabla n, & (x,t) \in \Omega \times (0,T), \\ c_t = \Delta c - nc - u \cdot \nabla c, & (x,t) \in \Omega \times (0,T,), \\ u_t = \Delta u + \nabla P + n \nabla \phi, & (x,t) \in \Omega \times (0,T), \\ \nabla \cdot u = 0, & (x,t) \in \Omega \times (0,T), \\ \nabla c \cdot \nu = (\nabla n - S(x,n,c) \nabla c) \cdot \nu = 0, \ u = 0, & (x,t) \in \partial\Omega \times (0,T), \\ n(x,0) = n_0(x), \ c(x,0) = c_0(x), \ u(x,0) = u_0(x), & x \in \Omega, \end{cases} \tag{1}$$

where Ω is a bounded smooth domain in 2D or 3D, ν is the outward normal vector to the boundary $\partial\Omega$, n is the density of bacteria, c the oxygen's concentration and u and P are the velocity and the pressure of the fluid respectively.

The equation $(1)_1$ describes the density of bacteria. As we can see, this equation is a parabolic equation that admits a diffusion term Δn and a chemotactic term $\nabla \cdot (nS(x,n,c) \cdot \nabla c)$, that says that bacteria always move towards the higher oxygen's areas.
The parabolic equation $(1)_2$ describes the motion of oxygen concentration where the diffusion term is represented by Δc. The equations $(1)_3$ and $(1)_4$ are the well known Navier-Stokes equations for an incompressible fluid, subjected to an external force, without the convective term. This choice is due to the fact that, as we know from [2], the uniqueness of the solution is not yet guaranteed for a three dimensional Navier-Stokes problem. The tensor S is a rotational tensor, that takes into account the rotations of bacteria, and the function ϕ is a potential function that can be associated to an external force, therefore the term $n\nabla\phi$ can be seen as buoyant or electric force of bacterial mass. As in [3], we assume the following regularity conditions for the tensor S

$$s_{ij} \in C^2\left(\overline{\Omega} \times [0,\infty) \times [0,\infty)\right), \tag{2}$$

$$|S(x,n,c)| := \max_{i,j \in \{1,2\}} \{|s_{ij}(x,n,c)|\} \leq S_0 \text{ for all } (x,n,c) \in \overline{\Omega} \times [0,\infty) \times [0,\infty). \tag{3}$$

In order to describe the functional setting for the initial data we need to define the following operators and spaces.

Definition 1 (Stokes operator). *The Stokes operator on $L^p_\sigma(\Omega)$ is defined as $A_p = -\mathcal{P}\Delta$ with domain $D(A_p) = W^{2,p}(\Omega) \cap W^{1,p}_0 \cap L^p_\sigma(\Omega)$, where $\mathcal{P}$ is the so-called Helmholtz projection. Since A_{p_1} and A_{p_2} coincide on the intersection of their domain for $p_1, p_2 \in (1,\infty)$, we will drop the index p.*

We will denote the first eigenvalue of A by λ_1', and by λ_1 the first nonzero eigenvalue of $-\Delta$ on Ω under Neumann boundary conditions.

The conditions on the initial data are as follows:

$$\begin{cases} n_0 \in L^\infty(\Omega), \\ c_0 \in W^{1,q}(\Omega), \quad q > N, \\ u_0 \in D(A^\alpha), \quad \alpha \in (\frac{N}{4}, 1), \end{cases} \tag{4}$$

$$n_0 \geq 0, \ c_0 \geq 0 \text{ on } \Omega. \tag{5}$$

As we will see, we also assume that $\|c_0\|_{L^\infty(\Omega)}$ is small. This assumption will be crucial for the existence of classical solution of (1), indeed the smallness of the initial concentration of bacteria can force the stability of system. With these assumptions, in [3], it has been proved that there exists a unique global classical solution for problem (1), for completeness, we report here these results.

Theorem 1. *Let $\Omega \subset \mathbb{R}^3$ be a bounded domain with smooth boundary. Assume that S fulfills (2) and (3). There is δ_0 with the following property: if the initial data fulfill (4) and (5), and*

$$\|c_0\|_{L^\infty(\Omega)} < \delta_0 \tag{6}$$

then (1) admits a global classical solution (n,c,u,P) which is bounded, and satisfies

$$\begin{cases} n \in C^{2,1}(\overline{\Omega} \times (0,\infty)) \cap C^0_{loc}((\overline{\Omega} \times [0,\infty)), \\ c \in C^{2,1}(\overline{\Omega} \times (0,\infty)) \cap C^0_{loc}((\overline{\Omega} \times [0,\infty)) \cap L^\infty((0,\infty); W^{1,q}(\Omega)), \\ u \in C^{2,1}(\overline{\Omega} \times (0,\infty)) \cap L^\infty((0,\infty); D(A^\alpha)) \cap C^0_{loc}([0,\infty); L^2(\Omega)), \\ P \in L^1((0,\infty); W^{1,2}(\Omega)). \end{cases} \tag{7}$$

3 Numerical scheme

We now aim to discretize the differential problem (1), by a suitable finite difference numerical scheme, according to the classical method-of-lines [4, 6–8, 17, 18]. As we said in the introduction, a simplifying assumption, we consider our problem in a two dimensional setting, therefore our spatial variable are given by (x, y). Moreover, the functions s_{ij} defined in (2) are given by $s_{i,j} = s_{i,j}(x, y, t)$, $i, j = 1, 2$. We assume that the domain Ω has the form $\Omega = [0, 1] \times [0, 1]$ and is discretized as follows. Given an integer N, we denote by $h = 1/(N + 1)$ the spatial stepsize and accordingly define the grid

$$\Omega_h = \{(x_i, y_j) \in \Omega : x_i = ih, \ i = 0, 1, \ldots, N + 1; \ y_j = jh, j = 0, 1, ..., N + 1\}.$$

For a given function $u = u(x, y, t)$, in correspondence of a generic point $(x_i, y_j) \in \Omega_h$, we recall the following finite differences for the approximation of

the first derivates

$$\begin{aligned}\frac{\mathrm{d}}{\mathrm{d}x}u(x_i,y_j,t) &= \frac{u(x_{i+1},y_j,t)-u(x_{i-1},y_j,t)}{2h}+O(h^2),\\ \frac{\mathrm{d}}{\mathrm{d}y}u(x_i,y_j,t) &= \frac{u(x_i,y_{j+1},t)-u(x_i,y_{j-1},t)}{2h}+O(h^2).\end{aligned} \tag{8}$$

As regards the finite difference approximation of the second derivative, we adopt the following usual central finite difference discretization

$$\begin{aligned}\frac{\mathrm{d}^2}{\mathrm{d}x^2}u(x_i,y_j,t) &= \frac{u(x_{i+1},y_j,t)-2u(x_i,y_j,t)+u(x_{i-1},y_j,t)}{h^2}+O(h^2),\\ \frac{\mathrm{d}^2}{\mathrm{d}y^2}u(x_i,y_j,t) &= \frac{u(x_i,y_{j+1},t)-2u(x_i,y_j,t)+u(x_i,y_{j-1},t)}{h^2}+O(h^2).\end{aligned} \tag{9}$$

Therefore, we have

$$\begin{aligned}\Delta u(x_i,y_j,t) = \frac{1}{h^2}\Big(&u(x_{i+1},y_j,t)+u(x_{i-1},y_j,t)+u(x_i,y_{j+1},t)\\ &+u(x_i,y_{j-1},t)-4u(x_i,y_j,t)\Big)+O(h^2).\end{aligned}$$

For any fixed time t, we denote by u_{ij} an approximate value of $u(x_i,y_j,t)$, with $i,j=0,1,\ldots,N+1$. Then, for $i,j=1,2,\ldots,N$, we obtain

$$\frac{\mathrm{d}u}{\mathrm{d}x}(x_i,y_j,t)\approx\frac{u_{i+1,j}-u_{i-1,j}}{2h},\qquad \frac{\mathrm{d}u}{\mathrm{d}y}(x_i,y_j,t)\approx\frac{u_{i,j+1}-u_{i,j-1}}{2h},$$

$$\frac{\mathrm{d}^2u}{\mathrm{d}x^2}(x_i,y_j,t)\approx\frac{u_{i+1,j}-2u_{i,j}+u_{i-1,j}}{h^2},\ \frac{\mathrm{d}^2u}{\mathrm{d}y^2}(x_i,y_j,t)\approx\frac{u_{i,j+1}-2u_{i,j}+u_{i,j-1}}{h^2}$$

and the five-point stencil

$$\Delta u(x_i,y_i,t)\approx\frac{u_{i+1,j}+u_{i-1,j}+u_{i,j+1}+u_{i,j-1}-4u_{i,j}}{h^2}$$

for the Laplacian operator.

For the time discretization, we divide the time interval $[0,T]$, in M equidistant parts of length

$$h_t=\frac{T}{M}.$$

Then, we define the grid

$$\mathcal{I}_{h_t}=\{t_k\in[0,T],\ t_k=kh_t,\ k=0,1,\ldots,M\} \tag{10}$$

and denote by $u_{ij}^{(k)}$ an approximation of $u(x_i,y_j,t_k)$, with $(x_i,y_j)\in\Omega_h$, $t_k\in\mathcal{I}_{h_t}$.

In view of a parallel implementation, we adopt as time discretization that arises from the forward Euler scheme because it is directly parallelizable.The

fully discretized problem reads as follows: as regards equation $(1)_2$, we have

$$c_{i,j}^{(k+1)} = c_{i,j}^{(k)} + h_t \Bigg(\frac{c_{i+1,j}^{(k)} - 4c_{i,j}^{(k)} + c_{i-1,j}^{(k)} + c_{i,j-1}^{(k)} + c_{i,j+1}^{(k)}}{4h^2} - n_{i,j}^{(k)} c_{i,j}^{(k)} - u_{1_{i,j}} \frac{c_{i+1,j}^{(k)} - c_{i-1,j}^{(k)}}{2h} - u_{2_{i,j}} \frac{c_{i,j+1}^{(k)} - c_{i,j-1}^{(k)}}{2h} \Bigg).$$

For equation $(1)_3$, we have, for the component u_1,

$$u_{1_{i,j}}^{(k+1)} = u_{1_{i,j}}^{(k)} + h_t \Bigg(\frac{u_{1_{i+1,j}}^{(k)} - 4u_{1_{i,j}}^{(k)} + u_{1_{i-1,j}}^{(k)} + u_{1_{i,j-1}}^{(k)} + u_{1_{i,j+1}}^{(k)}}{4h^2} + \frac{P_{i+1,j}^{(k)} - P_{i-1,j}^{(k)}}{2h} + n_{i,j}^{(k)} \frac{\phi_{i+1,j} - \phi_{i-1,j}}{2h} \Bigg)$$

and a similar formula also holds true for the component u_2. For equation $(1)_4$, we have

$$\frac{u_{1_{i+1,j}}^{(k)} - u_{1_{i-1,j}}^{(k)} + u_{2_{i,j+1}}^{(k)} - u_{2_{i,j-1}}^{(k)}}{2h} = 0.$$

As regards equation $(1)_1$, we have

$$n_{i,j}^{(k+1)} = n_{i,j}^{(k)} + \frac{h_t}{4h^2} \left[\alpha_{i,j}^{(k)} - 2h \left(u_{1_{i,j}}^{(k)} (n_{i+1,j}^{(k)} - n_{i-1,j}^{(k)}) + u_{2_{i,j}}^{(k)} (n_{i,j+1}^{(k)} - n_{i,j-1}^{(k)}) \right) \right],$$

for $i, j = 1, ..., N$, where $\alpha_{ij}^{(k)}$ contains all the terms of discretization independent on ${u_1}_{i,j}^{(k)}$ and ${u_2}_{i,j}^{(k)}$.

We finally provide a discretized equation for the pressure P. Indeed, by the incompressibility assumption $(1)_4$, the pressure P satisfies, at any time t, the equation

$$\Delta P = -\nabla \cdot (n \nabla \phi) = -\nabla n \cdot \nabla \phi - n \Delta \phi, \tag{11}$$

whose discretization leads to

$$\begin{aligned} &\frac{P_{i+1,j}^{(k)} + P_{i-1,j}^{(k)} + P_{i,j+1}^{(k)} + P_{i,j-1}^{(k)} - 4P_{i,j}^{(k)}}{h^2} = \\ &\quad - \frac{(n_{i+1,j}^{(k)} - n_{i-1,j}^{(k)})(\phi_{i+1,j} - \phi_{i-1,j}) - (n_{i,j+1}^{(k)} - n_{i,j-1}^{(k)})(\phi_{i,j+1} - \phi_{i,j-1})}{4h^2} \\ &\quad - n_{i,j}^{(k)} \frac{\phi_{i+1,j} + \phi_{i-1,j} + \phi_{i,j+1} + \phi_{i,j-1} - 4\phi_{i,j}}{h^2}, \end{aligned} \tag{12}$$

for $i, j = 1, 2, \ldots, N$.

We observe that, as regards the boundary conditions, we will always use the Dirichlet ones in the remainder of the treatise, that will give the values of the unknown functions when $(i, j) = (0, 0)$ and $(i, j) = (N + 1, N + 1)$.

4 Consistency and stability analysis

In this section, we want to analyze the consistency and stability of the numerical scheme introduced in the previous section. For the sake of clarity, here we distinguish the contribution to the global error arising from the spatial discretization and that coming from the time discretization. We observe that our analysis is given for problems having sufficient regularity in order to make the application of Taylor series arguments possible.

4.1 Analysis of the spatial discretization

Let us consider the system (1) and analyze the contribution to the global error associated to the space discretization of each equation. Let us first focus our attention on Equation $(1)_2$. Replacing the exact solution in the right-hand side of the spatially discretized equation referred to the generic point (x, y) of the grid gives

$$\begin{aligned}
&\Delta c(x,y) - n(x,y)c(x,y) - u(x,y)\cdot\nabla c(x,y) \approx \\
&\qquad \approx \frac{c(x+h,y)+c(x-h,y)+c(x,y+h)+c(x,y-h)-4c(x,y)}{h^2} \\
&\qquad - n(x,y)c(x,y) - u_1(x,y)\frac{c(x+h,y)-c(x-h,y)}{2h} \\
&\qquad - u_2(x,y)\frac{c(x,y+h)-c(x,y-h)}{2h},
\end{aligned} \tag{13}$$

where we have denoted the spatial stepsize by h and neglected the time dependence for the sake of brevity. Expanding $c(x+h, y)$, $c(x-h, y)$, $c(x, y+h)$ and $c(x, y-h)$ in Taylor series around (x, y) and collecting the resulting expressions in (13), we obtain

$$\begin{aligned}
&c_{xx}(x,y) + c_{yy}(x,y) - n(x,y)c(x,y) - u_1(x,y)c_x(x,y) - u_2(x,y)c_y(x,y) \\
&\quad \approx c_{xx}(x,y) + c_{yy}(x,y) + \frac{h^2}{12}\left(c_{xxxx}(x,y) + c_{yyyy}(x,y)\right) - n(x,y)c(x,y) \\
&\quad - u_1(x,y)\left(c_x(x,y) + \frac{h^2}{6}c_{xxx}(x,y)\right) - u_2(x,y)\left(c_y(x,y) + \frac{h^2}{6}c_{yyy}(x,y)\right).
\end{aligned}$$

This implies that the deviation between the exact operator and its spatial discretization is $\{\mathcal{O}\}(h^2)$. For all the other equations in (1), the analysis proceeds exactly in the same way. Therefore, the residuum obtained by replacing the exact solution in the spatially discretized problem is $\mathcal{O}(h^2)$. Then, the spatial discretization is consistent of order 2.

4.2 Analysis of the time discretization

Let us rewrite the numerical scheme introduced in Section 3 in the following form

$$\begin{cases} \dot{W}(t) = f(W(t)), \\ W(0) = W_0, \end{cases} \tag{14}$$

where $W^{(k)} \in \mathbb{R}^{4N^2}$ is the vector

$$W^{(k)} = \begin{bmatrix} c^{(k)} \\ n^{(k)} \\ u_1^{(k)} \\ u_2^{(k)} \end{bmatrix},$$

being $c^{(k)}$ is the vectorization of the matrix $\left(c_{ij}^{(k)}\right)_{i,j=1}^{N}$, and so it is for the other entries of $W^{(k)}$. By applying the explicit Euler method, we have, at time $t_k \in \mathcal{I}_{h_t}$ defined in (10),

$$W^{(k)} = W^{(k-1)} + h_t f(W^{k-1}). \tag{15}$$

Clearly, the numerical scheme (15) is consistent with problem (1). We now aim to provide the conditions ensuring its stability. To this purpose, it is useful to rewrite the vector field f of (14) in the form

$$f(W^{(k)}) = GW^{(k)} + F(W^{(k)}),$$

where the matrix $G \in \mathbb{R}^{4N^2 \times 4N^2}$ contains the linear part of numerical scheme and the nonlinear function $F(W^{(k)}) \in \mathbb{R}^{4N^2}$ is the vector containing the nonlinear part of f. In this regard, following the lines drawn in [4, 6, 7, 19], the following result holds.

Theorem 2. *The numerical method (15) is stable if*

$$\|I + h_t G\|_\infty + h_t F_{max} \leq 1,$$

being I the identity matrix and F_{max} an upper bound for the norm of the gradient of F.

Proof. Let us consider a perturbation of the solution at the step k, denoted by $\widetilde{W}^{(k)}$, that is,

$$\widetilde{W}^{(k)} = W^{(k)} + E^{(k)}.$$

By applying the method (15) to $\widetilde{W}^{(k)}$, we obtain

$$\widetilde{W}^{(k+1)} = \widetilde{W}^{(k)} + h_t[G\widetilde{W}^{(k)} + F(\widetilde{W}^{(k)})].$$

Therefore, we have

$$\begin{aligned} E^{(k+1)} &= W^{(k+1)} - \widetilde{W}^{(k+1)} \\ &= W^{(k)} + h_t[GW^{(k)} + F(W^{(k)})] - \widetilde{W}^{(k)} - h_t[G\widetilde{W}^{(k)} + F(\widetilde{W}^{(k)})] \\ &= E^{(k)} + h_t G E^{(k)} + h_t[F(W^{(k)}) - F(\widetilde{W}^{k})] \\ &= (I + h_t G)E^{(k)} + h_t[F(W^{(k)}) - F(W^k + E^{(k)})]. \end{aligned} \tag{16}$$

By Taylor expansion arguments for $F(W^{(k)} + E^{(k)})$ around $W^{(k)}$, we have

$$E^{(k+1)} \leq (I + h_t G)E^{(k)} + h_t[\nabla F(W^{(k)})E^{(k)}].$$

Therefore, passing to the norm, we obtain

$$\begin{aligned} \left\|E^{(k+1)}\right\|_\infty &\leq \|I + h_t G\|_\infty \left\|E^{(k)}\right\|_\infty + h_t \left\|\nabla F(W^{(k)})\right\|_\infty \left\|E^{(k)}\right\|_\infty \\ &\leq \|I + h_t G\|_\infty \left\|E^{(k)}\right\|_\infty + h_t F_{max} \left\|E^{(k)}\right\|_\infty . \end{aligned}$$

Thus, we obtained the following stability inequality

$$\left\|E^{(k+1)}\right\|_\infty \leq (\|I + h_t G\|_\infty + h_t F_{max}) \left\|E^{(k)}\right\|_\infty , \tag{17}$$

leading to the thesis.

□

5 Simulations and numerical results

In this section, we present our main numerical results. In particular, we are interested in observing the chemotactic effect of bacteria towards the oxygen. For simplicity, we have considered the rotational tensor S to be the identity. Moreover, we have supposed the vector field u to be null at the initial time and we have considered the following initial data for the function n, c and P:

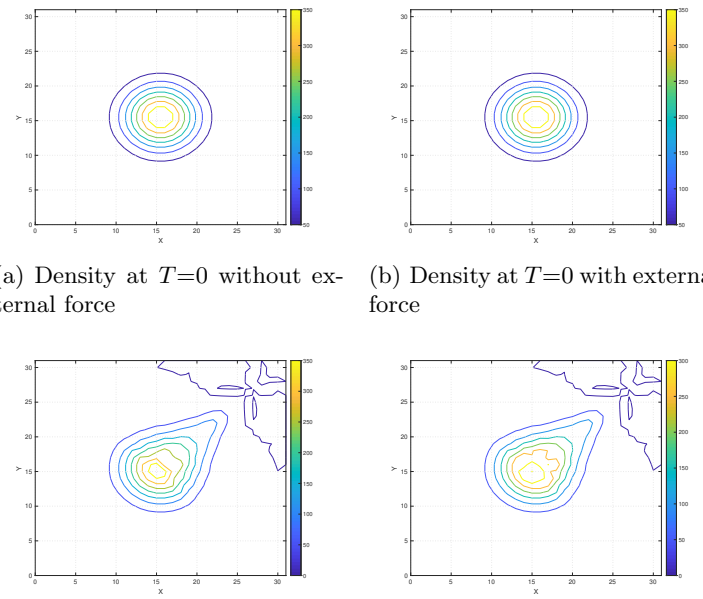
$$P(x,y) = n(x,y) = \frac{100}{\sqrt{2\pi\sigma_n^2}} e^{-\frac{1}{2\sigma_n^2}[(x-\mu_n)^2+(y-\mu_n)^2]},$$

$$c(x,y) = \frac{65}{\sqrt{2\pi\sigma_c^2}} e^{-\frac{1}{2\sigma_c^2}[(x-\mu_c)^2+(y-\mu_c)^2]},$$

where $\sigma_n^2 = 0.01$, $\sigma_c^2 = 0.025$, $\mu_n = 0.5$ and $\mu_c = 0.8$. Moreover, we have assumed the potential to have the following form

$$\phi(x,y) = -\frac{70}{\sqrt{2\pi\sigma_\phi^2}} e^{-\frac{1}{2\sigma_\phi^2}[(x-\mu_{\phi_x})^2+\frac{(y-\mu_{\phi_y})^2}{2}]},$$

where $\sigma_\phi^2 = 1$, $\mu_{\phi_x} = 1$ and $\mu_{\phi_y} = 0.5$ In order to see the phenomenon and to preserve the stability, we have to choose as time step $h_t = 2\text{e}-10$. The numerical pattern for the density at various time is depicted in Figure 1. In our simulations, at the initial time, the bacteria are concentrated on the centre of the domain. Therefore, the action of potential is not well visible. At time $t = 0.2$ms, we can see that the external force exerts a braking action on the bacteria in their motion toward the oxygen.



(a) Density at T=0 without external force

(b) Density at T=0 with external force

(c) Density at T=0.2 ms without external force

(d) Density at T=0.2 ms with external force

Fig. 1. Time evolution of the density towards the oxygen with grid size 32×32, where $T = T_{fin}$.

6 GPU programming and performance evaluation

In this section we describe the basic logical steps required to implement the GPU codes and also the performance evaluation metrics used to evaluate the computing performance. As discussed in Section 3, the numerical scheme relies on a finite differences method for the spatial discretization and a time integration based on the explicit Euler method. From the programming point of view this mathematical approach leads to design a code where a *for loop* defines the clock time steps and where spatial values at each time iteration are updated in parallel by the GPU using the aforementioned numerical scheme.

This basic idea is to use the CPU (host) as owner of time clock activities and the GPU (device) as the owner of the massive computing activities related to the spatial part of the equations. This will lead to a *master-slave* model in which the CPU is the master because it controls the parallel executions on GPU and, therefore, the GPU works only on the spatial part of our scheme. The implemented code employs only the *global memory* in the CUDA kernel codes, while further optimitazions will be subject of a future work.

According to all the principles above described, the code follows the following logical steps:

1. the CPU (host) loads the initial data from the its memory to the GPU (device) memory, global memory;
2. the GPU provides the massive computing activities, that is, the GPU has to execute the code related to spatial discretization because it is the parallelizable part of the code since, at each time step, the values referring to the current step only depends on those already computed in the previous one;
3. the GPU sends back to the CPU the partial/final results;
4. the CPU checks the time step and according to the maximum time value defined by the user restarts/stops the parallel computing process.

We have executed the code on two distinct architectures. The first has the following specifications: HP DL 585 G7 PROLIANT, with processors 4x AMD 6128 (8 core), with clock's frequency 2.0 GHz and RAM 64 Gb, in which a GTX GeForce 1080, 8 Gb RAM, is integrated. The GPU is the only difference between this architecture and the second one. Here, there are 3x GeForce GTX 670, 4 Gb RAM. The operative system used is Linux CentOS 6.5 . Finally, we have compiled the serial code with gcc 4.4.7 and the CUDA-C code with CUDA 9.1 in the first machine and CUDA 8.0 in the second one. In order to evaluate the performances of the two machines, it is very reasonable to compute the number of floating point operations executed for unity of time on GPUs, as a function of the dimension of the grid. Therefore, for any fixed size of the grid, if n is number of floating point operations and T is the CUDA code execution time on the GPUs, we have computed the number $f_{op} = n/T$ of floating point operations for unity of time (seconds). For the number of floating point operations n, we adopt the experimental values reported in Table 2. We remark that the following results have been obtained with a single precision and that we have estimated the number of operations using the NVIDIA *nvprof* tool.

In Table 1, we can see the time comparison between the serial execution, the parallel one on different devices by using GPUs and the parallel one by using a *shared memory* architecture with OpenMP with different number of threads, while in Table 2, we can observe the total number of floating points operations done by devices. We report the corresponding graphs in Figures 2 and 3. In particular, we can observe a good improvement by parallelization passing from the technology GTX GeForce 670 to GTX GeForce 1080. About the metrics, from Figure 3, we note that the numbers of floating point operations executed, for unity of time, on GTX Force 1080 is much higher than the one on GeForce GTX 670.

Dim	Serial kernel	GTX Force 1080	GeForce GTX 670	OpenMP(8)	OpenMP(16)	OpenMP(32)
32	1.156	13.920 $\times 10^{-6}$	18.677 $\times 10^{-3}$	0.524	0.686	6.918
64	4.134	25.120 $\times 10^{-6}$	21.203 $\times 10^{-3}$	1.708	1.637	1.487
128	18.629	124.06 $\times 10^{-6}$	50.671 $\times 10^{-3}$	11.565	9.455	9.547
256	75.635	741.16 $\times 10^{-6}$	183.45 $\times 10^{-3}$	46.065	38.635	41.320

Table 1. Computation times, in seconds, for the serial execution, parallel execution on GTX GeForce 1080, parallel execution on GTX GeForce 670 and parallel execution by using shared memory with openMP with different number of threads.

Grid	GTX Force 1080	GeForce GTX 670
32 × 32	204300	326700
64 × 64	872588	1395372
128 × 128	3603851	5762988
256 × 256	14633343	23419308
512 × 512	59035533	94416300
1024 × 1024	237085743	379147692
2048 × 2048	950039996	1519560108

Table 2. Number of operations floating points executed on GTX GeForce 1080 and on GTX GeForce 670.

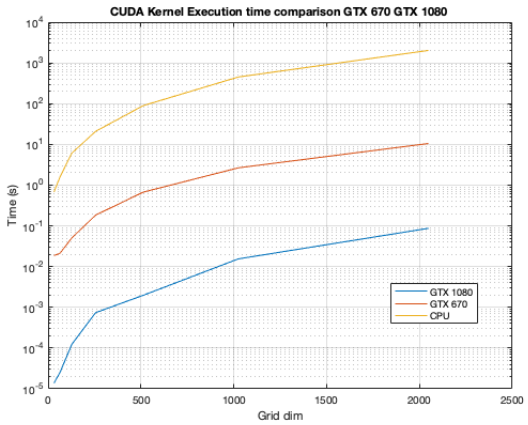


Fig. 2. Time comparison between the three architectures in the semilogarithmic scale. We can see the improvement that we have obtained by passing from CPU technology to GTX GeForce 670, until to GTX GeForce 1080.

7 Conclusions and future works

We have developed a parallel numerical scheme, implemented on GPU, to compute the solution of a chemotaxis system. We have made use of the central finite differences to approximate the spatial derivatives and of the explicit Euler method to discretize the time evolution of the system. We have analyzed accuracy and stability issues, implemented the code on two different architectures, namely, GTX GeForce 1080 and GTX GeForce 670 and compared their performances in terms of time execution and number of floating point operations for unity of time. For the GPU kernel execution, we have used the global memory and implemented a master-slave model, in which the CPU controls the time evolution while the GPU works exclusively on the spatial derivatives of our scheme. Future issues of this research are oriented to providing a 3D model with a deep optimized CUDA kernel code implemented by using dynamic parallelism and shared memory.

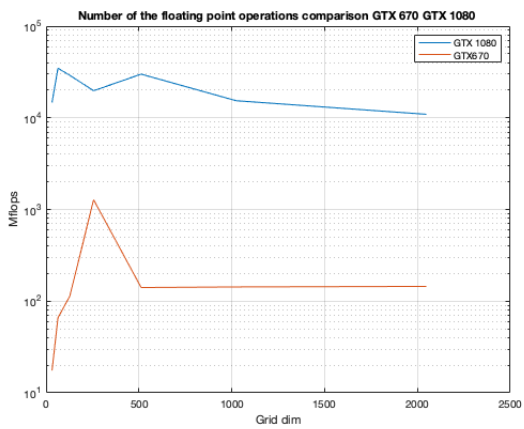


Fig. 3. Number of the floating points operations for unity of time comparison in the semilogarithmic scale. We note that the maximal value for GTX GeForce 1080 is reached in dimension 64×64 and the one for GTX GeForce 670 is reached in dimension 256×256.

References

1. M. Aissa, T. Verstraete, C. Vuik, Toward a GPU-aware comparison of explicit and implicit CFD simulations on structured meshes, Comput. Math. Appl. 74(1), 201–217 (2017).
2. F. Boyer, P. Fabrie, Mathematical Tools for the Study of the Incompressible Navier-Stokes Equations and Related Models, Springer-Verlag, New York (2013).

3. X. Cao, Global classical solutions in chemotaxis(-Navier)-Stokes system with rotational flux term, J. Differ. Equations 261(12), 6883–6914 (2016).
4. A. Cardone, R. D'Ambrosio, B. Paternoster, Exponentially fitted IMEX methods for advection-diffusion problems, J. Comput. Appl. Math. 316, 100-108 (2017).
5. D. Conte, R. D'Ambrosio, B. Paternoster, GPU acceleration of waveform relaxation methods for large differential systems, Numer. Algorithms 71(2), 293-310 (2016).
6. R. D'Ambrosio, M. Moccaldi, B. Paternoster, Adapted numerical methods for advection-reaction-diffusion problems generating periodic wavefronts, Comput. Math. Appl. 74(5), 1029-1042 (2017).
7. R. D'Ambrosio, M. Moccaldi, B. Paternoster, Parameter estimation in IMEX-trigonometrically fitted methods for the numerical solution of reaction-diffusion problems, Comput. Phys. Commun. 226, 55-66 (2018).
8. R. D'Ambrosio, B. Paternoster, Numerical solution of reaction-diffusion systems of lambda-omega type by trigonometrically fitted methods, J. Comput. Appl. Math. 294 C, 436-445 (2016).
9. S. de Oliveira, E.E. Rosowski, A. Huttenlocher, Neutrophil migration in infection and wound repair: going forward in reverse, Nat. Rev. Immunol. 16(6), 378–391 (2016).
10. M. Di Francesco, D. Donatelli, Singular convergence of nonlinear hyperbolic chemotaxis systems to Keller-Segel type models, Discrete Contin. Dyn. Syst. Ser. B 13(1), 79–100 (2010).
11. D.B. Kirk, W.M.W. Hwu, Programming Massively Parallel Processors: A Hands-on Approach, Morgan Kaufmann Publishers Inc., San Francisco (2010).
12. D.J. Magee, K.E. Niemeyer, Accelerating solutions of one-dimensional unsteady PDEs with GPU-based swept time–space decomposition, J. Comput. Phys. 357, 338–352 (2018).
13. C. Málaga, A.A. Minzoni, R.G. Plaza, C. Simeoni, A chemotactic model for interaction of antagonistic microflora colonies: front asymptotics and numerical simulations, Stud. Appl. Math. 130(3), 264–294 (2013).
14. Nvidia CUDA C Programming Guide, Version 4.2, NVIDIA Corporation (2012).
15. D. Pera, C. Málaga, C. Simeoni, R. G. Plaza, On the efficient numerical simulation of heterogeneous anisotropic diffusion models for tumor invasion using GPUs, Rend. Mat. Appl. 7(40), 233–255 (2019).
16. J. Sanders, E. Kandrot, CUDA by Example: An Introduction to General-Purpose GPU Programming, Addison-Wesley Professional (2010).
17. W.E. Schiesser, The Numerical Method of Lines: Integration of Partial Differential Equations, Academic Press, San Diego (1991).
18. W.E. Schiesser, G.W. Griffiths, A Compendium of Partial Differential Equation Models: Method of Lines Analysis with Matlab, Cambridge University Press (2009).
19. G.D. Smith, Numerical solution of partial differential equations: Finite Difference Methods, Clarendon Press, Oxford (1985).
20. C.H. Stuelten, C.A. Parent, D.J. Montell, Cell motility in cancer invasion and metastasis: insights from simple model organisms, Nat. Rev. Cancer. 18(5), 296–312 (2018).
21. I. Tuval, L. Cisneros, C. Dombrowski, C.W. Wolgemuth, J.O. Kessler, R.E. Goldstein, Bacterial swimming and oxygen transport near contact lines, P. Natl. Acad. Sci. Usa 102(7), 2277–2282 (2005).