Modelling and Simulation of a Polluted Water Pumping Process

Chitra Alavani[1], Roland Glowinski[2], Susana Gomez[3], Benjamin Ivorra[4], Pallavi Joshi[1] and Angel Manuel Ramos[4]

[1] Interdisciplinary School of Scientific Computing, University of Pune, India.

[2] Department of Mathematics, University of Houston, 651 P. G. Hoffman Hall, Houston, TX 77204-3008, USA

[3] Instituto de Matematicas Aplicadas y Sistemas, Universidad Nacional A. de Mexico, Mexico

[4] Departamento de Matemática Aplicada, Universidad Complutense de Madrid, Plaza de Ciencias 3, 28040, Madrid, Spain

Abstract

The objective of this article is to discuss the modeling and simulation of the motion of oil spots in the open sea, and the effect on the pollutant concentration when a polluted water pumping ship follows a pre–assigned trajectory to remove the pollutant. We assume here that the oil spots motion is due to the coupling of diffusion, the transport from the wind, sea currents and pumping process and the reaction due to the extraction of oil, implying that the mathematical model will be of advection-reaction-diffusion type. Our discussion includes the description of a parallelization of the selected numerical procedure. We present some results of numerical experiments showing that indeed the parallelization makes the model evaluation more efficient.

Keywords: *Modelling; dvection-reaction-diffusion equation; Upwind scheme; Finite volume scheme; Parallelization; Sea pollution*

1 Introduction

Sea pollution by oil spills has become a major environmental issue nowadays. Today it is commonly accepted by the various concerned communities that modelling and simulation can provide a significant help in the understanding of these catastrophic events and in ways to remediate their environmental consequences. In this article, which is a contribution to the understanding of the cleaning of sea pollution, we investigate the following issues:

i) The modelling of the motion of oil spots resulting from the combined effects of diffusion and of transport by wind and sea currents

ii) The modelling of the physical phenomena associated with the action of the pumping ship, assuming that it follows a pre-assigned trajectory (in a following article we will discuss the identification of a trajectory optimizing the efficiency of the pumping).

iii) The construction of a numerical model for the simulation of the propagation of the oil spots under the pumping.

iv) The parallelization of the numerical algorithms.

The results of numerical experiments are presented. They validate the computational methodology discussed here.

2 A mathematical model for oil spills

In this article we address the pollution cleaning problem assuming information on the wind and sea currents velocity fields in a time interval $(0, T)$. We consider a spatial domain $\Omega = (x_{1,\min}, x_{1,\max}) \times (x_{2,\min}, x_{2,\max}) \subset \mathbb{R}^2$, large enough to ensure that the pollutant will stay in Ω during the corresponding time interval. We assume for simplicity that the density of the pollutant is smaller than the one of the sea water (so that it remains at the top) and the layer-thickness of the pollutant is a constant h. The office of response and restoration of the U.S. National Ocean Service (see `http://response.restoration.noaa.gov`) provides the estimation of the values of h, given in Table 1, depending on the color of the oil in the water. We denote by $c(x, t)$ the pollutant superficial concentration, measured as the volume of pollutant per surface area at $\{x, t\} \in \Omega \times (0, T)$. We assume that in the absence of pumping, the evolution of c is governed by three main effects, namely:

(i) Diffusion of the pollutant

(ii) Transport due to the wind

(iii) Transport due to the sea currents

Oil Color	Layer-Thickness Interval (μm)
Silver	0.04 - 0.30
Rainbow	0.30 - 5.0
Metallic	5.0 - 50
Transitional Dark	50 - 200
Dark	$\geq$ 200

Table 1: Layer-thickness interval of the oil spot in metric unit for each of the oil codes. Data from the office of response and restoration of the U.S. National Ocean Service: `http://response.restoration.noaa.gov`

Under these assumptions, the space–time distribution of c is governed by the following advection-diffusion type equation:

$$\begin{cases} \dfrac{\partial c}{\partial t} + \nabla \cdot \mathbf{J} = 0 & \text{in } \Omega \times (0,T), \\ c = 0 & \text{on } \partial\Omega \times (0,T), \\ c(x,0) = c_0(x), & x \in \Omega. \end{cases} \tag{1}$$

In (1), $c_0(.)$ is the initial superficial concentration in Ω and the flux $\mathbf{J}$ can be decomposed as

$$\mathbf{J} = \mathbf{J}_d + \mathbf{J}_w + \mathbf{J}_s, \tag{2}$$

where

- $\mathbf{J}_d$ is the diffusion flux, given by

$$\mathbf{J}_d = -\mathbf{k}\nabla c, \tag{3}$$

  with $\mathbf{k} = \begin{pmatrix} k_1 & 0 \\ 0 & k_2 \end{pmatrix}$ and k_1, k_2 two positive constants.

- $\mathbf{J}_w$ is the wind flux, given by

$$\mathbf{J}_w = c \ \mathbf{w}, \tag{4}$$

  where $\mathbf{w}$ is the wind velocity multiplied by a suitable drag factor.

- $\mathbf{J}_s$ is the flux associated with the transport due to sea currents, given by

$$\mathbf{J}_s = c \ \mathbf{s}, \tag{5}$$

where $\mathbf{s}$ is the current velocity.

Combining (1) with (2)–(5), we obtain

$$\begin{cases} \dfrac{\partial c}{\partial t} - \nabla \cdot \mathbf{k}\nabla c + \nabla \cdot c\mathbf{w} + \nabla \cdot c\mathbf{s} = 0 & \text{in } \Omega \times (0,T), \\ c = 0 & \text{on } \partial\Omega \times (0,T), \\ c(x,0) = c_0(x), & x \in \Omega. \end{cases} \tag{6}$$

Remark 1 *The homogeneous boundary condition in (1)and (6) follows from that fact that Ω has been assumed large enough to contain the oil spills during the time interval $(0,T)$ (of course this assumes that $\mathbf{k}$ is small enough so that the positive values of c on $\partial\Omega$ resulting from diffusion can be neglected). In a forthcoming publication we will replace the linear diffusion term in (6) by a nonlinear one producing a diffusion propagating with finite velocity.*

Remark 2 *The Coriolis effect, is actually included in the sea current $\mathbf{J}_s$.*

3 Modelling of the pumping process

Among the various techniques which have been employed to remediate oil spills related pollution, we will focus on the one based on a pumping process carried on by a ship (see `http://apgdepollution.free.fr` for details). From a modelling point of view an important step is the inclusion of the reacting effect in system (7). Concerning the pumping process, we are going to assume that:

- The pumping ship follows a pre–assigned trajectory $\gamma_p(t), t \in [0,T]$, that remains inside the region Ω.
- The pump is a cylinder with a cross section of radius R_p and height h_p (we suppose $h_p \geq h$) that pumps the fluid at a velocity Q in the radial directions. Therefore, the pumped oil volume per unit time is $2\pi R_p Q c$ corresponding to a pump density of $\frac{2\pi R_p Q c}{\pi R_p^2}$.

From the above assumptions, the variant of (1) including the pumping effect reads as follows:

$$\begin{cases} \dfrac{\partial c}{\partial t} - \nabla \cdot \mathbf{k}\nabla c + \nabla \cdot c\,\mathbf{w} + \nabla \cdot c\,\mathbf{s} \\ \qquad\qquad +\nabla \cdot c\,\mathbf{p} = -\dfrac{2Q}{R_p}\, c\ \chi_{B(\gamma_p(t),R_p)}(x), & \text{in } \Omega \times (0,T), \\ c = 0, & \text{on } \partial\Omega \times (0,T), \\ c(x,0) = c_0(x), & x \in \Omega, \end{cases} \tag{7}$$

with

$$\mathbf{p} = \begin{cases} QR_p \dfrac{\overrightarrow{\gamma_p(t)x}}{|\overrightarrow{\gamma_p(t)x}|^2}, & \forall x \in \Omega \backslash \bar{B}(\gamma_p(t), R_p), \\ 0, & \forall x \in B(\gamma_p(t), R_p), \end{cases} \tag{8}$$

and

$$\chi_{B(\gamma_p(t),R_p)}(x) = \begin{cases} 0, & \forall x \in \Omega \backslash \bar{B}(\gamma_p(t), R_p), \\ 1, & \forall x \in B(\gamma_p(t), R_p), \end{cases} \tag{9}$$

where $\bar{B}(\gamma_p(t), R_p)$ is the ball of center $\gamma_p(t)$ and radius R_p.

Remark 3 *System (7) is an advection-reaction-diffusion problem, the reaction term being associated with the right hand side of the first equation in* (7) *and with* (8)*.*

Remark 4 *We could approximate* (8) *by*

$$\mathbf{p} = \psi(|\overrightarrow{\gamma_p(t)x}|) \frac{\overrightarrow{\gamma_p(t)x}}{|\overrightarrow{\gamma_p(t)x}|^2} \tag{10}$$

with

$$\psi(r) = \begin{cases} 0, & \text{if } 0 \leq r < R_p, \\ \dfrac{QR_p}{r} & \text{if } R_p \leq r \leq \bar{R}, \\ 0, & \text{if } r > \bar{R}, \end{cases} \tag{11}$$

where $\bar{R}$ is large enough to ensure that the velocity field generated by the pump, at a distance $\bar{R}$ of the pump, can be neglected.

Remark 5 *If the wind and the see currents velocity field $\mathbf{w} + \mathbf{s}$ depend only on time we can perform a change of variables $(t, x) \to (t, \bar{x})$ where $\bar{x}(t, x) = x + \int_0^t (\mathbf{w}(z) + \mathbf{s}(z))dz$. This transforms the first equation of* (7) *into a new one without the terms with $\mathbf{w}$ and $\mathbf{s}$ on a domain Ω that is moving in time, in the (t, x) variable,with the vector $\int_0^t (\mathbf{w}(z) + \mathbf{s}(z))dz$. This is interesting in order to reduce the size of the domain to be considered, since in the new problem, we only need to compute the effects of the diffusion and the transport/reaction due to the pump.*

4 Numerical approximation

The Finite Volume method is well suited for the space-time discretization of problem (7).

For the numerical approximation of (7), given $I, J \in N$ we divide the spatial domain $\Omega = (x_{1,\min}, x_{1,\max}) \times (x_{2,\min}, x_{2,\max})$ into control volumes $\Omega_{i,j}$. For $i = 1, \ldots, I$; $j = 1, \ldots, J$, we define

$$\begin{aligned} \Omega_{i,j} = (x_{1,\min} + (i-1)\Delta x_1, x_{1,\min} + i\Delta x_1) \\ \times (x_{2,\min} + (j-1)\Delta x_2, x_{2,\min} + j\Delta x_2), \end{aligned} \tag{12}$$

with $\Delta x_1 = \dfrac{x_{1,\max} - x_{1,\min}}{I}$, $\Delta x_2 = \dfrac{x_{2,\max} - x_{2,\min}}{J}$. We define $\Delta t = \dfrac{T}{N}$, where $N \in \mathbb{N}$ is the number of time steps.

Considering a fully implicit time discretization of the backward Euler type for the time discretization of (7) and an upwind scheme for the transport term, one obtains at $t = n\Delta t$ on the cell $\Omega_{i,j}$, for $i = 1, \ldots, I$ and $j = 1, \ldots, J$, the following scheme:

$$C^0_{i,j} = C_0(\xi_{i,j}), \;\; \xi_{i,j} \text{ being the center of cell } \Omega_{i,j}; \tag{13}$$

for $n \geq 0$ we compute $\{C^n_{i,j}\}$ from $\{C^{n-1}_{i,j}\}$ using :

$$\begin{aligned}
&\frac{C^n_{i,j} - C^{n-1}_{i,j}}{\Delta t} + 2\left(\frac{k_1}{(\Delta x_1)^2} + \frac{k_2}{(\Delta x_2)^2}\right) C^n_{i,j} \\
&- \frac{k_1}{(\Delta x_1)^2}\left(C^n_{i+1,j} + C^n_{i-1,j}\right) - \frac{k_2}{(\Delta x_2)^2}\left(C^n_{i,j+1} + C^n_{i,j-1}\right) \\
&+ \frac{1}{\Delta x_1}[\max(0, V^n_{1,i,j-\frac{1}{2}})C^n_{i,j} + \min(0, V^n_{1,i,j-\frac{1}{2}})C^n_{i+1,j} \\
&\qquad - \max(0, V^n_{1,i-1,j-\frac{1}{2}})C^n_{i-1,j} - \min(0, V^n_{1,i-1,j-\frac{1}{2}})C^n_{i,j}] \\
&+ \frac{1}{\Delta x_2}[\max(0, V^n_{2,i-\frac{1}{2},j})C^n_{i,j} + \min(0, V^n_{2,i-\frac{1}{2},j})C^n_{i,j+1} \\
&\qquad - \max(0, V^n_{2,i-\frac{1}{2},j-1})C^n_{i,j-1} - \min(0, V^n_{2,i-\frac{1}{2},j-1})C^n_{i,j}] \\
&+ \frac{2\pi R_p Q}{\Delta x_1 \Delta x_2} C^n_{i_p,j_p} \chi^{p,n}_{i,j} = 0,
\end{aligned} \tag{14}$$

where in (14)

- $C^n_{k,l} = 0$ if $k = 0$ or $I + 1$ (respectively, $l = 0$ or $J + 1$),
- $\Omega_{i_{p,n},j_{p,n}}$ is the cell containing $\gamma_p(n\Delta t)$ and $\chi^{p,n}_{i,j} = 0$ if $\{i, j\} \neq \{i_{p,n}, j_{p,n}\}$, $\chi^{p,n}_{i,j} = 1$ if $\{i, j\} = \{i_{p,n}, j_{p,n}\}$,
- $\mathbf{V}(x,t) = (V_1(x,t), V_2(x,t)) = \mathbf{w}(x,t) + \mathbf{s}(x,t) + \mathbf{p}(x,t)$, where $x \in \Omega$ and $t \in [0, T]$.
- $V^n_{1,i,j-\frac{1}{2}} = V_1((x_{1,\min} + i\Delta x_1, x_{2,\min} + (j - \frac{1}{2})\Delta x_2), n\Delta t)$,
- $V^n_{2,i-\frac{1}{2},j} = V_2((x_{1,\min} + (i - \frac{1}{2})\Delta x_1, x_{2,\min} + j\Delta x_2), n\Delta t)$.

System (13)-(14) can be rewritten in the form:

$$\mathbf{A}\mathbf{x} = \mathbf{b}, \quad with \quad \mathbf{x} = \mathbf{C}^n, \quad \mathbf{b} = \mathbf{C}^{n-1}, \tag{15}$$

where $\mathbf{A}$ is a $IJ \times IJ$-matrix and $\mathbf{C}^n$ and $\mathbf{C}^{n-1}$ are vectors of dimension IJ corresponding to the concentration matrices $(C^n_{i,j})$ and $(C^{n-1}_{i,j})$ stored column wise.

The solution of the linear system (15) is obtained by using a Bi-Conjugate gradient type algorithm (Bi-CG), well suited for non-symmetric matrices such as $\mathbf{A}$. A particular parallel implementation of this algorithm is presented in Section 5.

Remark 6 *: In Section 6.2.1, we will discuss the advantages of using the implicit instead of explicit schemes.*

Remark 7 *: To space discretize the advecting term* $\nabla \cdot c\mathbf{V}$ *we have employed a first order upwinding scheme.*

Remark 8 *: For a thorough discussion of finite volume methods for the solution of partial differential equations see [2].*

5 Parallel Bi-Conjugate Gradient type method

In order to choose the numerical method appropiate to solve system (15), we have taken into account the fact that A is a non-symmetric matrix, that the sea region under consideration may be quite large and thus that the space discretization might produce a large dimensional problem that requires large storage space, and that the time step necessary for stability might increase the computational time.

The generalized minimal residual method GMRES, is a suitable method for nonsymmetric systems, but it generates long recurrences in order to keep all residuals orthogonal, at the cost of increasing storage demand.

The biconjugate gradient method (Bi-CG) first developed by Lanczos [10] and further studied by Fletcher [4], takes another approach, replacing the orthogonal sequence of residuals by two mutually orthogonal sequences, at the price of no longer providing a minimization of the residuals as the GMRES. However, it has theoretical properties similar to the CG method, and returns the exact solution in at most IJ iterations (for exact computations) and can be seen as a direct method. If A is symmetric, it produces the same iterates x_k as CG but at twice the computational cost.

In terms of number of iterations, the Bi-CG method is comparable to GMRES [5] and attains similar accuracy. The generation of the basis for Bi-CG is cheap and thus it requieres low memory. Considering that Bi-CG needs to compute two matrix-vector multiplications (instead of one with GMRES), an efficient way to perform the matrix-vector products has to be designed to make the method efficient. (See also ([16])).

Taking into account the above considerations, we have chosen the Bi-CG method to use in our problem, and to make it efficient we have implemented on a Distributed Parallel computer. In fact, to get smooth convergence and better accuracy, a stabilization method called Bi-CGSTAB developed by Van der Vost [13, 12], was employed.

5.1 Bi-CG and Bi-CGSTAB methods

The induction relations used to update the residuals are constructed so that the original residuals $\mathbf{r_j} = \mathbf{b} - A\mathbf{x_j}$ are bi- orthogonal with respect to the residuals of another system $A^T\,\tilde{\mathbf{x}} = \tilde{\mathbf{b}}$, $\tilde{\mathbf{r}}_\mathbf{j} = \tilde{\mathbf{b}} - A^T\tilde{\mathbf{x}}_\mathbf{j}$. This bi-orthogonality of the residuals can be accomplished by two 3-term update or induction relations. There are

also relations to update $\mathbf{x_j}$ and $\tilde{\mathbf{x}}_\mathbf{j}$ and the new directions $\mathbf{p_j}$ and $\tilde{\mathbf{p}}_\mathbf{j}$, which are also orthogonal to the residuals, $\tilde{\mathbf{p}}_\mathbf{i}^\mathbf{T}\ \mathbf{r_j}\ =\ 0$ and $\tilde{\mathbf{r}}_\mathbf{j}^\mathbf{T}\ \mathbf{p_k}\ =\ 0$.

The two directions are conjugate, that is $\tilde{\mathbf{p}}_\mathbf{i}^\mathbf{T}\ A\ \mathbf{p_j}\ =\ 0,\ i\ \neq\ j$ and the residuals are orthogonal, that is $\tilde{\mathbf{r}}_\mathbf{j}^\mathbf{T}\ \mathbf{r_j}\ =\ 0,\ i\ \neq\ j$. Choosing $\mathbf{p_0}\ =\ \mathbf{r_0}$ and $\tilde{\mathbf{p}}_\mathbf{0} = \tilde{\mathbf{r}}_\mathbf{0}$, the method terminates in at most IJ steps.

To get smooth convergence, better precision and a faster method, the Bi-CGSTAB method was used. We derive it now.

The residuals can be written as $\mathbf{r_j}\ =\ P_j(A)\ \mathbf{r_0}$ and $\tilde{\mathbf{r}}_\mathbf{j}\ =\ P_j(A^T)\ \tilde{\mathbf{r}}_\mathbf{0}$. The polynomial $P_j(A)$ should reduce $\mathbf{r_0}$, (that depends on the initial $\mathbf{x_0}$), but it might not reduce any other vector, including $P_i(A)\ \mathbf{r_0}$, which is needed to obtain

$$(P_i(A)\ \mathbf{r_0}, P_j(A^T)\ \tilde{\mathbf{r}}_\mathbf{0})\ =\ (P_j(A)\ P_i(A)\ \mathbf{r_0},\ \tilde{\mathbf{r}}_\mathbf{0})\ =\ 0, \quad for\ \ i < j.$$

To avoid this possible irregularity, the residuals can be written as

$$\mathbf{r_j} =\ Q_j(A)\ P_j(A)\ \mathbf{r_0} \quad where \quad Q_i(x) = (1-\omega_1\ x)(1-\omega_2\ x)\ldots(1-\omega_i\ x)$$

Constants ω_j are chosen to minimize the residuals $\mathbf{r_j}$, in the j-iteration. Due to the orthogonality property $(P_i(A)\ \mathbf{r_0}, Q_j(A^T)\ \tilde{\mathbf{r}}_\mathbf{0})\ = 0$, if $j < i$, we get finite termination in at most IJ steps. The Bi-CGSTAB algorithm follows:

- given $\mathbf{x_0}$ (the initial approximation), compute $\mathbf{r_0} = \mathbf{b} - \mathbf{A}\mathbf{x_0}$;
- take an arbitrary vector $\tilde{\mathbf{r}}_\mathbf{0}$ such that $(\tilde{\mathbf{r}}_\mathbf{0}, \mathbf{r_0}) \neq 0$, like $\tilde{\mathbf{r}}_\mathbf{0} = \mathbf{r_0}$;
- take $\rho_0 = \alpha = \omega_0 = 1$ and $\mathbf{v_0} = \mathbf{p_0} = 0$;

  for i=1 until convergence

  - $\rho_i = (\tilde{\mathbf{r}}_\mathbf{0}, \mathbf{r_{i-1}})$;
  - $\beta = (\frac{\rho_i}{\rho_{i-1}})(\frac{\alpha}{\omega_{i-1}})$;
  - $\mathbf{p_i} = \mathbf{r_{i-1}} + \beta(\mathbf{p_{i-1}} - \omega_{i-1}\mathbf{v_{i-1}})$;
  - $\mathbf{v_i} = A\mathbf{p_i}$;
  - $\alpha = \dfrac{\rho_i}{(\tilde{\mathbf{r}}_\mathbf{0}, \mathbf{v_i})}$;
  - $\mathbf{s} = \mathbf{r_{i-1}} - \alpha\mathbf{v_i}$;
  - norm= $(\mathbf{s}, \mathbf{s})$;

    If norm $\leq tol1$ set $\mathbf{x_i} = \mathbf{x_{i-1}} + \alpha_i\mathbf{p_i}$, **stop** ;

    $-\ \mathbf{t} = A\mathbf{s}$;

    $-\ \omega_i = \dfrac{(\mathbf{t}, \mathbf{s})}{(\mathbf{t}, \mathbf{t})}$;

    $-\ \mathbf{x_i} = \mathbf{x_{i-1}} + \alpha_i\mathbf{p_i} + \omega_i\mathbf{s}$;

    If $\|x_i - x_{i-1}\| \leq tol2$, **stop** ;

    - set $\tilde{\mathbf{r}}_\mathbf{i} = s - \omega_i\mathbf{t}$;

  end for

- $\mathbf{x_i}$ is the approximation to the solution.

In this algorithm two stopping conditions have been used, with two tolerances $tol1$ and $tol2$ that depend on the desired accuracy for the solution.

With Bi-CGSTAB, besides the two matrix vector products also needed in Bi-CG, four inner products have to be computed,instead of two as in the original method. However, this extra work will be compensated by the decrease in the number of iterations and the smoother convergence behavior. In addition, four additional IJ-vectors have to be stored $\tilde{\mathbf{r}}_\mathbf{0}, \mathbf{p}, \mathbf{v}$ and $\mathbf{t}$. Some break down situations may occur when either ρ_i or $(\tilde{\mathbf{r}}_\mathbf{0},\ \mathbf{v_i})$ become very small, and this situation has to be checked in the algorithm. Eventually, another initial $\tilde{\mathbf{r}}_\mathbf{0}$ has to be chosen.

5.2 Parallelization of the Bi-CGSTAB

In this paper, we are presenting results from a straight forward parallelization method, testing both row and column wise algorithms to perform the matrix-vector multiplications and the inner products. A more complex parallelization alternative, based on the very recent work of Tong et al.2009, [8], is now in progress and will be reported in a future paper.

In order to parallelize Bi-CGSTAB, we consider the following methodology:

The IJ rows (or columns, depending on the selected parallelization method) of matrix $\mathbf{A}$ are distributed equally to the N_{proc} number of processors. If IJ is not a multiple of N_{proc}, the $N_f = (IJ \text{ modulus } IJ_{\text{proc}})$ remaining rows will be send to the first N_f processors. We use a coordinate system to store the structured matrix $\mathbf{A}$.

To generate the initial $\mathbf{x_0}$ for the Bi-CGSTAB algorithm, the polluted areas in the discretized space are assumed to be circular spots with predefined center and radius. The vector $b^0 = C^0_{i,j}$ represents the concentration of oil in the rectangular grid at the initial time. In this case, it will take values 0 or 1 depending on whether the point (i, j) in the grid is polluted or not and every processor will need to update its components of the right hand side $b^n = C^{n-1}_{i,j}$ (the current concentration), at every time step.

To test the best alternative, the computation of the two matrix-vector products (of the form $\mathbf{Ax}$) needed in the Bi-CGSTAB algorithm, will be performed Column and Row wise.

Column wise method: The matrix vector multiplication can be expressed as a linear combination of the columns $\mathbf{A}_i$ of matrix $\mathbf{A}$, with the coefficients being the elements $\mathbf{x}_i$ of vector $\mathbf{x}$. The result of this linear combination, is the sum of vectors $x_i\mathbf{A_i}$ where $\mathbf{A_i}$ is the i-th column of $\mathbf{A}$. However as the columns are stored in different processors, the computation of the linear combination needs to be done among processors, with communication of the partial sums of the IJ-vectors $x_i\mathbf{A_i}$. The complete sum can be performed using a Fan-in process (see for instance [7]) or the MPI-Allreduce (see for instance [11]) and they had two different computing times. The final vector $\mathbf{Ax}$ will be stored in the root processor and needs to be redistributed for further operations. The lenght of the vectors to be send to the processors is IJ.

Row wise method: Each processor has only part of $\mathbf{x_i}$ and the entire vector is needed to compute $\mathbf{Ax}$. Then, this information has to be communicated to all processors. However, the length of this information depends on the number of rows per proccessor, and is then much smaller than the one used in the column wise communicated data. Therefore as the number of processors increases the length of the data to be communicated decreases. This is main reason why this alternative is more efficient. Thus the matrix vector multiplication $\mathbf{Ax}$ is performed in perfect parallelism, where $\mathbf{Ax}$ remains distributed without any further communication.

The four inner products of the Bi-CGSTAB algorithm (of the form (y, z)) are performed using a row wise method and communication is again needed to compute the final sum, and to send the result back to the processors.

The tests performed using both alternatives will be presented in Section 6.2.2, and they show clearly the superiority of the row wise method, which will then be used to perform the numerical simulations presented in Section 6.2.3.

6 Numerical experiments

6.1 Parameters

The model parameters are set in the following way (using the SI unit system):

The modeling domain Ω is defined by $x_{1,\min} = 0$, $x_{1,\max} = 2\times10^4$, $x_{2,\min} = 0$ and $x_{2,\max} = 2 \times 10^4$.

The simulation time is equal to one day, $T = 86400$.

We consider 4 alternative spatial discretization meshes (I, J): $(50, 50)$, (100, 100), $(200, 200)$ or $(300, 300)$.

The time step is $\Delta t = 86.4$ (i.e. N =1000). This choice is discussed in Section 6.2.1.

The diffusion coefficients are $k_1 = k_2 = 0.5$.

The wind plus sea velocity field $\mathbf{s}(x, t) + \mathbf{w}(x, t)$ is defined by

$$\Big(\frac{x_1}{4x_{1,\max}}\cos(\frac{\pi t}{3600}), \frac{x_2}{4x_{2,\max}}\sin(\frac{\pi t}{3600})\Big), \tag{16}$$

for $t \in [0, T]$ and $x = (x_1, x_2) \in \Omega$.

The pump parameters are $Q = 100$ and $R_p = 1$. We consider two different trajectories of the pumping ship:

- Trajectory **T1**: The initial position of the pumping ship is (600, 1400) and its trajectory is given by

$$\gamma_p(t) = (x_{1,\max}(0.5 + 0.2\cos(\frac{\pi t}{86400})), x_{2,\max}(0.3 + 0.1\sin(\frac{\pi t}{86400}))). \tag{17}$$

- Trajectory **T2**: The initial position of the pumping ship is 800, 1400) and its trajectory is given by

$$\gamma_p(t) = (x_{1,\max}(0.5 + 0.2\cos(\frac{\pi t}{21600})), x_{2,\max}(0.4 + 0.1\sin(\frac{\pi t}{86400}))). \tag{18}$$

The two pumping ship trajectories **T1** and **T2** are depicted in Figures 5 and 6.

The initial pollutant concentration, presented in Figure 1, is given by

$$c(x,0) = \chi_{B((8000,8000),1200)}(x) + \chi_{B((8000,12000),1200)}(x), \tag{19}$$

where $\chi_{B(b,a)}$ is defined in (9).

The stopping conditions used for the Bi-CGSTAB Method is $tol1 = tol2 = 10^{-6}$.

For all computations, we have used the `BlueGene/L System` from the `Dares -bury Laboratory` in England. We have used up to $N_{\text{proc}} = 32$ bi-cores processors. Each processor has a 32-Bit architecture and 1 GB of local memory. The code is programmed in Fortran 90 and uses the MPI standard protocol. Double precision values were used in all computations. All runs were performed using the row wise method. For the mesh 200×200 we have also tested the Fan-in and reduced column wise methods to illustrate the comparison between the parallelization methods.

6.2 Results

In this Section, we first present a comparison between the implicit scheme (14) and its corresponding fully explicit version. Then we report the results related to the parallelization schemes. Finally, we analyze and discuss the behavior of our model when modifying the time and space discretization steps.

6.2.1 Comparison between implicit and explicit schemes

In order to check the efficiency of the implicit scheme (14) and the corresponding explicit one, we compare their stability. To do this, we consider the trajectory **T2** (similar results have been obtained with **T1**), the meshes 50×50, 100×100, 200×200 and 300×300 and the time steps $\Delta t = 8640$, 1728, 864, 345.6, 172.8, 115.2, 86.4, 17.28, and 8.64 (i.e. N =10, 50, 100, 250, 500, 750, 1000, 5000 and 10000, respectively). The quantity of pollutant pumped during the simulation processes are shown in Table 2 for the implicit scheme.

When $N \geq 250$, the implicit scheme was stable. Furthermore, the time step $\Delta t = 86.4$ (i.e. N =1000) gives a good compromise between computational complexity and precision. Indeed, the solutions obtained with this time step and $\Delta t = 8.64$ (i.e. N =10000) exhibit a difference on the quantity of pollutant pumped inferior to 0.6%, which is quite acceptable. This time step value is used in all computations in Sections 6.2.2 and 6.2.3.

On the other hand, the explicit scheme is always unstable for those values of the discretization. It is well known that the explicit schemes demand time step restrictions of CFL type (see [6, 9]). More precisely, for all $t \in [0,T]$ and $x \in \Omega$, Δt should satisfy

$$\Delta t \quad < \frac{\Delta x_1 \Delta x_2}{\Delta x_2 |\mathbf{V}_1(x,t)| + \Delta x_1 |\mathbf{V}_2(x,t)|}. \tag{20}$$

In the considered numerical tests, for all $t \in [0,T]$ and $x \in \Omega$ we have

$$|\mathbf{V}_1(x,t)| + |\mathbf{V}_2(x,t)| < (Q + \frac{1}{4})(\sin(\frac{\pi}{4}) + \cos(\frac{\pi}{4})) < 142. \tag{21}$$

Hence, we should set Δt <2.80, 1.40, 0.70 and 0.46 (i.e. $N > 30858$, 61715, 123429 and 187827, respectively), when using the 50×50, 100×100, 200×200 and 300×300 meshes, respectively.

Thus, the explicit scheme needs time steps much smaller than the implicit one, especially for the most refined meshes, and it is then less interesting from a computational point of view.

Remark 9 *The reaction term added on the right hand side of the first equation of* (7)*, i.e.* $-(2Qc\chi_{B(\gamma_p(t),R_p)}(x))/R_p$*, also impose a restriction on* Δt *when considering the explicit scheme. Indeed, the explicit discrete version of this term, i.e.* $(2\pi R_p Q C^{n-1}_{i_p,j_p}\chi^p_{i,j})/(\Delta x_1 \Delta x_2)$ *(using the same notations as in* (14)*), needs time steps satisfying* $\Delta t < (\Delta x_1 \Delta x_2)/(2\pi R_p Q)$ *in order to ensure the positivity of the solution and the stability of the scheme. More precisely, we must set* $\Delta t < 254,63,15$ *and 7 (i.e.* $N > 341$*, 1371, 5760 and 12343, respectively) when considering the* 50×50*,* 100×100*,* 200×200 *and* 300×300 *meshes, respectively. Similar results have been obtained for other advection-reaction-diffusion equations, see for example [14, 1].*

N	50×50	100×100	200×200	300×300
10	55.37	54.54	unstable	unstable
50	64.68	66.87	64.90	unstable
100	67.15	69.04	68.87	unstable
250	69.74	71.11	70.11	69.99
500	70.32	71.76	70.78	70.40
750	70.75	71.99	70.92	70.50
1000	70.82	72.10	71.00	70.54
5000	71.15	72.39	71.17	70.64
10000	71.2	72.42	71.19	70.65

Table 2: Quantity of pollutant pumped, expressed in % with respect to the initial pollutant quantity, observed during the simulation process considering the trajectory **T2**, the meshes 50×50, 100×100, 200×200 and 300×300 and the time steps associated to the given values of N for the implicit discretization scheme. We also report in this table the unstable results.

6.2.2 Parallelization schemes

As described in Section 5.2, the matrix-vector multiplication of the Bi-CGSTAB has been performed using column (combined with Fan-in or All-Reduced summation method) and row wise schemes. The comparison of these methods, when

applied to our model, is reported in Table 3 and in Figure 2 only for trajectory **T1**, results for **T2** being similar.

N_{proc}	Fan-In Col.	All-Reduced Col.	Row
1	178.2	178.0	164.0
4	98.4	79.5	57.7
8	94.4	62.7	39.3
16	100.8	54.0	30.4
32	110.1	49.9	26.1

Table 3: Time, in seconds, needed to solve the model considering trajectory **T1** and using Fan-In, All-Reduced column and row wise methods in function of N_{proc} for mesh 200×200.

These results show the advantage of using the row wise scheme, and if column wise is used, they indicate the clear advantage of the All Reduced algorithm. From now on, all results reported were performed in the row wise mode.

An important issue is the degree of discretization and the computational time needed to accomplish a given accuracy in the model solution. Therefore, we report and compare the total time, in seconds, needed to perform the simulation in the prefixed time interval (here, one day), for all selected discretization meshes and for $N_{\text{proc}} = 1, 4, 8, 16, 32$. These results are presented in Table 4 and in Figure 3.

N_{proc}	50×50	100×100	200×200	300×300
1	32.3	78.9	164.0	377.4
4	10.6	27.5	57.7	120.9
8	7.6	19.1	39.3	76.9
16	5.9	15.0	30.4	55.4
32	5.3	12.9	26.1	45.0

Table 4: Time, in seconds, needed to solve the model with respect to N_{proc} (line) and the discretization mesh (column).

The computational time is reduced from six up to eight times when using from 1 to 32 processors. As we can observe in Figure 3, the largest gain is obtained from 1 to 16 processors.

6.2.3 Model analysis

In Table 5, we report the percentage of the pollutant pumped for the three meshes considered and for T1 and T2. To be able to determine which mesh is the most efficient (in computational time and precision) for the trajectories considered here, we compute the average and maximum difference (in time) between the amount of pumped pollutant obtained with the finest mesh ($300 \times$

300) and the other meshes. We observe that the accuracy increases with the level of discretization. However, we point out, that the final concentrations obtained with the two finest meshes show similar amount of pumped pollutant (with a mean and maximum difference less than 1.0% and 2.4%, respectively). This makes clear that we need to use fine meshes only up to certain degree (as 200×200 is numerically close to 300×300 and is four times faster).

Part of the previous differences between the amount of pumped pollutant obtained with the finest mesh and the other meshes, is due to the artificial diffusion of the numerical scheme (14). This can be observed in Figure 4 that depicts the final concentration, for each mesh, considering the trajectory **T2** (results are similar using **T1**). This artificial diffusion phenomena increases with the spatial block size.

In Figures 5 to 6, we show the evolution of the concentration of the pollutant considering the trajectories **T1** and **T2** at different times for the mesh 300×300. These figures and Table 5 show the impact of the trajectory in the amount of pumped pollutant, pointing out the need to find optimal trajectories to pump a maximum quantity.

Traj.	Poll. pump.	50×50	100×100	200×200	300×300
T1	Poll. pump.	84.06	93.01	97.54	98.59
	Mean Diff.	9.99	4.09	0.89	—
	Max Diff.	19.50	9.33	2.32	—
T2	Poll. pump.	70.81	72.09	71.00	70.54
	Mean Diff.	2.27	1.40	0.35	—
	Max Diff.	6.90	3.00	1.20	—

Table 5: Quantity of pollutant pumped (Poll. pump.) (expressed in % respecting to the initial pollutant quantity), Mean difference (Mean Diff.) and maximum difference (Max Diff.) (in % with respect to the solution obtained with the mesh 300×300) observed during the simulation considering the trajectories (Traj.) **T1** and **T2** and the meshes 50×50, 100×100, 200×200 and 300×300.

7 Conclusions and future work

In this paper we have presented a two dimensional mathematical model of the motion of oil spots in the open sea, taking into account the velocity and direction of the sea currents, the wind, and the advection and reaction due to a pumping process, carried out by a ship with a fixed trajectory.

The implicit and explicit numerical schemes used in this article to perform the simulation of the model, are finite volume schemes, implemented in parallel on a cluster with up to 32 processors. The results obtained, show clearly that the row wise method used to solve the linear systems associated with the discrete model, is highly superior to the column wise method.

Also, the efficiency obtained in the parallelization is improved when increasing the number of processors, showing the parallel code we developed has the expected scaling properties.

The results presented in this paper show the importance of four features to be considered:

- The need to use an implicit scheme to get a computationally efficient algorithm.
- The need to control the propagation of the diffusion (avoiding infinite speed and artificial diffusion) by replacing the linear diffusion term in (3) by a nonlinear one.
- The need to use fine meshes only up to certain degree in the simulation.
- The need to find the optimal trajectories of the pumping ship, so that the amount of pollutant pumped in a fixed period of time is maximized. The optimization process will require a large number of evaluations of the model and thus the necessity to develop robust and efficient simulators. The results in this article are a first step in that direction.

Another important issue to be investigated is the possibility of combining explicit and implicit schemes for speeding-up the computational time. We will investigate splitting and un-split schemes and this might lead us to use other upwinding scheme like superbee [17] to discretize the advection terms in the model.

Acknowledgements

This work has been done in the framework of project MTM2008-04621 of the Spanish Ministry of Science and Innovation, the Research Group MOMAT (Ref. 910480) supported by Banco Santander and UCM and the PASPA program from the national university of Mexico (UNAM). The authors wish to thank Doctor David Emerson who kindly help us to get access to the parallel machine (`BLUEGENE`) at the `Daresbury Laboratory` in England. This is also a consequence of an ALFA project, Scientific Computing Advanced Training (SCAT) 2005-2008. The authors would like to thank Nelson Del Castillo for its valuable help during this work.

References

[1] M.P. Calvo, J. de Frutos, J. Novo (2001) *Linearly implicit RungeKutta methods for advection-reaction-diffusion equations.* Applied Numerical Mathematics 37(4): 535549.

[2] R. Eymard, T. Gallouet, R. Herbin and A. Michel (2002) Convergence of a finite volume scheme for nonlinear degenerate parabolic equations. newblock *Numer. Math.*, 92(1):41–82.

[3] Faber, V. and Manteuffel T. (1984) *Necessary and Sufficient Conditions for the Existence of a Conjugate Gradient Method* SIAM J. Numer. Anal. 21, 315–339.

[4] R. Fletcher (1976) *Conjugate Gradient Methods for Indefinite Systems*, in Proc. Dundee Conference on Numerical Analysis, Lecture Notes in Mathematics 506, G. A. Watson, ed., Springer-Verlag, Berlin, 73-89.

[5] R. Freund and N. Nachtigal (1991), *QMR: A Quasi-Minimal Residual Method for Non-Hermitian Linear Systems* Numer. Math. 60, 315–339.

[6] R. Glowinski, P. Neittaanmaki (2008) *Partial Differential Equations. Modelling and Numerical Simulation.* Series: Computational Methods in Applied Sciences, 16, Springer.

[7] G. Golub, J.M. Ortega (1993) *Scientific Computing: An Introduction with Parallel Computing.* San Diego, CA: Academic Press.

[8] T. Gu , X. Zuo , X. Liu , P. Li (2009) *An improved parallel hybrid biconjugate gradient method suitable for distributed parallel computing*, Journal of Computational and Applied Mathematics, 226, 1, 55–65.

[9] W. Hundsdorfer, J.G. Verwer (2003) *Numerical Solution of Time-Dependent Advection-Diffusion-Reaction Equations* Springer Series in Comput. Math. 33.

[10] C. Lanczos (1952) *Solution of Systems of Linear Equations by Minimized Iterations*, J. Res. Natl. Bur. Stand. 49, 33–53.

[11] J.G. Lewis, D.G. Payne, R.A. van de Gejin (1993) *Matrix-Vector Multiplication and Conjugate Gradient Algorithms on Distributed Memory Computers.* Proceedings of Supercomputing'93, 15–19, Portland, OR.

[12] G.L.G. Sleijpen and H.A. van der Vorst (1995) *Maintaining convergence properties of BiCGstab methods in finite precision arithmetic*, Numerical Algorithms, 10, 203–223.

[13] H. A. Van der Vorst (1992) *Bi-CGSTAB : A Fast and Smoothly Converging Variant of Bi-CG for the Solution of Nonsymmetric Linear Systems*, SIAM J. Sci. Stat. Comput., 13:631–644.

[14] J.G. Verwer, B.P. Sommeijer, W. Hundsdorfer,(2004) *RKC time-stepping for advection-diffusion-reaction problems*, Journal of Computational Physics 201: 61–79.

[15] V. Voevodin (1983) *The Problem of Non-Self-Adjoint Generalization of the Conjugate Gradient Method is Closed* U.S.S.R. Comput. Maths. and Math. Phys. 23, 143–144.

[16] Q. Ye (1991) *A convergence analysis of nonsymmetric Lanczos algorithms*, Math. Comp. 56, 677–691 (1991).

[17] R.J. LeVeque (2002) Finite Volume Methods for Hyperbolic Problems. *Cambridge University Press; 1 edition* .

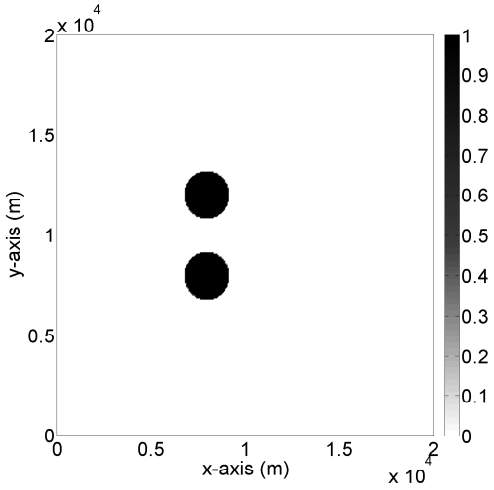

Figure 1: Initial position of the pollutant spots (in black) in the domain Ω for the 300×300 mesh.

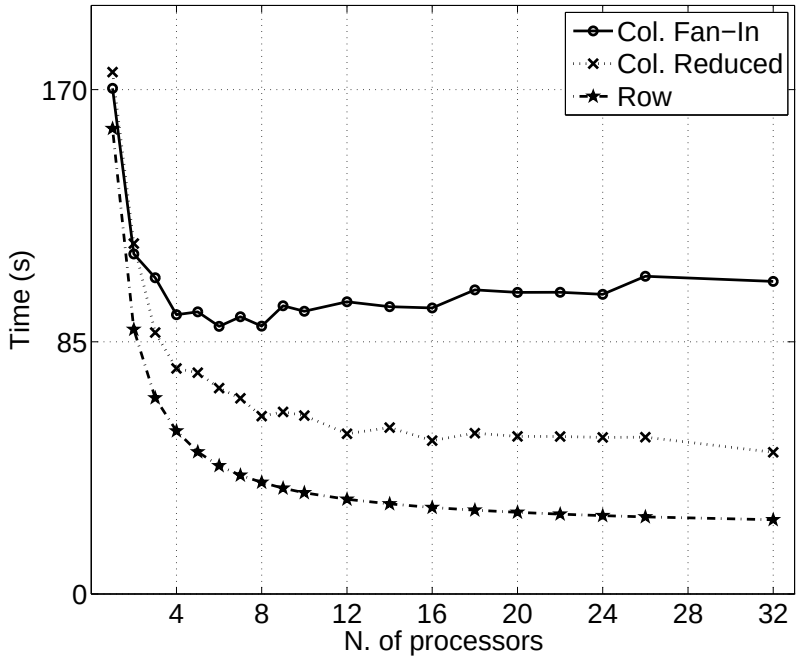

Figure 2: Total time, in seconds, using Fan-In (–) and Reduced (..) Column Wise and Row Wise (-.) methods Vs number of processors for mesh 200×200.

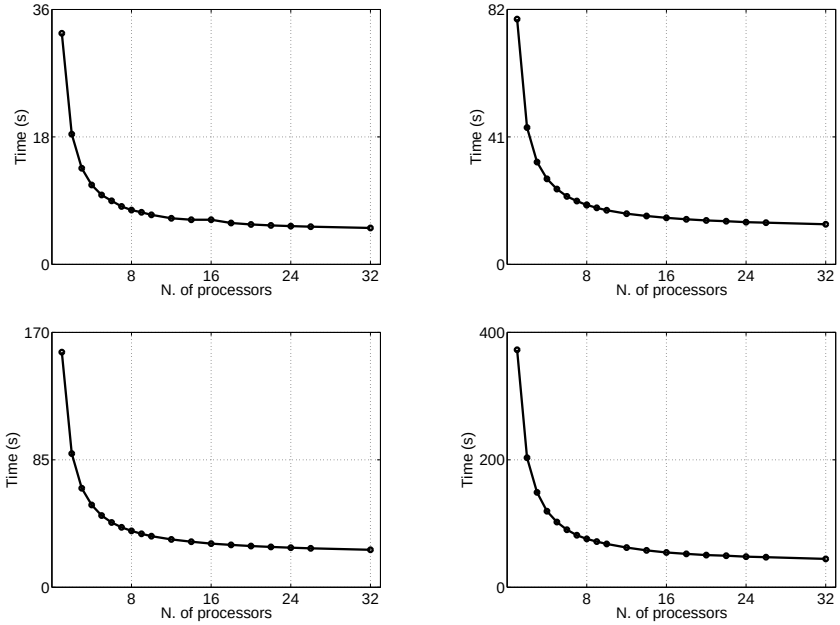

Figure 3: Total time (s) vs number of processors for meshes (**Top-Left**) 50×50, (**Top-Right**) 100×100, (**Bottom-Left**) 200×200 and (**Bottom-Right**) 300×300.

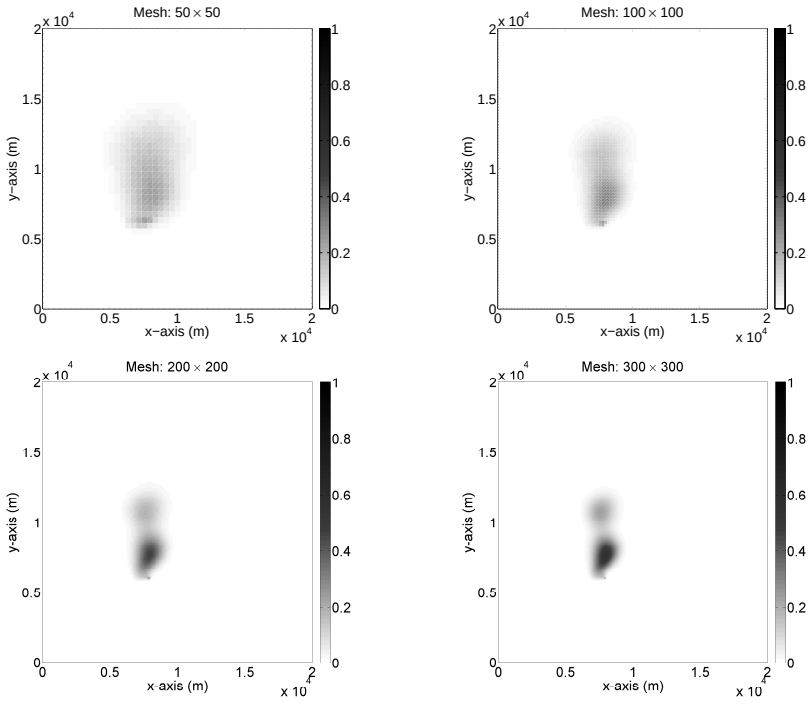

Figure 4: Final concentration obtained considering the trajectory **T2** for the meshes (**Top-Left**) 50×50, (**Top-Right**) 100×100 , (**Bottom-Left**) 200×200 and (**Bottom-Right**) 300×300.

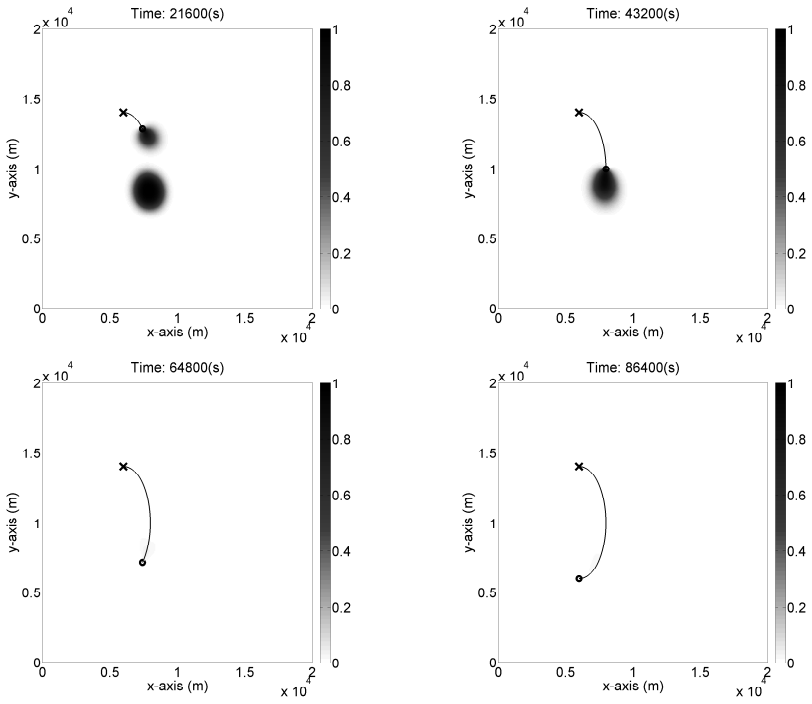

Figure 5: The evolution of the concentration considering the trajectory **T1** for the mesh 300×300 at times (**Top-Left**) 21600 s, (**Top-Right**) 43200 s, (**Bottom-Left**) 64800 s and (**Bottom-Right**) 86400 s. The initial position (**X**), current position (**o**) and trajectory (—) of the pump are also shown.

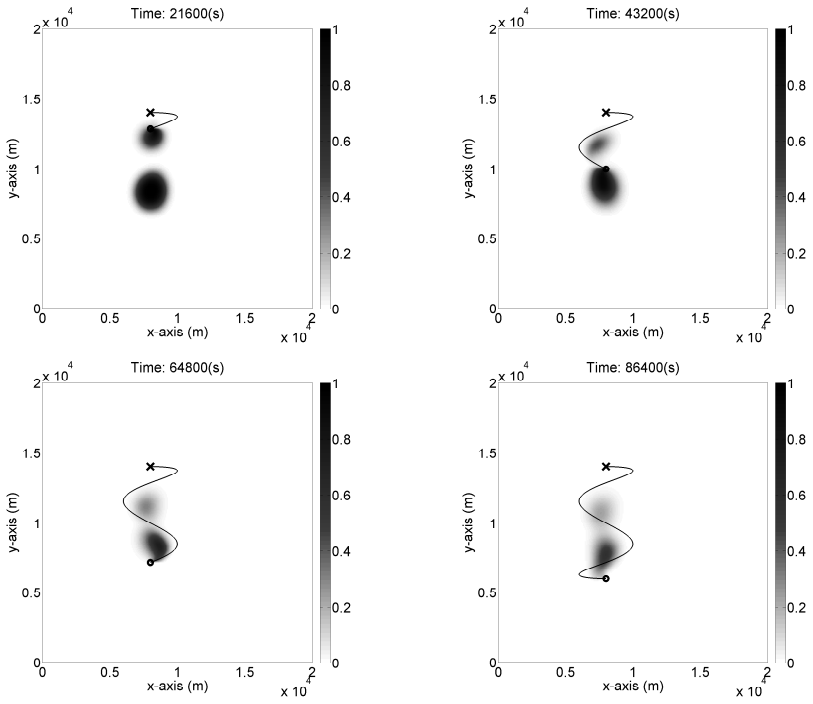

Figure 6: The evolution of the concentration considering the trajectory **T2** for the mesh 300×300 at times (**Top-Left**) 21600 s, (**Top-Right**) 43200 s, (**Bottom-Left**) 64800 s and (**Bottom-Right**) 86400 s. The initial position (**X**), current position (**o**) and trajectory (—) of the pump are also shown.