The Finite Volume Method for Linear Reaction-Diffusion Problems in Two Spatial Dimensions

Michael Muscedere, Carlos Barajas, Matthias K. Gobbert

Department of Mathematics and Statistics, UMBC

Technical Report HPCF–2026–7, hpcf.umbc.edu > Publications

Abstract

We explore the application of the finite volume method (FVM) in a method of lines approach to solving a linear reaction-diffusion equation. The goals are to bring out the key steps in the derivation of the FVM from the conservation form of the partial differential equation to the system of ordinary differential equations of a semi-discrete formulation that need to be solved. In order to illustrate the different techniques needed for control volumes in the interior or touching the boundary of the domain, we use a square domain in two dimensions discretized by a uniform Cartesian mesh. Using the implicit Euler method as concrete example, we also bring out the system of linear equations that needs to be solved at every time step to show its properties. The numerical sample problem is an application model of the spread of pollution with a point source modeled by a Dirac delta distribution that highlights the two key advantages of the FVM to conserve mass in the discretized problem and to handle non-smooth source terms rigorously.

1 Introduction

In this report, we consider a problem that models diffusion of aerosols emitted from a pollution source into a stable environment (no wind), with the emission modeled as a point source in the center of a two-dimensional region on the scale of a city or county. This is sketched in Figure 1.1 as a square domain of size 100 km by 100 km, with a factory with a polluting smokestack indicated at the center. We model the scenario where the source pollutes with a given amount of material, κ kg per hour for the duration of a typical work day of 8 hours and then shuts off. We will simulate the model for 24 hours, beginning when the pollution starts. The simulation starts with no pollution present in the region. The environment is modeled initially as a closed system. That is, we enforce no-flow boundary conditions along the domain boundary; we have showed in [4] how to approximate the effect of an open system by increasing the size of the numerical domain such that the pollution does not reach the edge of the calculation domain and that sub domains can be considered as open systems. That report from the UMBC CyberTraining program (cybertraining.umbc.edu) considered the problem as combining directly **Atmospheric Physics** in the modeled material flow and **High Performance Computing** to enable fast calculations of the problem with increasing domain size by parallelizing a C code with MPI. The area of **Big Data** could be applied to ouput if, for instance, the value of the parameter κ is not assumed to be known. Section 2 provides the mathematical formulation of the model and specifies a test problem used to validate the implementation.

Section 3 provides an introduction to the finite volume method (FVM) as a numerical method that is particularly appropriate to the type of partial differential equation in this report. Here, note that a point source is modeled mathematically by the Dirac delta distribution, which is zero everywhere except the location of the source and attains an infinite value at this location in such a way that the total integral over it has the value 1. Numerical methods such as the *finite difference method* cannot be set up for this model, since the value of the source term is necessarily infinite

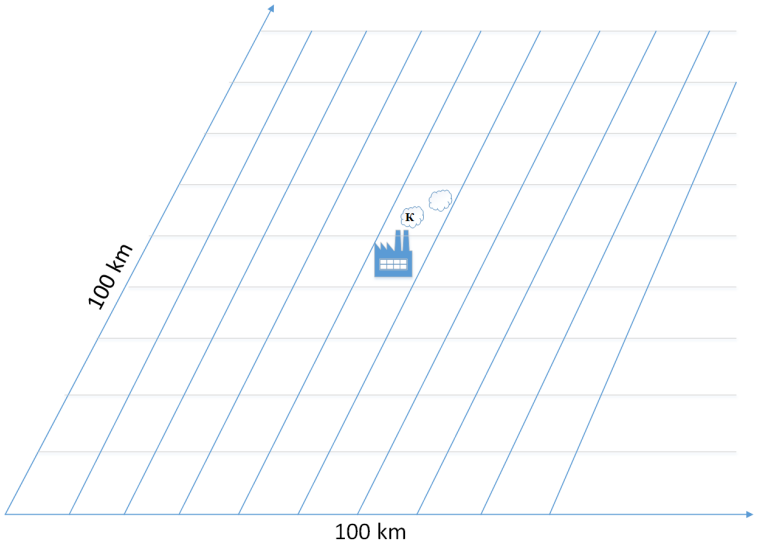

Figure 1.1: Pollution source at center of two-dimensional square region.

at the critical point. The *finite element method* is another alternative for discretizing a measure-valued right-hand side [2], but it does not provide conservation of mass discretely; moreover, finite elements require special approximations to handle advection and would not conserve mass in them, either. By contrast, the *finite volume method* combines the advantages of mass conservation of the discrete approximations with providing a rigorous way to handle the point source; the FVM also handles advection very naturally and with mass conservation.

Section 4 presents the results of simulations. Section 4.1 uses a test problem with smooth source term and a known true solution in closed form. This problem tests the implementation of the FVM and confirms that it converges in agreement with applicable theory. Then, Section 4.2 solves the pollution problem from Section 2.1 with its point source. We show how the mass conservation of the FVM confirms that also this problem is solved correctly, even though the solution is not known in closed form. Section 5 summarizes our conclusions and outlines possible future applications of this model and method.

2 Mathematical Model

The problems considered are described by the partial differential equation (PDE), the boundary condition (BC), and the initial condition (IC), respectively, for the concentration $u(x, y, t)$ of pollution, whose units are kg/km^2,

$$u_t - \nabla \cdot (D \nabla u) = q(x, y, t) \qquad \text{for } (x, y) \in \Omega \text{ and } t > 0, \tag{2.1}$$

$$\mathbf{n} \cdot (D \nabla u) = 0 \qquad \text{for } (x, y) \in \partial\Omega \text{ and } t > 0, \tag{2.2}$$

$$u = u_{ini}(x, y) \qquad \text{for } (x, y) \in \overline{\Omega} \text{ at } t = 0, \tag{2.3}$$

with right-hand side function in the PDE

$$q(x, y, t) = f(x, y, t) + s(x, y, t) \tag{2.4}$$

composed of a smooth source term $f(x, y, t)$ and a point source term $s(x, y, t)$; these terms differentiate the problems being solved and are specified in the subsections below. We use kilometers km for units on the spatial domain region $\Omega = (-50, 50) \times (-50, 50) \subset \mathbb{R}^2$ and hours h for the time domain where the time ranges $0 \le t \le 24$. In the PDE (2.1), the diffusivity coefficient D is a positive constant scalar in km^2/h. In the BC (2.2), the vector $\mathbf{n} \equiv \mathbf{n}(x, y) \in \mathbb{R}^2$ denotes the unit outward normal vector at $(x, y) \in \partial\Omega$. In IC (2.3), setting $u_{ini} \equiv 0$ models the situation of no pollution present at the initial time $t = 0$. The variables and parameters and their units are collected in Table 2.1.

Table 2.1: Variables of the model and their units.

Variable	Definition	Units
x	spatial position variable	km
y	spatial position variable	km
t	time variable	h
$u(x, y, t)$	concentration of pollutant	kg/km^2
D	diffusivity coefficient	km^2/h
$q(x, y, t)$	right-hand side source term	kg km^{-2} h^{-1}
$f(x, y, t)$	smooth source term	kg km^{-2} h^{-1}
τ	time scale of steady state approach	h
$s(x, y, t)$	point source term	kg km^{-2} h^{-1}
κ	injection rate	kg km^{-2} h^{-1}

2.1 The Pollution Problem

To model the injection of pollution at center point $(0, 0) \in \Omega$, we set the smooth source term $f \equiv 0$ and model the point source by

$$s(x, y, t) = \kappa\, \delta_{(0,0)}(x, y) \chi_{[0,8]}(t) \tag{2.5}$$

with output rate κ in kg km^{-2} h^{-1}. Here, $\delta_{(0,0)}(x, y) := \delta(x-0)\,\delta(y-0)$ defines the Dirac delta distribution in 2-D as product of delta distributions in 1-D[1] to model the injection of pollution at

[1]The **Dirac delta distribution** (in one dimension) $\delta(x)$ for $x \in (a, b)$ is defined by the properties $\delta(x-c) = 0$ for $x \neq c$ and $\int_a^b \varphi(x)\,\delta(x-c)\,dx = \varphi(c)$ for any continuous function $\varphi(x)$ and point $c \in (a, b)$.

center point $(0,0) \in \Omega$, and χ denotes the indicator function $\chi_{[0,8]}(t) = 1$ for times $0 \leq t \leq 8$ and 0 otherwise to control the switching of the pollution source.

2.2 Test Problem with Smooth Source Term

We will also, and in fact first, use a test problem that has the form of (2.1)–(2.3), but admits a chosen, known true solution $u(x, y, t)$ in closed form. Concretely, we choose

$$u(x, y, t) = (1 - e^{-(t/\tau)^2}) \cos^2(\pi x/100) \cos^2(\pi y/100) \tag{2.6}$$

with $\tau = 8$, which is the solution of the model problem with a smooth source term by setting

$$\begin{aligned} f(x, y, t) = & (2t/\tau^2) e^{-(t/\tau)^2} \cos^2(\pi x/100) \cos^2(\pi y/100) \\ & + (1 - e^{-(t/\tau)^2}) \left((-2\pi^2/100^2) \cos(2\pi x/100) \cos^2(\pi y/100) + \right. \\ & \left. (-2\pi^2/100^2) \cos^2(\pi x/100) \cos(2\pi y/100) \right) \end{aligned} \tag{2.7}$$

and without point source by setting $s \equiv 0$.

3 Numerical Method

3.1 Spatial Discretization

We use the finite volume method (FVM) for the spatial discretization, because it conserves mass of the discrete concentration approximations and because the FVM can directly and rigorously handle a point source modeled by a Dirac delta distribution.

The problem (2.1)–(2.3) is stated on a two-dimensional domain

$$\Omega = (x_{min}, x_{max}) \times (y_{min}, y_{max}) \subset \mathbb{R}^2 \tag{3.1}$$

that is assumed to be square and symmetric about $(0,0)$ in both x- and y-direction. The symmetric property of Ω requires that $x_{min} = -x_{max}$ and $y_{min} = -y_{max}$. For the domain to be square requires furthermore that $x_{min} = y_{min}$ and $x_{max} = y_{max}$. To derive the finite volume method, we define the primal mesh points as $(x_i, y_j) \in \overline{\Omega}$ with $x_i = x_{min} + (i-1)\,h$, $i = 1, \ldots, N_0$, and $y_j = y_{min} + (j-1)\,h$, $j = 1, \ldots, N_0$, with uniform mesh spacing $h = L/(N_0 - 1)$, where $L = x_{max} - x_{min}$. We restrict N_0 to an odd integer, so that center point $(0,0)$ of the domain Ω in (3.1) is guaranteed to be a mesh point $(x_{i_c}, y_{j_c}) = (0,0)$ for some (i_c, j_c). Then we define the dual mesh on triangulation $\mathcal{T}_h = \{\Omega_{ij}\}_{(i,j)}$ of cells around each mesh point (x_i, y_j), $i, j = 1, \ldots, N_0$ of the primary mesh, such that $\Omega_{i_1 j_1} \cap \Omega_{i_2 j_2} = \emptyset$ for all $(i_1, j_1) \neq (i_2, j_2)$ and $\overline{\Omega} = \bigcup_{(i,j)} \overline{\Omega}_{ij}$. For the concrete value $N_0 = 5$, Figure 3.1 shows an example of the primary mesh with coordinates x_i, $i = 1, \ldots, N_0$, and y_j, $j = 1, \ldots, N_0$, of the primary mesh points and cells Ω_{42}, Ω_{14}, and Ω_{11} of the dual mesh as sample interior, boundary, and corner cells, respectively, whose interior cell boundaries lie halfway between

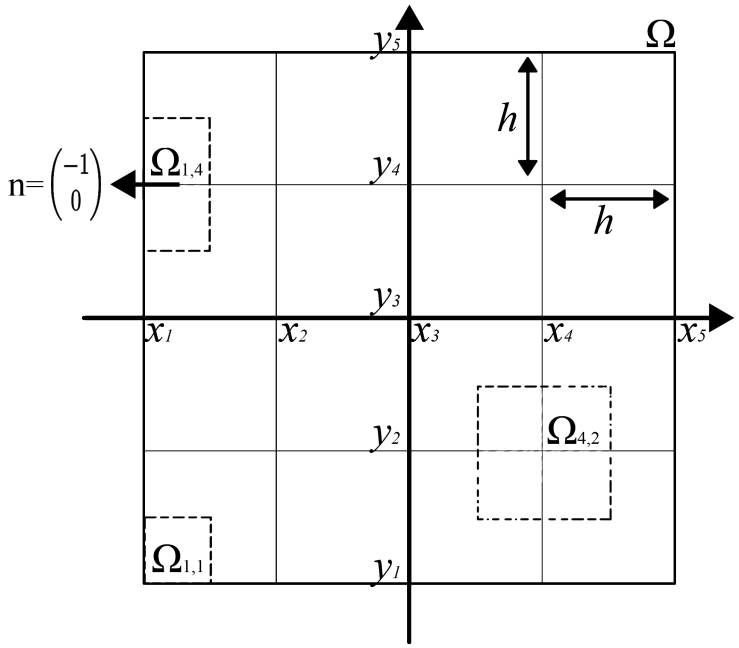

Figure 3.1: For $N_0 = 5$, primal mesh with uniform mesh spacing h in solid lines and sample cells Ω_{42}, Ω_{14}, Ω_{11} of the dual mesh in dashed lines.

primary mesh points.

For each cell $\Omega_{ij} \in \mathcal{T}_h$ of the dual mesh, we introduce now as the unknown approximation of the FVM the spatial mean values or cell averages

$$\overline{u}_{ij}(t) := \frac{1}{|\Omega_{ij}|} \iint_{\Omega_{ij}} u(x,y,t)\,dx\,dy, \quad i,j = 1,\ldots,N_0, \tag{3.2}$$

where $|\Omega_{ij}| = \iint_{\Omega_{ij}} 1\,dx\,dy$ denotes the area of the cell. The finite volume method starts by integrating the PDE (2.1) over each cell $\Omega_{ij} \in \mathcal{T}_h$ to obtain the conservative integral form

$$\iint_{\Omega_{ij}} \frac{\partial u}{\partial t}\,dx\,dy - \iint_{\Omega_{ij}} \nabla \cdot (D\,\nabla u)\;dx\,dy = \iint_{\Omega_{ij}} q\,dx\,dy. \tag{3.3}$$

We apply the divergence theorem[2] to the diffusion term in (3.3) to explicitly track the flow through the boundary $\partial\Omega_{ij}$ of each cell Ω_{ij}. Taking flux $J = D\,\nabla u$ and $W = \Omega_{ij}$, we obtain

$$\frac{d}{dt} \iint_{\Omega_{ij}} u\,dx\,dy - \int_{\partial\Omega_{ij}} \mathbf{n} \cdot (D\,\nabla u)\;dS = \iint_{\Omega_{ij}} q\,dx\,dy \tag{3.4}$$

for each cell $\Omega_{ij} \in \mathcal{T}_h$, $i,j = 1,\ldots,N_0$, of the dual mesh, where we also interchanged the order of integration and differentiation in the first integral. Dividing now by $|\Omega_{ij}|$ yields

$$\frac{d}{dt}\left(\frac{1}{|\Omega_{ij}|} \iint_{\Omega_{ij}} u(x,y,t)\,dx\,dy\right) - \frac{1}{|\Omega_{ij}|} \int_{\partial\Omega_{ij}} \mathbf{n} \cdot (D\,\nabla u)\;dS = \frac{1}{|\Omega_{ij}|} \iint_{\Omega_{ij}} q\,dx\,dy \tag{3.5}$$

and finally with (3.2) in the first term

$$\frac{d\overline{u}_{ij}(t)}{dt} - \frac{1}{|\Omega_{ij}|} \int_{\partial\Omega_{ij}} \mathbf{n} \cdot (D\,\nabla u)\;dS = \frac{1}{|\Omega_{ij}|} \iint_{\Omega_{ij}} q\,dx\,dy \tag{3.6}$$

as semi-discretization for the FVM unknowns $\overline{u}_{ij}(t)$, $i,j = 1,\ldots,N_0$. This system is not self-contained, yet, since the diffusion and source terms still need to be cast in terms of the $\overline{u}_{ij}(t)$.

3.1.1 Semi-Discretization of an Interior Cell

All interior cells of the dual mesh have the form $\Omega_{ij} = (x_i - \frac{h}{2}, x_i + \frac{h}{2}) \times (y_j - \frac{h}{2}, y_j + \frac{h}{2})$, $i,j = 2,\ldots,N_0-1$, that is, a square of size h^2 centered around each primal mesh point (x_i, y_j), with each of the four boundary segments of $\partial\Omega_{ij}$ located halfway between primal mesh points. For $N_0 = 5$, Figure 3.1 shows an example of interior cell Ω_{42} around the mesh point (x_4, y_2).

The discretization of the right-hand side integral of $q = f + s$ in (3.6) is considered in two parts, the smooth source term f which yields

$$\frac{1}{|\Omega_{ij}|} \iint_{\Omega_{ij}} f\,dx\,dy \approx \frac{1}{|\Omega_{ij}|} \int_{y_j-h/2}^{y_j+h/2} \int_{x_i-h/2}^{x_i+h/2} 1\,dx\,dy\,f(x_i,y_j,t) = f(x_i,y_j,t). \tag{3.7}$$

and the point source term $s(x,y,t) = \kappa\,\delta_{(0,0)}(x,y)\;\chi_{[0,8]}(t)$, where we obtain

$$\frac{1}{|\Omega_{ij}|} \iint_{\Omega_{ij}} s\,dx\,dy = \frac{\kappa}{|\Omega_{ij}|} \iint_{\Omega_{ij}} \delta_{(0,0)}(x,y)\;dx\,dy\;\chi_{[0,8]}(t) = \frac{\kappa}{h^2}\,\delta_{ii_c}\,\delta_{jj_c}\,\chi_{[0,8]}(t)\,, \tag{3.8}$$

[2]The **divergence theorem** states $\iint_W \nabla \cdot J\,dx\,dy = \int_{\partial W} \mathbf{n} \cdot J\,dS$ for a vector field $J : W \to \mathbb{R}^2$ and $W \subset \mathbb{R}^2$.

which is 0 if $(i,j) \neq (i_c, j_c)$ or $t > 8$ and is κ/h^2 if $(i,j) = (i_c, j_c)$ and $0 \leq t \leq 8$.[3]

The surface integral over the diffusion term in (3.6)

$$\begin{aligned}
&\frac{1}{|\Omega_{ij}|} \int_{\partial\Omega_{ij}} \mathbf{n} \cdot (D\,\nabla u)\; dS \\
&= \frac{1}{|\Omega_{ij}|} \int_{y_j-h/2}^{y_j+h/2} \mathbf{n} \cdot (D\,\nabla u)\big|_{x=x_i-\frac{h}{2}}\; dy + \frac{1}{|\Omega_{ij}|} \int_{y_j-h/2}^{y_j+h/2} \mathbf{n} \cdot (D\,\nabla u)\big|_{x=x_i+\frac{h}{2}}\; dy \\
&+ \frac{1}{|\Omega_{ij}|} \int_{x_i-h/2}^{x_i+h/2} \mathbf{n} \cdot (D\,\nabla u)\big|_{y=y_j-\frac{h}{2}}\; dx + \frac{1}{|\Omega_{ij}|} \int_{x_i-h/2}^{x_i+h/2} \mathbf{n} \cdot (D\,\nabla u)\big|_{y=y_j+\frac{h}{2}}\; dx
\end{aligned} \tag{3.9}$$

is calculated by considering each of the four linear segments of $\partial\Omega_{ij}$ individually. Consider the left-hand segment, which is located at $x = x_i - \frac{h}{2}$ for $y_j - \frac{h}{2} \leq y \leq y_j + \frac{h}{2}$, with outward unit normal vector $\mathbf{n} = (-1, 0)^T$. Then the diffusion integral over this segment is

$$S_1 = \frac{1}{|\Omega_{ij}|} \int_{y_j-h/2}^{y_j+h/2} \mathbf{n} \cdot (D\,\nabla u)\big|_{x=x_i-\frac{h}{2}}\; dy = \frac{1}{|\Omega_{ij}|} \int_{y_j-h/2}^{y_j+h/2} (-1, 0) \left(D \begin{pmatrix} u_x \\ u_y \end{pmatrix}\Big|_{x=x_i-\frac{h}{2}} \right) dy. \tag{3.10}$$

The surface integral is now approximated by a midpoint quadrature rule centered at $y = y_j$ to yield the approximation for the diffusive flux at $x = x_i - \frac{h}{2}$ on the boundary betweens cells Ω_{i-1j} and Ω_{ij}

$$S_1 \approx \frac{1}{|\Omega_{ij}|} \int_{y_j-h/2}^{y_j+h/2} 1\, dy\; (-1, 0)\; (D\,\nabla u)\big|_{(x,y)=(x_i-\frac{h}{2}, y_j)} = -\frac{1}{h^2}\, h\, D\, u_x(x_i - \tfrac{h}{2}, y_j, t). \tag{3.11}$$

The derivative $u_x(x_i - \frac{h}{2}, y_j, t)$ is approximated by the second-order accurate centered finite difference approximation to obtain

$$S_1 \approx -\frac{1}{h}\, D\, \frac{u(x_i, y_j, t) - u(x_{i-1}, y_j, t)}{h} \approx -\frac{D}{h}\, \frac{\overline{u}_{ij}(t) - \overline{u}_{i-1j}(t)}{h}, \tag{3.12}$$

where the final approximation holds, since for each interior cell holds

$$\overline{u}_{ij}(t) = \frac{1}{|\Omega_{ij}|} \iint_{\Omega_{ij}} u(x, y, t)\, dx\, dy \approx \frac{1}{|\Omega_{ij}|} \iint_{\Omega_{ij}} 1\, dx\, dy\; u(x_i, y_j, t) = u(x_i, y_j, t) \tag{3.13}$$

to second-order accuracy, provided (x_i, y_j) is the center of Ω_{ij}. The above derivation justifies how an integral approximation and a centered finite difference of the *pointwise value* $u_x\left(x_i - \frac{h}{2}, y_j, t\right)$ can be leveraged to yield a flux in terms of a finite difference of the *cell averages* $\overline{u}_{i-1j}$ and $\overline{u}_{ij}$.

Using (3.12) and with analogous derivations of the other segments S_2, S_3, S_4 of $\partial\Omega_{ij}$, we obtain in total for the diffusion term in (3.9), while also implementing a change in ordering of the segments,

$$\begin{aligned}
&\frac{1}{|\Omega_{ij}|} \int_{\partial\Omega_{ij}} \mathbf{n} \cdot (D\,\nabla u)\; dS = S_1 + S_2 + S_3 + S_4 = S_3 + S_1 + S_2 + S_4 \\
&\quad \approx \frac{D}{h}\, \frac{\left(\overline{u}_{ij-1} - \overline{u}_{ij}\right)}{h} + \frac{D}{h}\, \frac{\left(\overline{u}_{i-1j} - \overline{u}_{ij}\right)}{h} + \frac{D}{h}\, \frac{\left(\overline{u}_{i+1j} - \overline{u}_{ij}\right)}{h} + \frac{D}{h}\, \frac{\left(\overline{u}_{ij+1} - \overline{u}_{ij}\right)}{h} \\
&\quad = \frac{D}{h^2} \Big(\overline{u}_{ij-1} + \overline{u}_{i-1j} - 4\overline{u}_{ij} + \overline{u}_{i+1j} + \overline{u}_{ij+1} \Big).
\end{aligned} \tag{3.14}$$

[3]Note we also use the notation of the **Kronecker delta function** $\delta_{i\ell} = \begin{cases} 0 & \text{for } i \neq \ell, \\ 1 & \text{for } i = \ell \end{cases}$ for integers i, ℓ.

Inserting all approximations (3.7), (3.8), and (3.14) into the integral equation (3.6), we get the semi-discretization

$$\frac{d\overline{u}_{ij}(t)}{dt} + \frac{D}{h^2}\Big(-\overline{u}_{ij-1}(t) - \overline{u}_{i-1j}(t) + 4\overline{u}_{ij}(t) - \overline{u}_{i+1j}(t) - \overline{u}_{ij+1}(t)\Big) = f(x_i, y_j, t) + \frac{\kappa}{h^2}\,\delta_{ii_c}\,\delta_{jj_c}\,\chi_{[0,8]}\,(t) \tag{3.15}$$

for all interior cells Ω_{ij}, $i = 2, \ldots, N_0 - 1$, $j = 2, \ldots, N_0 - 1$.

Remark: We can define the diffusive flux functions from cell Ω_{i-1j} to Ω_{ij} as

$$H_{d,x}\left(\overline{u}_{i-1j}(t), \overline{u}_{ij}(t), x_i - \tfrac{h}{2}\right) = D\,\frac{\overline{u}_{ij}(t) - \overline{u}_{i-1j}(t)}{h} \tag{3.16}$$

and from cell Ω_{ij-1} to Ω_{ij} as

$$H_{d,y}\left(\overline{u}_{ij-1}(t), \overline{u}_{ij}(t), y_j - \tfrac{h}{2}\right) = D\,\frac{\overline{u}_{ij}(t) - \overline{u}_{ij-1}(t)}{h} \tag{3.17}$$

to write formally from (3.12) for S_1 and its analogues for S_2, S_3, and S_4 to obtain

$$\begin{aligned}\frac{1}{|\Omega_{ij}|}\int_{\partial\Omega_{ij}} \mathbf{n}\cdot(D\,\nabla u)\;dS &= S_1 + S_2 + S_3 + S_4\\ &\approx -\frac{1}{h}H_{d,x}(\overline{u}_{i-1j}, \overline{u}_{ij}, x_i - \tfrac{h}{2}) + \frac{1}{h}H_{d,x}(\overline{u}_{ij}, \overline{u}_{i+1j}, x_i + \tfrac{h}{2})\\ &\quad -\frac{1}{h}H_{d,y}(\overline{u}_{ij-1}, \overline{u}_{ij}, y_j - \tfrac{h}{2}) + \frac{1}{h}H_{d,y}(\overline{u}_{ij}, \overline{u}_{ij+1}, y_j + \tfrac{h}{2}),\end{aligned} \tag{3.18}$$

which is a more abstract presentation of (3.14). Furthermore, we can use this abstraction in (3.15)

$$\begin{aligned}\frac{d\overline{u}_{ij}(t)}{dt} &- \frac{1}{h}H_{d,x}(\overline{u}_{i-1j}, \overline{u}_{ij}, x_i - \tfrac{h}{2}) + \frac{1}{h}H_{d,x}(\overline{u}_{ij}, \overline{u}_{i+1j}, x_i + \tfrac{h}{2})\\ &- \frac{1}{h}H_{d,y}(\overline{u}_{ij-1}, \overline{u}_{ij}, y_j - \tfrac{h}{2}) + \frac{1}{h}H_{d,y}(\overline{u}_{ij}, \overline{u}_{ij+1}, y_j + \tfrac{h}{2})\\ &= f(x_i, y_j, t) + \frac{\kappa}{h^2}\,\delta_{ii_c}\,\delta_{jj_c}\,\chi_{[0,8]}\,(t)\end{aligned} \tag{3.19}$$

to describe the semi-discretization for the interior cells.

3.1.2 Semi-Discretization of a Boundary Cell

The following derivation not documented in detail, yet, but should result in

$$\frac{1}{2}\frac{d\overline{u}_{ij}(t)}{dt} + \frac{D}{h^2}\Big(-\frac{1}{2}\overline{u}_{i-1j}(t) + 2\overline{u}_{ij}(t) - \frac{1}{2}\overline{u}_{i+1j}(t) - \overline{u}_{ij+1}(t)\Big) = \frac{1}{2}\,f(x_i, y_j, t) \tag{3.20}$$

for all boundary mesh points (x_i, y_j), $i = 2, \ldots, N_0 - 1$, $j = 1$. This derivation applies to a cell at the bottom boundary of Ω. Analogous derivations are applied to the top, left, and right boundaries.

3.1.3 Semi-Discretization of a Corner Cell

The following derivation not documented in detail, yet, but should result in

$$\frac{1}{4}\frac{d\overline{u}_{ij}(t)}{dt} + \frac{D}{h^2}\Big(\overline{u}_{ij}(t) - \frac{1}{2}\overline{u}_{i+1j}(t) - \frac{1}{2}\overline{u}_{ij+1}(t)\Big) = \frac{1}{4}\, f(x_i, y_j, t) \tag{3.21}$$

for the corner mesh point (x_i, y_j), $i = 1$, $j = 1$. This derivation applies to a cell at the left bottom corner of $\partial\Omega$. Analogous derivations are applied to the right bottom, left top, and right top corners of the boundary.

3.1.4 System of All Semi-Discretizations

Define the column-vector $\mathbf{u}(t) = (\mathbf{u}_k(t)) \in \mathbb{R}^N$, $k = 1, \ldots, N$ with $N := N_0^2$, with components $\mathbf{u}_k(t) = \overline{u}_{ij}(t)$, ordered by $k = i + N_0\,(j-1)$ for $i, j = 1, \ldots, N_0$. Also analogously define the shorthand notation $\mathbf{q}(t) = (\mathbf{q}_k(t)) \in \mathbb{R}^N$ with $\mathbf{q}_k(t) = f(x_i, y_j, t) + \frac{\kappa}{h^2}\delta_{ii_c}\,\delta_{jj_c}\,\chi_{[0,8]}\,(t)$, using the same ordering of the components. Assembling all types of semi-discretizations (3.15), (3.20), (3.21) yields then in vector form

$$\hat{M}\,\frac{d\mathbf{u}(t)}{dt} + K\,\mathbf{u}(t) = \hat{M}\mathbf{q}(t), \tag{3.22}$$

with the constant matrices $\hat{M}, K \in \mathbb{R}^{N\times N}$. Here, the stiffness matrix $K = \frac{D}{h^2}(S\otimes T + T\otimes S) \in \mathbb{R}^{N\times N}$ can be computed using the sum of Kronecker products between the diagonal matrix $S \in \mathbb{R}^{N_0\times N_0}$ and the tri-diagonal matrix $T \in \mathbb{R}^{N_0\times N_0}$ given by

$$S = \begin{bmatrix} \frac{1}{2} & & & & & & \\ & 1 & & & & & \\ & & 1 & & & & \\ & & & \ddots & & & \\ & & & & 1 & & \\ & & & & & 1 & \\ & & & & & & \frac{1}{2} \end{bmatrix}, \qquad T = \begin{bmatrix} 1 & -1 & & & & & \\ -1 & 2 & -1 & & & & \\ & -1 & 2 & -1 & & & \\ & & \ddots & \ddots & \ddots & & \\ & & & -1 & 2 & -1 & \\ & & & & -1 & 2 & -1 \\ & & & & & -1 & 1 \end{bmatrix}. \tag{3.23}$$

The diagonal mass matrix $\hat{M} = S \otimes S \in \mathbb{R}^{N\times N}$ collects the pre-factors in front of the $d\overline{u}_{ij}(t)/dt$ terms in (3.15), (3.20), (3.21) and can be computed as one Kronecker product using the same S.

For instance for $N_0 = 5$, the matrix K would read concretely

$$K = \frac{D}{h^2} \left[\begin{array}{cccccccccccccccccccccccc}
1 & -\frac{1}{2} & & & & -\frac{1}{2} & & & & & & & & & & & & & & & & & & \\
-\frac{1}{2} & 2 & -\frac{1}{2} & & & & -1 & & & & & & & & & & & & & & & & & \\
 & -\frac{1}{2} & 2 & -\frac{1}{2} & & & & -1 & & & & & & & & & & & & & & & & \\
 & & -\frac{1}{2} & 2 & -\frac{1}{2} & & & & -1 & & & & & & & & & & & & & & & \\
 & & & -\frac{1}{2} & 1 & & & & & -\frac{1}{2} & & & & & & & & & & & & & & \\
-\frac{1}{2} & & & & & 2 & -1 & & & & -\frac{1}{2} & & & & & & & & & & & & & \\
 & -1 & & & & -1 & 4 & -1 & & & & -1 & & & & & & & & & & & & \\
 & & -1 & & & & -1 & 4 & -1 & & & & -1 & & & & & & & & & & & \\
 & & & -1 & & & & -1 & 4 & -1 & & & & -1 & & & & & & & & & & \\
 & & & & -\frac{1}{2} & & & & -1 & 2 & & & & & -\frac{1}{2} & & & & & & & & & \\
 & & & & & -\frac{1}{2} & & & & & 2 & -1 & & & & -\frac{1}{2} & & & & & & & & \\
 & & & & & & -1 & & & & -1 & 4 & -1 & & & & -1 & & & & & & & \\
 & & & & & & & -1 & & & & -1 & 4 & -1 & & & & -1 & & & & & & \\
 & & & & & & & & -1 & & & & -1 & 4 & -1 & & & & -1 & & & & \\
 & & & & & & & & & -\frac{1}{2} & & & & -1 & 2 & & & & & -\frac{1}{2} & & & \\
 & & & & & & & & & & -\frac{1}{2} & & & & & 1 & -\frac{1}{2} & & & & & & \\
 & & & & & & & & & & & -1 & & & & -\frac{1}{2} & 2 & -\frac{1}{2} & & & & & \\
 & & & & & & & & & & & & -1 & & & & -\frac{1}{2} & 2 & -\frac{1}{2} & & & & \\
 & & & & & & & & & & & & & -1 & & & & -\frac{1}{2} & 2 & -\frac{1}{2} & & & \\
 & & & & & & & & & & & & & & -\frac{1}{2} & & & & -\frac{1}{2} & 1 & & &
\end{array}\right] \tag{3.24}$$

and the mass matrix $\hat{M}$ concretely

$$\hat{M} = \left[\begin{array}{cccccccccccccccccccccccc}
\frac{1}{4} & & & & & & & & & & & & & & & & & & & & & & & \\
 & \frac{1}{2} & & & & & & & & & & & & & & & & & & & & & & \\
 & & \frac{1}{2} & & & & & & & & & & & & & & & & & & & & & \\
 & & & \frac{1}{2} & & & & & & & & & & & & & & & & & & & & \\
 & & & & \frac{1}{4} & & & & & & & & & & & & & & & & & & & \\
 & & & & & \frac{1}{2} & & & & & & & & & & & & & & & & & & \\
 & & & & & & 1 & & & & & & & & & & & & & & & & & \\
 & & & & & & & 1 & & & & & & & & & & & & & & & & \\
 & & & & & & & & 1 & & & & & & & & & & & & & & & \\
 & & & & & & & & & \frac{1}{2} & & & & & & & & & & & & & & \\
 & & & & & & & & & & \frac{1}{2} & & & & & & & & & & & & & \\
 & & & & & & & & & & & 1 & & & & & & & & & & & & \\
 & & & & & & & & & & & & 1 & & & & & & & & & & & \\
 & & & & & & & & & & & & & 1 & & & & & & & & & & \\
 & & & & & & & & & & & & & & \frac{1}{2} & & & & & & & & & \\
 & & & & & & & & & & & & & & & \frac{1}{2} & & & & & & & & \\
 & & & & & & & & & & & & & & & & \frac{1}{4} & & & & & & & \\
 & & & & & & & & & & & & & & & & & \frac{1}{2} & & & & & & \\
 & & & & & & & & & & & & & & & & & & \frac{1}{2} & & & & & \\
 & & & & & & & & & & & & & & & & & & & \frac{1}{2} & & & & \\
 & & & & & & & & & & & & & & & & & & & & \frac{1}{2} & & & \\
 & & & & & & & & & & & & & & & & & & & & & \frac{1}{4} & &
\end{array}\right] \tag{3.25}$$

To see in what sense the FVM conserves mass at the discrete level, recall what it means at the

continuous level: If $u(x, y, t)$ denotes the concentration of a quantity throughout domain Ω, then the total mass in the domain $\iint_\Omega u\,dx\,dy$ is conserved if the time derivative of the total is zero for all $t > 0$. We see that the PDE (2.1) without a source by $q \equiv 0$ satisfies this condition by integrating and applying the divergence theorem in

$$\frac{d}{dt}\iint_\Omega u\,dx\,dy = \iint_\Omega \frac{\partial u}{\partial t}\,dx\,dy = \iint_\Omega \nabla\cdot(D\,\nabla u)\,dx\,dy = \int_{\partial\Omega} \mathbf{n}\cdot(D\,\nabla u)\,dS = 0, \tag{3.26}$$

which is 0 by BC (2.2). The analogue condition on the discrete level starts by noting that Ω is the union of all cells Ω_{ij}, that is, $\Omega = \bigcup_{ij}\Omega_{ij}$, and thus by linearity of the integral

$$\frac{d}{dt}\iint_\Omega u\,dx\,dy = \frac{d}{dt}\sum_{(i,j)}\iint_{\Omega_{ij}} u\,dx\,dy = \sum_{(i,j)}|\Omega_{ij}|\;\frac{d}{dt}\left(\frac{1}{|\Omega_{ij}|}\iint_{\Omega_{ij}} u\,dx\,dy\right) = \sum_{(i,j)}|\Omega_{ij}|\;\frac{d\overline{u}_{ij}}{dt}. \tag{3.27}$$

Since the area of each cell $|\Omega_{ij}|$ is h^2 for interior cell, $h^2/2$ for boundary cell, and $h^2/4$ for corner cell, the terms $|\Omega_{ij}|\;\frac{d\overline{u}_{ij}}{dt}$ under the sums are, aside from common factor h^2, exactly the time derivative with pre-factor in (3.15), (3.20), (3.21), respectively. Replacing each $|\Omega_{ij}|\;\frac{d\overline{u}_{ij}}{dt}$ by the sum of the approximations from the diffusion term (noting that $q \equiv 0$) yields a large sum of approximations with common factor $-D$ (since the h^2 cancel). To see that this sum is 0, notice for instance that each interior approximation $\overline{u}_{ij}$ has factor 4 in (3.15) and cancels against the $-\overline{u}_{ij-1}$, $-\overline{u}_{i-1j}$, $-\overline{u}_{i+1j}$, $-\overline{u}_{ij+1}$ from the four neighboring cells; this remains true if the neighboring cell is a boundary cell with (3.20). In turn, boundary approximations $\overline{u}_{ij}$ have factor 2 in (3.20) and cancel against one term from the neighboring interior cell with factor 1 and one term each from the two neighboring boundary cells with factor 1/2; this remains true if one of the neighboring cells is a corner cell with (3.21). In this way, the final sum in (3.27) is 0, which shows the mass conservation on the discrete level.

3.2 Time Discretization

Since the semi-discretization of a parabolic PDE such as (2.1) is necessarily a stiff system of ODEs, we need an appropriate ODE solver, namely an ODE solver that is appropriate for stiff ODEs. There are sophisticated ODE solver choices available, such as the family of backward differentiation formulas (BDFk) and numerical differentiation formulas (NDFk) methods with automatic selection of time step size Δt and method order $1 \le k \le 5$ implemented in Matlab's `ode15s` function, whose use we describe in the later Subsection 3.2.2. But first, we describe in Subsection 3.2.1 the time discretization with the simplest stiff ODE solver, namely the implicit Euler method, in order to see how a full discretization (i.e., both space and time variables discretized) looks like; moreover, for this problem, this shows that the full discretization is actually a system of linear equations that can be particularly easily coded, if code for a stationary Poisson equation is available.

3.2.1 The Implicit Euler Method

For an ODE method with constant time step size Δt, we introduce the discrete times $t_n := n\,\Delta t$ for $n = 0, 1, \ldots, N_t$, where $\Delta t > 0$ and integer $N_t > 0$ are such that $N_t\,\Delta t = t_{fin}$ for the final time t_{fin}; we assume an initial time $t_0 = 0$ for simplicity of the notation. The implicit Euler method is based on using the backward finite difference approximation

$$\frac{d\mathbf{u}(t_{n+1})}{dt} \approx \frac{\mathbf{u}(t_{n+1}) - \mathbf{u}(t_n)}{\Delta t} \tag{3.28}$$

of the first-order time derivative evaluated at $t_{n+1} = t_n + \Delta t$. The implicit Euler method is then derived by first evaluating the semi-discretization (3.22) at time $t = t_{n+1}$ and inserting the derivative approximation (3.28) to yield

$$\hat{M}\,\frac{\mathbf{u}(t_{n+1}) - \mathbf{u}(t_n)}{\Delta t} + K\,\mathbf{u}(t_{n+1}) \approx \hat{M}\mathbf{q}(t_{n+1}) \tag{3.29}$$

and second using this as defining equation for the approximations $\mathbf{u}^{(n)} \approx \mathbf{u}(t_n)$ as

$$\hat{M}\,\frac{\mathbf{u}^{(n+1)} - \mathbf{u}^{(n)}}{\Delta t} + K\,\mathbf{u}^{(n+1)} = \hat{M}\mathbf{q}^{(n+1)} \tag{3.30}$$

where we also use the shorthand notation $\mathbf{q}^{(n)} := \mathbf{q}(t_n)$. Multiplying (3.30) by the time step Δt produces

$$\hat{M}(\mathbf{u}^{(n+1)} - \mathbf{u}^{(n)}) + \Delta t\,K\,\mathbf{u}^{(n+1)} = \Delta t\,\hat{M}\mathbf{q}^{(n+1)}.$$

At this point, this equation has unknowns and knowns on both sides. Organizing such that the unknown vector $\mathbf{u}^{(n+1)} \in \mathbb{R}^N$ appears only on the left-hand side and all terms with $\mathbf{u}^{(n)} \in \mathbb{R}^N$, which is known at time t_n, and with $\mathbf{q}^{(n+1)} \in \mathbb{R}^N$, which is computable at any time t_{n+1}, on the right-hand side yields

$$\hat{M}\mathbf{u}^{(n+1)} + \Delta t\,K\,\mathbf{u}^{(n+1)} = \hat{M}\mathbf{u}^{(n)} + \Delta t\,\hat{M}\mathbf{q}^{(n+1)}.$$

This can finally be arranged in the form of a system of linear equations that needs to be solved at every time step t_n, $n = 0, 1, \ldots, N_t - 1$, to give the outline of the algorithm:
Initialize $\mathbf{u}^{(0)} = (\mathbf{u}_k(0)) = (\overline{u}_{ij}(0)) = (u_{ini}(x_i, y_j)) \in \mathbb{R}^N$, then

$$\text{Solve } \big(\hat{M} + \Delta t\,K\big)\mathbf{u}^{(n+1)} = \hat{M}\mathbf{u}^{(n)} + \Delta t\,\hat{M}\mathbf{q}^{(n+1)} \quad \text{ for } n = 0, 1, \ldots, N_t - 1. \tag{3.31}$$

3.2.2 The Numerical Differentiation Formulas

The numerical differentiation formulas (NDFk) implemented in Matlab's `ode15s` function are a generalization of the backward differentiation formulas (BDFk), as explained in detail in the paper by Shampine and Reichelt [3]. The original BDFk with method order k are a generalization of the idea in (3.28) by using a higher-order, one-sided approximation to the derivative $du(t_{n+1})/dt$. The idea of generalizing the BDFk to the NDFk methods is to introduce a term with a free parameter that does not change the order of the local truncation error, but that allows the use of a larger time step at the cost of a moderate degradation of method stability [3]. Both NDFk and BDFk allow for a range of orders $1 \le k \le k_{max}$ with default $k_{max} = 5$. Restricting $k_{max} = 1$ and thus fixing $k = 1$ in the BDFk recovers the implicit Euler method as BDF1.

As implemented in Matlab's `ode15s` function, these methods integrate systems of differential equations of the form

$$M\,\frac{d\mathbf{u}(t)}{dt} = f^{(ode)}(t, \mathbf{u}(t)). \tag{3.32}$$

These methods require the Jacobian $J^{(ode)}(t, \mathbf{u})$ of the ODE function $f^{(ode)}(t, \mathbf{u})$ with respect to $\mathbf{u}$. This Jacobian matrix can be generated numerically by default by the solver or can be analytically supplied by the user. For our problem (3.22), we can identify in (3.32)

$$M = \hat{M}, \quad f^{(ode)}(t, \mathbf{u}) = -K\,\mathbf{u} + \hat{M}\mathbf{q}(t). \tag{3.33}$$

Differentiating we see that the Jacobian

$$J^{(ode)}(t, \mathbf{u}) = \nabla_{\mathbf{u}} f^{(ode)}(t, \mathbf{u}) = -K \tag{3.34}$$

is a constant matrix, owing to the linearity of the original PDE (2.1).

3.3 Linear Solver

We observe that the system (3.31) that has to be solved at every time step t_n is linear in the unknown vector $\mathbf{u}^{(n+1)} \in \mathbb{R}^N$. Moreover, the system matrix $A = \hat{M} + \Delta t\, K \in \mathbb{R}^{N \times N}$ is large and sparse. Note that A is symmetric by construction. Therefore, the conjugate gradient (CG) method is a suitable linear solver, which iteratively computes a solution to a linear system, starting from a chosen initial guess, such that the Euclidean norm of the relative residual of the solution is less than a selected tolerance.

3.4 Implementation in Matlab

The Matlab implementation of the method is based on the code supplied for the Poisson problem $-\Delta u = f$ in [1]. The `setupA` function in that code can be readily modified to implement the calculation of K, as it appears here, then K is multiplied by Δt and $\hat{M}$ added. We use here that the system matrix $\hat{M} + \Delta t\, K \in \mathbb{R}^{N \times N}$ is the same at all time steps. The main code is then extended to enclose the call to the CG method in a for loop on the time step index $n = 0, 1, \ldots, N_t - 1$, where the right-hand side of the system is computed each in time step as $\mathbf{b} = \hat{M}\mathbf{u}^{(n)} + \Delta t\, \hat{M}\mathbf{q}^{(n+1)}$. The solution $\mathbf{u}^{(n)}$ at the current time step t_n is used as initial guess for the CG iteration that computes $\mathbf{u}^{(n+1)}$. For reasonably small time steps and for a smooth solution in time, this initial guess is typically very good, and only few iterations per time step are required.

4 Results

4.1 Test Problem with Smooth Source Term

With the finite volume method derived in Section 3, the next task is to implement it. To do so we utilize Matlab as it has straight forward syntax and a number of useful built-in functions which we can utilize. Namely, it has the conjugate gradient (CG) linear solver and the Kroneker product, the first of which is important for solving the matrix equation derived as (3.31) and the second which we use to create the matrix A in that same expression.

To show that our method is generally applicable and accurately solves a partial differential equation of the form of (2.1)–(2.3), we utilize the test problem specified in Section 2.2. This test problem satisfies both Neumann and Dirichlet boundary conditions, making it ideal for a general examination of our method's flexibility. Note that the test problem has a number of constants in it whose value we can set. The most important of these is $\tau = 8$. This parameter controls how quickly the test problem approaches a steady state, thereby defining the time evolution we expect to see. We have already defined our primary problem statement as a mass injection such that the function u is growing from $t = 0$ to $t = 8$ hours and diffusing afterward, hence why we set our test problem now to grow in that same interval with $\tau = 8$. We choose $D = 10$, domain $\Omega = (-50, 50) \times (-50, 50)$, and initial condition $u_{ini} \equiv 0$ as in the pollution problem.

There must also be defined a few numerical parameters, once we have written our method. Namely, we must choose the size of our timesteps ($\Delta t = 10^{-1}$) such that there is minimal build up of error as we iterate over time, the tolerance of our CG method (tol $= 10^{-9}$) such that the method accurately converges to the right solution, and the computational size of our mesh (here 129×129 cells, to start with).

The output of our test problem, solved by finite volume method (FVM), is shown in Figure 4.1. Note that our choice of τ was consistent with significant growth until $t = 8$ followed by an approach to a steady state. Intuitively, this plot looks reasonable, but the value of using a test problem is that it has a known solution, (2.6). Using this easily verified function, it is straightforward in Matlab to subtract the known solution from the calculated solution everywhere and show the difference.

In Figure 4.2 the results of such a comparison are shown. Here it is important to note the scale of the result (above the vertical axis). Namely, the entirety of the structure shown is on the order of about 10^{-3}. That is to say: the error is generally small. This is good, but it is worth noting the structure as well. Namely, we see the largest error at the center, where the problem is changing rapidly, and around the edges where our boundary approximations are made deriving the FVM.

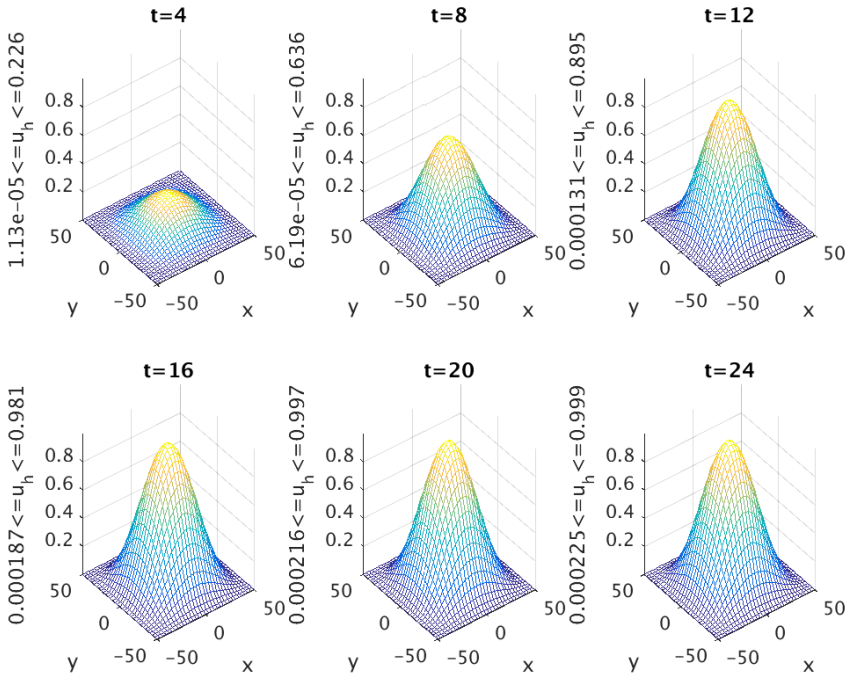

Figure 4.1: Numerical solution of the test problem at $t = 4, 8, \ldots, 24$.

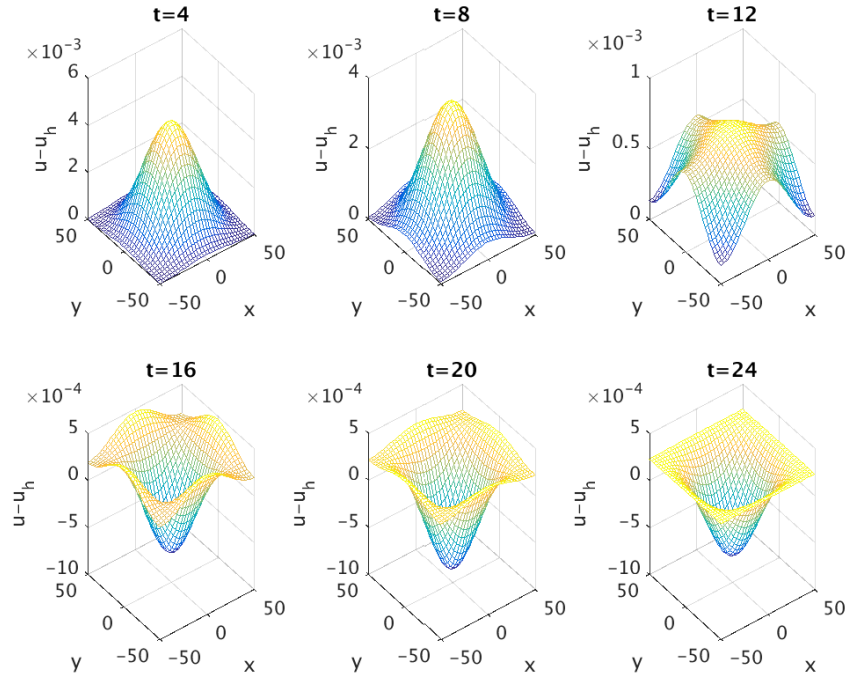

Figure 4.2: Numerical error of the test problem at $t = 4, 8, \ldots, 24$.

It is also useful to see how our backend methods are working and what the exact error peaks as. To do this we have Matlab output a number of useful data points shown in Table 4.1. Here, the column "enorminf" references the maximum of the absolute (sign independent) error shown in Figure 4.2. The values n are the number of steps through time (with t_n then being the time at that step), "it" is the number of iterations of the CG method at time step t_n, and "cumit" is the cumulative iteration count. As one might expect the values "min" and "max" are simply the minimum and maximum of the numerical solution of u.

Table 4.1: Numerical data of the test problem at $t = 4, 8, \ldots, 24$.

n	t_n	it	cumit	min(u)	max(u)	enorminf
39	4.0	7	273	1.131256e-05	2.256441e-01	4.444927e-03
79	8.0	7	561	6.185924e-05	6.356576e-01	3.537069e-03
119	12.0	8	850	1.314239e-04	8.953037e-01	7.029653e-04
159	16.0	7	1140	1.866904e-04	9.810095e-01	6.748332e-04
199	20.0	5	1420	2.164897e-04	9.972224e-01	8.470974e-04
239	24.0	5	1619	2.250605e-04	9.991729e-01	7.037356e-04

4.2 Pollution Problem on Original Numerical Domain

Confident now in the fact that our method can solve a PDE of the form of (2.1)–(2.3), it is time to return to the pollution problem defined in Section 2.1. There, our source function f is the point source (2.5) with amount κ material injected per hour into the domain at the center point $(0, 0)$. Recall that the FVM was pointedly chosen because of its ability to handle such a source and here we will begin exploring the results of that choice. To begin, we choose the value $\kappa = 10$ kg km^{-2} h^{-1} and run the simulation on $\Omega = (-50, 50) \times (-50, 50)$ in units of km. The snapshots of the concentration across the domain at six times are shown in Figure 4.3.

When examining the result, we first look to see that it makes intuitive sense. Namely, the point source is present from $t = 0$ to $t = 8$ h, and afterward the mass it has injected begins to spread out. Figure 4.3 shows this behavior, but to be truly sure of the result we need to examine quantitatively the numerical output.

Again we use direct Matlab output and put the results in Table 4.2. This table is similar to the output from the test problem shown in Table 4.1. Note though that we do not have a true solution to compare to and therefore no error. So instead we check the mass conservation of the system by integrating over the entire calculation space. The result is that the mass increases by κ for each hour to reach 80 kg by $t = 8$ h, and then that value is constant afterward, corresponding to when the source is turned off and the mass is just spreading out. This verifies not only that the FVM can handle point sources, but that it is mass conservative as well, making it ideal for physical modelling.

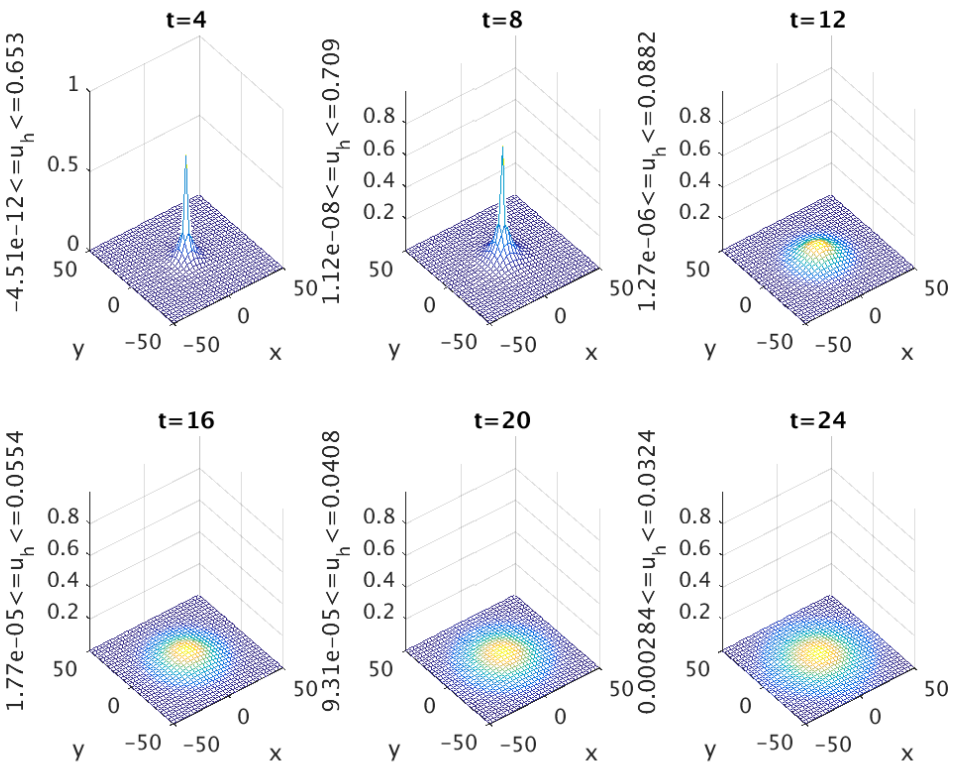

Figure 4.3: Numerical solution of pollution problem with $\kappa = 10$ kg km^{-2} h^{-1} at $t = 4, 8, \ldots, 24$ using 129×129 mesh, and on the numerical domain $(-50, 50) \times (-50, 50)$ km^2.

Table 4.2: Numerical data of pollution problem with $\kappa = 10$ kg km^{-2} h^{-1} at $t = 4, 8, \ldots, 24$ using 129×129 mesh, and on the numerical domain $(-50, 50) \times (-50, 50)$ km^2.

n	t_n	it	cumit	min(u)	max(u)	mass
39	4.0	12	755	-4.510617e-12	6.534136e-01	4.000000e+01
79	8.0	11	1182	1.117690e-08	7.091521e-01	8.000000e+01
119	12.0	11	1926	1.272000e-06	8.819662e-02	8.000000e+01
159	16.0	10	2357	1.767457e-05	5.544710e-02	8.000000e+01
199	20.0	10	2753	9.312632e-05	4.080380e-02	8.000000e+01
239	24.0	8	3135	2.842079e-04	3.236277e-02	8.000000e+01

5 Conclusions

In conclusion, we show that the finite volume method (FVM) is an ideal numerical scheme for computing the solutions of mass conservative partial differential equations. The method also performs well in handling the usage of delta distributions, and both these properties make it a good method for the handling of physical situations like those of the pollution problem defined in Section 2. While the application problem considered here was only a reaction-diffusion equation, the FVM allows the addition of an advection term without losing any of the advantages.

Acknowledgments

The hardware in the UMBC High Performance Computing Facility (HPCF) is supported by the U.S. National Science Foundation through the MRI program (grant nos. CNS–0821258, CNS–1228778, and OAC–1726023) and the SCREMS program (grant no. DMS–0821311), with additional substantial support from the University of Maryland, Baltimore County (UMBC). See hpcf.umbc.edu for more information on HPCF and the projects using its resources. Co-author Carlos Barajas was supported as HPCF RA.

References

[1] Sai K. Popuri and Matthias K. Gobbert. A comparative evaluation of Matlab, Octave, R, and Julia on Maya. Technical Report HPCF–2017–3, UMBC High Performance Computing Facility, University of Maryland, Baltimore County, 2017.

[2] Thomas I. Seidman, Matthias K. Gobbert, David W. Trott, and Martin Kružík. Finite element approximation for time-dependent diffusion with measure-valued source. *Numer. Math.*, vol. 122, no. 4, pp. 709–723, 2012.

[3] Lawrence F. Shampine and Mark W. Reichelt. The MATLAB ODE suite. *SIAM J. Sci. Comput.*, vol. 18, no. 1, pp. 1–22, 1997.

[4] Noah Sienkiewicz, Arjun Pandya, Tim Brown, Carlos Barajas, and Matthias K. Gobbert. Numerical methods for parallel simulation of diffusive pollutant transport from a point source. Technical Report HPCF–2018–11, UMBC High Performance Computing Facility, University of Maryland, Baltimore County, 2018.