Journal of Applied and Numerical Analysis
2025, Vol. 3, pp. 35–52
doi: 10.30970/ana.2025.3.35

Directional diffusion splitting method for advection-diffusion-reaction model

R. Drebotiy[1], H. Shynkarenko[1]

[1]*Ivan Franko National University of Lviv,*
1, Universytets'ka str., 79000, Lviv, Ukraine

Анотація. У контексті дослідження потенційного застосування квантових комп'ютерів для виконання процедур методу скінченних елементів запропоновано специфічний підхід до розв'язування двовимірної початково-крайової задачі адвекції-дифузії-реакції, що полягає у перетворенні її на набір відповідних одновимірних задач. Цей метод використовує спеціалізоване розщеплення оператора, що призводить до рекурентної дробово-крокової схеми інтегрування за часом. Хоча ця схема пропонує значний потенціал розпаралелювання, основна мета полягає у відображенні вихідної еліптичної двовимірної задачі, що виникає на кожному часовому кроці, в набір менших одновимірних задач такого ж типу. Це перетворення у поєднанні з нещодавно запропонованими класично-квантовими схемами стабілізації дозволяє нам передавати системи лінійних рівнянь меншої розмірності квантовому комп'ютеру, що робить їх придатними для квантових комп'ютерів із меншою кількістю кубітів. Запропонований у статті алгоритм поєднано із методом скінченних елементів для розв'язання отриманих одновимірних задач, хоча ці задачі також можуть бути розв'язані за допомогою інших методів дискретизації, таких як метод скінченних об'ємів або скінченних різниць.

Abstract. In the context of investigating the potential application of quantum computers to the finite element method, we propose a specific approach for solving the two-dimensional parabolic advection-diffusion-reaction boundary value problem by transforming it into a set of corresponding one-dimensional problems. This method employs specialized operator splitting, resulting in a recurrent fractional-step scheme for time integration. While this scheme offers significant parallelization potential, the main goal is to map the original large elliptic 2D problem arising at each time step into a set of smaller 1D problems of the same kind. This transformation, combined with recently proposed quantum-assisted stabilization schemes, allows us to pass smaller linear systems to a quantum linear solver, making them suitable for near-term quantum computers. We combine the proposed algorithm with the finite element method to solve the resulting one-dimensional problems, although these problems can also be addressed using other discretization techniques, such as the finite volume or finite difference methods.

1 Introduction

In this article, we consider a two-dimensional parabolic advection-diffusion-reaction (ADR) problem. Several well-known general methods can be applied to obtain numerical solutions of such problems. In the spatial dimensions, we have a boundary value problem (BVP) for an elliptic-type equation, which can be successfully discretized using finite difference, finite volume or finite element

Key words: advection-diffusion-reaction model, finite element method, Cauchy problem, parabolic equation, elliptic equation, boundary value problem, recurrent integration scheme, variational problem

methods (FEM). In the time dimension, we have a first-order differential equation. Typically, these problems are discretized separately in time and space. Depending on the order in which we perform the discretization, we can obtain a Cauchy problem for a system of first-order equations or a sequence of elliptic problems for each time "slice". In the spatial dimensions we can use finite elements or finite differences for discretization. In the time dimension common discretization methods for first-order initial value problems are employed, such as the Euler, Runge-Kutta or Adams methods.

Our initial interest was in solving singularly perturbed problems. This interest arises because such problems require special treatment due to the presence of boundary or inner layers with large gradients in their solution structure. A common approach is to use adaptivity and/or stabilization to achieve the required accuracy while minimizing the computational resources needed.

For singularly perturbed problems, in our recent works [6–8], we proposed stabilized and adaptive-stabilized finite element schemes that leverage the quantum Harrow-Hassidim-Lloyd (HHL) algorithm [10] (or any other quantum linear solver) for solving intermediate systems of linear equations arising within these schemes. However, currently available quantum computers have a very limited number of qubits. Moreover, error-tolerant schemes are needed, which use a certain number of qubits to provide error correction during the computation process.

Given these constraints, a natural question arises when considering the applicability of quantum linear solvers in the near term: is it possible to reduce the size of the linear systems passed to the quantum computer? Moreover, *we need to have those systems obtained from the boundary value problems of the same kind* (to be able to use the mentioned stabilization schemes). Thus, a more specialized question arises: can we decompose the original 2D problem into a set of smaller 1D problems? A positive answer to the last question is already known. Is there a chance to decompose the problem in such a way that the percentage of obtained 1D problems which are singularly perturbed is as small as possible?

The goal of this paper is to build a method that decomposes large problems into smaller ones, which can then be combined with stabilization schemes for the treatment of singularly perturbed problems and leverage a quantum computer *with more limited bandwidth*. An important property that we aim to achieve at the same time is minimizing the number of those 1D problems (out of the total number of obtained 1D problems) that are affected by dominant advection. As a side effect, the smaller linear systems can be solved independently.

The paper is structured as follows: we first define the model ADR problem, then describe the new algorithm and finally present a numerical experiment along with some further discussions.

2 Advection-diffusion-reaction problem

Let us consider the following problem for a parabolic advection-diffusion-reaction equation

$$\begin{cases} \text{find function } \ u = u(x,t) : \bar{\Omega} \times [0,T] \to \mathbb{R} \ \text{ such that:} \\ u'_t - \mu \Delta_x u + \vec{\beta} \cdot \nabla_x u + \sigma u = f \quad \text{in } \ \Omega \times (0,T], \\ u(x,t) = 0, \quad (x,t) \in \Gamma \times [0,T], \\ u(x,0) = u_0(x), \quad x \in \bar{\Omega}. \end{cases} \tag{2.1}$$

Here Ω is a bounded domain with a Lipschitz boundary Γ, $\mu = \text{const} > 0$ and $\sigma = \text{const} \geq 0$ are the coefficients of diffusion and reaction respectively. The function $f = f(x)$ and the vector $\vec{\beta} = (\beta_1(x), \beta_2(x))$ represent the sources and the advection flow velocity respectively. We assume that the flow is incompressible, i.e., $\nabla \cdot \vec{\beta} = 0$ in Ω. Additionally, we consider the case where the vector field $\vec{\beta}$ does not have closed integral curves that lie completely in $\bar{\Omega}$.

It is important to note that we consider the case where the problem data is independent of the time variable.

We assume that the problem data is such that the problem satisfies the conditions of the Lax-Milgram lemma and thus it has a unique weak solution.

In the formulation (2.1) the subscript x is used for the corresponding operators in the spatial dimensions. We also assume the time interval $[0, T]$ is finite for clarity.

Note that for $T = +\infty$ it can be shown that the concentration u stabilizes over time, i.e., we have dynamic equilibrium: $u(x,t) \to \tilde{u}(x)$ and $u'_t \to 0$ as $t \to +\infty$. Therefore we have the degeneration of the problem (2.1) to a stationary problem

$$\begin{cases} \text{find function } \ \tilde{u} : \bar{\Omega} \to \mathbb{R} \ \text{ such that:} \\ -\mu\Delta\tilde{u} + \vec{\beta} \cdot \nabla\tilde{u} + \sigma\tilde{u} = f \quad \text{in } \ \Omega \subset \mathbb{R}^2, \\ \tilde{u} = 0 \quad \text{on } \ \Gamma = \partial\Omega. \end{cases} \tag{2.2}$$

In this article we propose a method for non-stationary problem (2.1).

3 Motivation from the first-order hyperbolic problems

Let us define a part of the domain boundary as

$$\Gamma_0 = \{x \in \Gamma | \vec{n}(x) \cdot \vec{\beta}(x) < 0\},$$

where $\vec{n} : \partial\Omega \to \mathbb{R}^2$ is a unit vector of outward normal to the boundary of domain Ω.

Consider the following hyperbolic problem

$$\begin{cases} \text{find function } \ \tilde{u} \in C^1(\bar{\Omega}) \ \text{ such that:} \\ \vec{\beta}(x) \cdot \nabla\tilde{u} + \sigma\tilde{u} = f(x), \quad x \in \Omega, \\ \tilde{u}|_{\Gamma_0} = 0. \end{cases} \tag{3.1}$$

Let us define the function $[0, mes\Gamma_0] \ni \xi \mapsto \rho(\xi) = (\rho_1(\xi), \rho_2(\xi)) \in \bar{\Gamma}_0$ which maps the parameter ξ bijectively onto the $\bar{\Gamma}_0$.

For each value of ξ we can construct the integral curve of the vector field $\vec{\beta}$: $x(\tau, \xi) = (x_1(\tau, \xi), x_2(\tau, \xi)) \in \Omega$ as a solution of the following Cauchy problem

$$\begin{cases} \dfrac{\partial x(\tau,\xi)}{\partial\tau} = \vec{\beta}(x(\tau,\xi)), \\ x(0,\xi) = \rho(\xi). \end{cases} \tag{3.2}$$

Let us define the function $z(\tau, \xi) = \tilde{u}(x(\tau, \xi))$. Taking into account (3.2) and using the chain rule, we can derive the following

$$\frac{\partial z(\tau,\xi)}{\partial\tau} = \vec{\beta}(x(\tau,\xi)) \cdot \nabla_x\tilde{u}(x(\tau,\xi)). \tag{3.3}$$

Combining (3.2) and (3.3), we can rewrite the problem (3.1) as

$$\begin{cases} \dfrac{\partial x(\tau,\xi)}{\partial\tau} = \vec{\beta}(x(\tau,\xi)), \\ \dfrac{\partial z(\tau,\xi)}{\partial\tau} + \sigma z(\tau,\xi) = f(x(\tau,\xi)), \\ x(0,\xi) = \rho(\xi), \\ z(0,\xi) = 0. \end{cases} \tag{3.4}$$

We decomposed the original problem to the *parametrized set of 1D equations of the same kind*, defined on certain curves (Fig. 3.1) by using the well-known *method of characteristics*.

The complete discretization of (3.1) is achieved by discretizing the set Γ_0 and then discretizing the resulting 1D Cauchy problems (3.4). Such an algorithm can be parallelized to a certain extent.

Can we apply the same approach to the equation with diffusion present? The answer is partially positive in the sense that we cannot use the same approach directly in the same way, but we can leverage it in combination with the time integration scheme.

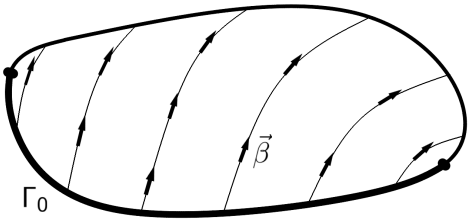

Fig. 3.1: Flow diagram for problem (3.1).

4 Diffusion splitting

Consider the diffusion term of the problem (2.1). Let us represent all vectors as columns by default. Recall, that the directional derivative of the function u in the spatial direction $\vec{e} = (e_1, e_2)^T$, $\|\vec{e}\| = 1$ can be computed as $u'_{\vec{e}} = \vec{e} \cdot \nabla_x u$. By applying this formula two times, it is easy to see, that the second derivative in the same direction can be computed by the formula $u''_{\vec{e}\vec{e}} = \nabla_x \cdot (\vec{e}\ \vec{e}^T \nabla_x u)$.

Let us consider the normalized advection vector field

$$\vec{b}(x) = (b_1(x), b_2(x)) = \vec{\beta}/\|\vec{\beta}\|. \tag{4.1}$$

Consider also the orthogonal vector field

$$\vec{\gamma}(x) = (\beta_2(x), -\beta_1(x)) \tag{4.2}$$

and corresponding normalized field

$$\vec{p} = \vec{\gamma}/\|\vec{\gamma}\|. \tag{4.3}$$

It is obvious, that the identity matrix I can be decomposed as a sum of two orthogonal projections

$$I = \vec{b}\ \vec{b}^T + \vec{p}\ \vec{p}^T.$$

Now we can transform the diffusion term of the ADR equation

$$\mu \Delta_x u = \mu \nabla_x \cdot (I \nabla_x u) = \mu \nabla_x \cdot (\vec{b}\ \vec{b}^T \nabla_x u) + \mu \nabla_x \cdot (\vec{p}\ \vec{p}^T \nabla_x u). \tag{4.4}$$

Note, that each of the obtained terms represents the second directional derivative of u in the appropriate direction.

In the next sections we will combine such splitting with the time integration scheme.

5 Variational formulation and semi-discretization in time

As a basis of the full discretization of the considered problems we use the finite element method and thus we need to reformulate problem (2.1) in the form of a variational equation.

Let us start from the boundary value problem (2.2). It admits the following variational formulation

$$\begin{cases} \text{find} \quad u \in V = H_0^1(\Omega) \quad \text{such that :} \\ a(u, v) = \langle l, v \rangle \quad \forall v \in V, \end{cases} \tag{5.1}$$

where

$$\begin{cases} a(u, v) = \int\limits_\Omega (\mu \nabla u \cdot \nabla v + \vec{\beta} v \cdot \nabla u + \sigma u v) dx \quad \forall u, v \in V, \\ \langle l, v \rangle = \int\limits_\Omega f v dx \quad \forall v \in V. \end{cases} \tag{5.2}$$

Under the assumption that the problem satisfies the conditions of the Lax-Milgram lemma, (5.1) has a unique solution.

Let us define the time step Δt and the parameter $\theta \in (0,1)$. Consider the standard Lebesgue scalar product $(w,q) = \int_\Omega wqdx$. Using components (5.2) and the standard technique from [18], we can formulate the following one-step recurrent scheme for the semi-discretization in time of the non-stationary problem (2.1)

$$\begin{cases} (\dot{u}_{j+\frac{1}{2}}, v) + \theta \Delta t a(\dot{u}_{j+\frac{1}{2}}, v) = \langle l, v\rangle - a(u_j, v), \\ u_{j+1} = u_j + \Delta t \dot{u}_{j+\frac{1}{2}}, \quad j = 0, 1, ..., \end{cases} \tag{5.3}$$

where $u_j(x)$ is an approximation to the function $u(x, t_j)$, $t_j = j\Delta t$ and by $\dot{u}_{j+\frac{1}{2}}(x)$ we denote the constant in time approximation with finite difference to the derivative u'_t on the segment $[t_j, t_{j+1}]$ (which actually is defined by the second equation in (5.3) as $(u_{j+1} - u_j)/\Delta t$).

Here and below we will preserve this "dot" notation for the approximations to time derivatives used in our recurrent integration schemes. Keeping the same letter with a dot will make clear to which function that derivative approximation "belongs." To avoid confusion with actual derivatives, we use "prime" notation for them (as in formula (2.1)).

6 Semi-discretization splitting

Let us substitute the formula (4.4) into the bilinear form $a(u,v)$ defined in (5.2). We can then rewrite it in the following way

$$a(u,v) = s(u,v) + m(u,v),$$

where

$$\begin{cases} s(u,v) = \int\limits_\Omega (\mu(\vec{b}^T \nabla_x u)(\vec{b}^T \nabla_x v) + v\|\vec{\beta}\|\vec{b}^T \nabla_x u + \sigma uv)dx, \\ m(u,v) = \int\limits_\Omega \mu(\vec{p}^T \nabla_x u)(\vec{p}^T \nabla_x v)dx. \end{cases} \tag{6.1}$$

We propose to use instead of the recurrent scheme (5.3) the following two-step scheme

$$\begin{cases} (\dot{u}_{j+\frac{1}{4}}, v) + \theta \Delta t s(\dot{u}_{j+\frac{1}{4}}, v) = \langle l, v\rangle - s(u_j, v), \\ u_{j+\frac{1}{2}} = u_j + \Delta t \dot{u}_{j+\frac{1}{4}}, \\ (\dot{u}_{j+\frac{3}{4}}, v) + \theta \Delta t m(\dot{u}_{j+\frac{3}{4}}, v) = -m(u_{j+\frac{1}{2}}, v), \\ u_{j+1} = u_{j+\frac{1}{2}} + \Delta t \dot{u}_{j+\frac{3}{4}}, \quad \forall v \in V, \quad j = 0, 1, ... \end{cases} \tag{6.2}$$

The idea behind the introduced splitting is to decouple the diffusion component, which is orthogonal to the direction of the advection flow. Visually, we can demonstrate this scheme with Fig. 6.1, which shows the modeling of the movement of an ink drop along the current of some liquid.

In the initial integration scheme (5.3), we applied both processes, diffusion and advection, simultaneously on each time step. From the topmost part of the figure we see that the drop is moved along the advection vector and it also grows in diameter due to diffusion.

The splitting scheme (6.2) decomposes this process into two substeps. In the first substep we model the advection and the diffusion part along the advection direction. In the second step, we add the orthogonal part of the diffusion to the process. This is shown in the lower part of Fig. 6.1.

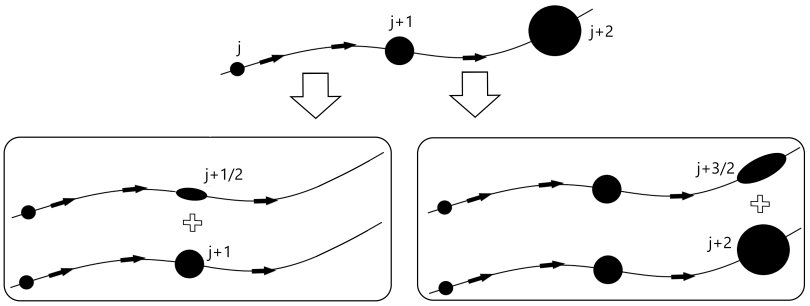

Fig. 6.1: Example: movement + diffusion of the ink drop and the diffusion splitting.

We will show in the next section that such splitting leads to an effective way of solving the obtained variational problems by decomposing them into sets of smaller 1D ADR problems, which, in the case of singularly perturbed problems, will allow us to use the mentioned quantum-assisted stabilization schemes for those 1D problems. We may note that, as a side effect, we can solve the obtained 1D problems in parallel, since they are independent of each other.

The general idea of *operator splitting*, which we used, is well-known in the literature [1–3, 9, 12, 13, 19–21]. Different names are used for such or similar methods: operator splitting, time splitting, split-step methods, dimensional splitting, alternating direction methods and fractional-step methods. A notable example is *Chorin's projection method* [3] for the Navier-Stokes equations and its variations. In this method the computations of the velocity and pressure fields are decoupled. The proposed splitting approach is fundamentally similar to the general splitting approach based on the application of the BakerCampbellHausdorff formula [14], which allows switching from one time step into two steps with individual terms of the original operator (with asymptotically quadratic error in time step length). In general, such methods can be considered a special case of fractional-step methods, where a single time step in the integration scheme is represented as a composition of several substeps (not necessarily parts of the original operator). This formula is widely used in the modeling of complex quantum systems to split a system with Hamiltonian $H = \sum_i H_i$ into smaller systems with the Hamiltonians H_i [13, 14].

The term "fractional-step method" can be considered broader than "splitting method". The latter usually refers to splitting the operator into additive parts and does not specify how these partial problems should be solved. In contrast, the "fractional-step method" takes into account various ways of representing a single integration step in time by a sequence of substeps. The term also includes schemes for the lower-dimensional problems obtained from operator splitting. A nice review of such methods can be found in [12, 13].

Within this terminology, we may consider our method as a "splitting method" since, in its origin, it does not specify which 1D integrators should be used.

The proposed method itself does not provide any inherent advantage in solving singularly perturbed problems. It can be further combined with other schemes to handle singular perturbations, but at the level of 1D problems, rather than the original 2D problem.

It is important to note that there are many different linear system solvers available today that extensively use parallelization, vector computations, GPU etc. This raises the following question: what is the advantage of splitting schemes for PDEs if they merely provide parallelization, while there are existing methods that already do the same? The key fact to consider is that the potential parallelization (which may be less efficient when compared to other methods) is not the main advantage. The main advantage is that splitting schemes decompose large problems of a certain kind (such as a PDE describing a specific process) into smaller problems on reduced dimensions, which can still be treated as problems describing other processes, i.e., prescribing the nature of the problems. These smaller problems can still be continuous and have a meaningful interpretation. Only this property allows us to propagate the developed quantum-assisted stabilization [6–8] from

the 2D problem to the corresponding smaller 1D problems obtained from the procedure, thereby reducing the required bandwidth of a quantum register. This would not be possible if we were to consider just some algebraic linear solver, even with parallelization.

If we wish to solve the stationary problem with dominant advection using such a combination of schemes, we should consider the corresponding non-stationary counterpart with $u_0 \in V$ calculated as the solution of the following equation

$$s(u_0, v) = \langle l, v \rangle \quad \forall v \in V.$$

In this way, we obtain "from start" a better approximation to the original equation, since the dynamics of the process in this case is primarily governed by the advection flow.

While the proposed method can also be applied iteratively via the time integration scheme to obtain an approximation for the elliptic problem (2.2), it can be shown that the time step used for such integration must be sufficiently small [4]. This fact makes the method inefficient for stationary problems, as a small time step leads to a large number of required iterations. Therefore, we consider the proposed method only for parabolic problems. To extend its applicability to elliptic problems, further research is required to explore the possibility of accelerating the convergence of this iterative process.

At the end of this section, we should also mention that there is ongoing work on preparing a proof of convergence of the method for parabolic problems. In the preprint [4] we have already outlined a sketch of a possible proof based on the application of the Lax equivalence theorem [11].

7 One-dimensional reduction of semi-discretization substeps

We denote by $\vec{n}(x)$ the vector of the outward unit normal to the boundary $\partial\Omega$ at the point x. Let us define the sets

$$\begin{aligned} \Gamma_{\vec{\beta}} &= \{x \in \partial\Omega | \vec{n}(x) \cdot \vec{\beta}(x) < 0\}, \\ \Gamma_{\vec{\gamma}} &= \{x \in \partial\Omega | \vec{n}(x) \cdot \vec{\gamma}(x) < 0\}. \end{aligned} \tag{7.1}$$

We suppose that each one of those sets is connected.

Consider the functions $x = x(\tau, \xi)$ and $y = y(\nu, \eta)$, $x, y \in \Omega, \tau, \nu \geq 0, \xi, \eta \in [0, 1]$, such that $x(0, \xi) = \rho(\xi) \in \Gamma_{\vec{\beta}}$ and $y(0, \eta) = \kappa(\eta) \in \Gamma_{\vec{\gamma}}$ are parametrizations of the curves $\Gamma_{\vec{\beta}}$ and $\Gamma_{\vec{\gamma}}$ correspondingly. Let us define those functions as the solutions to the following Cauchy problems

$$\begin{cases} x'_\tau = \vec{\beta}(x), \\ x(0, \xi) = \rho(\xi) \end{cases} \tag{7.2}$$

and

$$\begin{cases} y'_\nu = \vec{\gamma}(y), \\ y(0, \eta) = \kappa(\eta). \end{cases} \tag{7.3}$$

Described problems define the families of integral curves of the vector fields $\vec{\beta}$ and $\vec{\gamma}$ covering the domain Ω as depicted in the Fig. 7.1

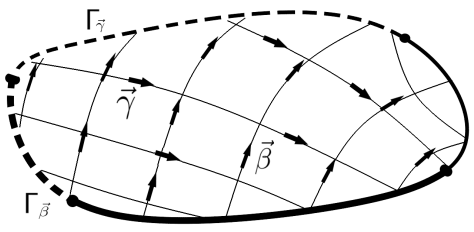

Fig. 7.1: Two sets of curves.

Consider now some finite sets of those curves: $B = \{x(\tau, \xi_i)\}_{i=1}^n$ and $G = \{y(\nu, \eta_i)\}_{i=1}^m$.

Now we can restrict the substeps of (6.2) to those sets of curves.

Let us rewrite the forms from (6.1) using the notion of directional derivative from the section 4. We have

$$\begin{cases} s(u,v) = \int\limits_\Omega (\mu u'_{\vec{b}} v'_{\vec{b}} + \|\vec{\beta}\| u'_{\vec{b}} v + \sigma uv) dx, \\ m(u,v) = \int\limits_\Omega \mu u'_{\vec{p}} v'_{\vec{p}} dx. \end{cases}$$

Let us recall the first equation from (6.2)

$$(\dot{u}_{j+\frac{1}{4}}, v) + \theta \Delta t s(\dot{u}_{j+\frac{1}{4}}, v) = \langle l, v\rangle - s(u_j, v). \tag{7.4}$$

Let us suppose that the problem data in (2.1) (and therefore in (7.4)) is sufficiently smooth, so we can assume the solution of the equation above is classical. Also in that case it will be sufficient to consider only smooth test functions v.

For the considered setting, let us apply integration by parts to rewrite the bilinear form s as follows

$$s(u,v) = (Su, v),$$

where

$$Su = -\mu u''_{\vec{b}\vec{b}} + \|\vec{\beta}\| u'_{\vec{b}} + \sigma u. \tag{7.5}$$

Now we can rewrite (7.4) in the following way

$$(\dot{u}_{j+\frac{1}{4}} + \theta \Delta t S \dot{u}_{j+\frac{1}{4}} + Su_j - f, v) = 0. \tag{7.6}$$

Alternatively, we may write

$$(R_j, v) = 0, \tag{7.7}$$

where $R_j = \dot{u}_{j+\frac{1}{4}} + \theta \Delta t S \dot{u}_{j+\frac{1}{4}} + Su_j - f$.

Let us construct a curvilinear grid using the integral curves of the normalized vector fields $\vec{b}$ and $\vec{p}$. To do this, we consider the same Cauchy problems as in (7.2) and (7.3), but substitute $\vec{b}$ for $\vec{\beta}$ and $\vec{p}$ for $\vec{\gamma}$. That is, we consider the following equations

$$x'_\tau = \vec{b}(x) \tag{7.8}$$

and

$$y'_\nu = \vec{p}(y).$$

It is important to note that with this change of parametrization, the parameters τ and ν become the *natural* parameters of the corresponding curves. That is, they represent the arc lengths of the curves measured from their respective starting points.

Now, consider a curve $L_i \in B$, but parametrized with the new parameter τ. In a small neighborhood of a point on this curve, let us construct a curvilinear domain ω, whose sides are aligned with the grid curves. In this neighborhood, we can define a curvilinear orthogonal coordinate system via the parameters (τ, ν), mapping the domain onto the rectangle $E = \{(\tau, \nu) | (\tau, \nu) \in [\tau_1, \tau_2] \times [\nu_1, \nu_2]\}$. This construction is illustrated in Fig. 7.2.

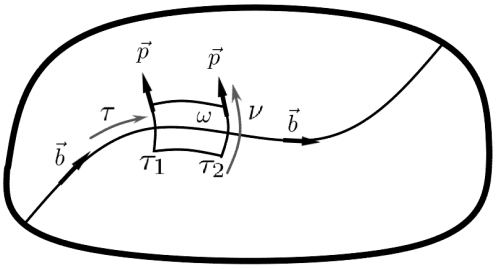

Fig. 7.2: Curvilinear grid neighbourhood.

Let us consider the scalar product (7.7) but only on the ω, i.e.

$$\int\limits_{\omega} R_j v dx = \int\limits_{\omega} R_j(x_1, x_2) v(x_1, x_2) dx_1 dx_2. \tag{7.9}$$

Let us denote mapping between curvilinear and initial coordinate systems (i.e. map from E to ω) as

$$\begin{cases} x_1 = x_1(\tau, \nu), \\ x_2 = x_2(\tau, \nu). \end{cases} \tag{7.10}$$

We suppose that $\vec{\beta}, \vec{\gamma}$ are smooth enough, so induced mapping (7.10) is differentiable.

Let us change the variables in the integral (7.9)

$$\int\limits_{\omega} R_j(x_1, x_2) v(x_1, x_2) dx_1 dx_2 = \int\limits_{E} \tilde{R}_j \tilde{v} |J| d\tau d\nu, \tag{7.11}$$

where $\tilde{R}_j(\tau, \nu) = R_j(x_1(\tau, \nu), x_2(\tau, \nu))$, $\tilde{v}(\tau, \nu) = v(x_1(\tau, \nu), x_2(\tau, \nu))$ and Jacobian is defined as

$$J = \begin{vmatrix} x'_{1,\tau} & x'_{1,\nu} \\ x'_{2,\tau} & x'_{2,\nu} \end{vmatrix}.$$

Taking into account relations (4.1), (4.2) and (4.3), we can see that

$$|J| = |\begin{vmatrix} x'_{1,\tau} & x'_{1,\nu} \\ x'_{2,\tau} & x'_{2,\nu} \end{vmatrix}| = |\begin{vmatrix} b_1 & p_1 \\ b_2 & p_2 \end{vmatrix}| = |\begin{vmatrix} b_1 & b_2 \\ b_2 & -b_1 \end{vmatrix}| = \|\vec{b}\|^2 = 1.$$

Thus we can rewrite the right part of the (7.11) as an iterated integral

$$\int\limits_{\omega} R_j v dx = \int\limits_{\tau_1}^{\tau_2} d\tau \int\limits_{\nu_1}^{\nu_2} \tilde{R}_j \tilde{v} d\nu. \tag{7.12}$$

Denote now by ν_L such ν value, that $L_i \cap \omega = \{(x_1(\tau, \nu_L), x_2(\tau, \nu_L)) | \tau \in [\tau_1, \tau_2]\}$.

Consider now the sequence of the infinitely differentiable functions $\{d_k(\nu)\}$ with compact support and which is weakly convergent to the Dirac δ-function $\delta(\nu)$ (there are well-known examples of such sequences). Suppose that supp $d_k(\nu - \nu_L) \subset [\nu_1, \nu_2]$.

Taking in (7.12) $\tilde{v} = \tilde{v}_k = \tilde{w}(\tau) d_k(\nu - \nu_L)$ (note, that we have corresponding bijective mapping from $\tilde{v}_k$ to some smooth $v = v_k$ and $\tilde{w}$ to the function $w(x)$ defined only on the curve L_i). So we have

$$\int\limits_{\omega} R_j v_k dx = \int\limits_{\tau_1}^{\tau_2} d\tau \int\limits_{\nu_1}^{\nu_2} \tilde{R}_j(\tau, \nu) \tilde{w}(\tau) d_k(\nu - \nu_L) d\nu. \tag{7.13}$$

Let us denote by $|L_i|$ the length of the curve L_i. Taking the limit in (7.13) for $k \to \infty$, $\tau_1 \to 0$, $\tau_2 \to |L_i|$ and taking into account relation (7.7) and locality of the support of d_k we have

$$\int_0^{|L_i|} \tilde{R}_j(\tau, \nu_L)\tilde{w}(\tau)d\tau = 0. \tag{7.14}$$

Parameter τ is natural for the curve L_i and then $\tilde{R}_j(\tau, \nu_L)$ is a restriction of R_j to the curve L_i. So we can rewrite (7.14) as

$$\int_{L_i} R_j w dl = 0.$$

Taking into account (7.14), let us revisit the expressions (7.6) and (7.5) restricted onto L_i. Directional derivatives $u'_{\vec{b}}$ and $u''_{\vec{b}\vec{b}}$ are actually first- and second-order derivatives of the function along the curve L_i. If we now revert the sequence in which we obtained (7.6) but only on the curve L_i, it is not hard to see, that the restrictions of the functions $\dot{u}_{j+\frac{1}{4}}|_{L_i} = \dot{u}^i_{j+\frac{1}{4}}$ and $u_{j+\frac{1}{2}}|_{L_i} = u^i_{j+\frac{1}{2}}$ are well-defined and can be found by solving the following 1D variational problem

$$\begin{cases} \text{find} \;\; \dot{u}^i_{j+\frac{1}{4}}, u^i_{j+\frac{1}{2}} \in H^1_0([0, |L_i|]) \;\; \text{such that :} \\ (\dot{u}^i_{j+\frac{1}{4}}, w_i) + \theta\Delta t s_i(\dot{u}^i_{j+\frac{1}{4}}, w_i) = \langle l_i, w_i\rangle - s_i(u^i_j, w_i), \\ u^i_{j+\frac{1}{2}} = u^i_j + \Delta t \dot{u}^i_{j+\frac{1}{4}}, \quad \forall w_i \in H^1_0([0, |L_i|]), \quad i = 1..n, \quad j = 0, 1, ..., \end{cases} \tag{7.15}$$

where

$$\begin{cases} s_i(u, v) = \int_0^{|L_i|} (\mu u'v' + \|\vec{\beta}(\tilde{x}_i(\tau))\|u'v + \sigma uv)d\tau, \\ \langle l_i, v\rangle = \int_0^{|L_i|} f(\tilde{x}_i(\tau))v d\tau \quad \forall u, v \in H^1_0([0, |L_i|]) \end{cases}$$

and $\tilde{x}_i(\tau)$ is a natural parametrization of the curve L_i found from the equation (7.8).

In the same way we can consider the arbitrary curve $K_i \in G$. Using the same procedure we can build corresponding 1D variational problem for that curve

$$\begin{cases} \text{find} \;\; \dot{u}^i_{j+\frac{3}{4}}, u^i_{j+1} \in H^1_0([0, |K_i|]) \;\; \text{such that :} \\ (\dot{u}^i_{j+\frac{3}{4}}, w_i) + \theta\Delta t m_i(\dot{u}^i_{j+\frac{3}{4}}, w_i) = -m_i(u^i_{j+\frac{1}{2}}, w_i), \\ u^i_{j+1} = u^i_{j+\frac{1}{2}} + \Delta t \dot{u}^i_{j+\frac{3}{4}}, \quad \forall w_i \in H^1_0([0, |K_i|]), \quad i = 1..m, \quad j = 0, 1, ..., \end{cases} \tag{7.16}$$

where

$$m_i(u, v) = \int_0^{|K_i|} \mu u'v' d\nu.$$

It is also important to note that if the original problem (2.1) is singularly perturbed by dominated advection, then upon examining the problem sets (7.15) and (7.16), we observe that only the problems in (7.15) will be singularly perturbed in this case. This implies that solving the second set of problems, as given by (7.16), can be performed without the need for any stabilization.

This property is specific to the method we have developed, unlike some other similar schemes, where the splitting is done merely along the X and Y directions. While this method introduces some

overhead, in the context of parabolic problems where we must solve for a large number of time slices, this overhead can be neglected. This is due to the fact that all computations required to construct the curvilinear grid are performed only once, prior to the starting of time integration loop.

To solve the obtained 1D problems approximately, we can apply the finite element method.

Thus, we have transformed the two steps from (6.2) into two sets of independent boundary value problems (BVPs) defined on the different sets of orthogonal curves. Therefore, we need the ability to interpolate the values of a function defined on one set of curves to the other set of curves, orthogonal to the first one. This interpolation is necessary in order to substitute the values $u^i_{j+\frac{1}{2}}$ obtained from (7.15) into (7.16). The latter are technically defined on a different set of curves, which intersect at a series of points (see Fig. 7.1).

7.1 Mapping onto the intermediate grid

In the next two subsections we describe a simple approach for the interpolation of the function values between two systems of curves, defined in the previous sections. It uses the idea from the approach that was used in [5] for interpolation.

Let us define the bounding box for Ω as

$$P = [a,b] \times [c,d] = \\ = \left[\min_{(x_1,x_2)\in\bar{\Omega}} x_1,\ \max_{(x_1,x_2)\in\bar{\Omega}} x_1\right] \times \left[\min_{(x_1,x_2)\in\bar{\Omega}} x_2,\ \max_{(x_1,x_2)\in\bar{\Omega}} x_2\right].$$

Let us cover this bounding box by the rectangular grid of cells $R = \{C_{ij}\}$, defined using M horizontal and N vertical lines (including ∂P). So we have

$$P = \bigcup_{i=1}^{M-1} \bigcup_{j=1}^{N-1} C_{ij},$$

where

$$C_{ij} = [a + (j-1)\delta_{x_1}, a + j\delta_{x_1}] \times [c + (i-1)\delta_{x_2}, c + i\delta_{x_2}] \tag{7.17}$$

and $\delta_{x_1} = (b-a)/N$, $\delta_{x_2} = (d-c)/M$.

Let us define the set of all internal nodes

$$R_p = \{(a + j\delta_{x_1}, c + i\delta_{x_2}) | i = \overline{1, M-2}, j = \overline{1, N-2}\}. \tag{7.18}$$

Consider the sets of internal horizontal (R_h) and vertical (R_v) cell boundary segments

$$R_h = \{C_{i-1,j} \cap C_{ij} | i = \overline{2, M-1}, j = \overline{1, N-1}\},$$

$$R_v = \{C_{i,j-1} \cap C_{ij} | i = \overline{1, M-1}, j = \overline{2, N-1}\}.$$

Fig. 7.4 demonstrates the example of such bounding box with the rectangular grid, covering the initial domain $\bar{\Omega}$. It also depicts the boundary part $\Gamma_{\vec{\beta}}$ and the finite set of the advection curves $B = \{x(\tau, \xi_i)\}_{i=1}^n$, which was defined earlier (see (7.1)).

Our aim is to interpolate the obtained integral curves from the set B and the corresponding values of some arbitrary function $\alpha(x)$, defined on those curves, onto the set of the grid nodes (7.18) of the defined rectangular mesh of cells (7.17).

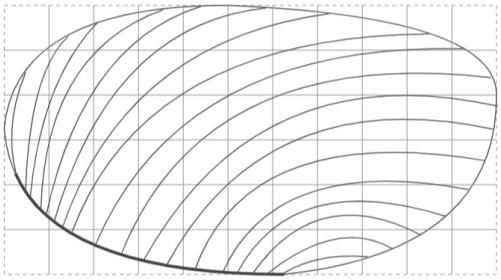

Fig. 7.3: Interpolation grid.

First of all, we should note, that to approximate the curves from B we use some ODE integration methods like Runge-Kutta or similar. Thus, those curves will be defined numerically as a polygonal chains, i.e., piecewise-linear curves, built on the certain sets of nodes. Let us use the same letter B interchangeable to depict the set of those approximated curves, defined as polygonal chains.

Let us obtain the value in certain grid point in the above setting. First of all, let us consider the set of all points lying on the curves (polygonal chains) from the set B

$$B_p = \{x(\tau, \xi_i) | \tau \in T_i, i = \overline{1, n}\},$$

where

$$T_i = \{\tau \in \mathbb{R} | x(\tau, \xi_i) \in \Omega\}.$$

Consider now some point $Q \in R_p$. Let us consider also two horizontal segments $h_1, h_2 \in R_h$, $h_1 \neq h_2$ and two vertical segments $v_1, v_2 \in R_v$, $v_1 \neq v_2$ such that $Q \in h_1 \cap h_2 \cap v_1 \cap v_2$.

Consider the sets of points of the intersection between segments h_1, h_2, v_1, v_2 defined above and the integral curves from the set B. We can obtain them as

$$\tilde{h}_i = h_i \cap B_p, i = 1, 2, \tag{7.19}$$

$$\tilde{v}_i = v_i \cap B_p, i = 1, 2. \tag{7.20}$$

Algorithmically, we can find those points by finding intersection between polygon segment and the grid lines and using linear interpolation on the polygonal chains B.

This construction is depicted on the Fig. 7.4.

Corresponding values of the function $\alpha(x)$ on the sets of points (7.19) and (7.20) in the computer implementation can be found also using linear interpolation between the values of α on the subsequent polygonal chain nodes.

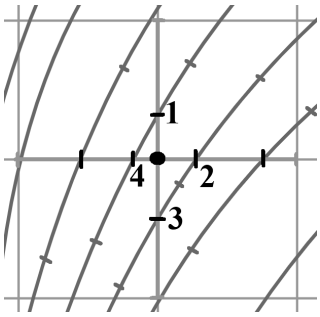

Fig. 7.4: Interpolation grid node.

Consider now four points: $Q_1 \in \tilde{v}_2, Q_2 \in \tilde{h}_2, Q_3 \in \tilde{v}_1, Q_4 \in \tilde{h}_1$, such that each selected point is the closest intersection point to the point Q on a containing segment. Those points are depicted on the Fig. 7.4.

Let us define

$$\lambda_h = \frac{\|Q - Q_4\|}{\|Q_2 - Q_4\|},$$

$$\lambda_v = \frac{\|Q - Q_3\|}{\|Q_1 - Q_3\|}.$$

Using linear interpolation, let us define the approximations to the $\alpha(Q)$

$$\alpha_h = (1 - \lambda_h)\alpha(Q_4) + \lambda_h \alpha(Q_2),$$

$$\alpha_v = (1 - \lambda_v)\alpha(Q_3) + \lambda_v \alpha(Q_1).$$

For the final approximation we consider

$$\alpha(Q) \approx \frac{\alpha_h + \alpha_v}{2}.$$

Thus, taking $\alpha = u^i_{j+\frac{1}{2}}$, we can use described procedure to interpolate the intermediate solution onto the auxiliary rectangular grid.

7.2 Bilinear interpolation

After we have interpolated the values onto the auxiliary rectangular grid, we can easily interpolate the value in arbitrary point and so we are able to map the values generated on $\vec{\beta}$ integral curves onto appropriate $\vec{\gamma}$ curves and vice versa. For that we can use bilinear interpolation on each grid cell C_{ij} to find approximate values in the needed points. Suppose that we want to find the value of the interpolated function at the point $(x_1, x_2) \in \Omega$. We have corresponding bounding box $P = [a, b] \times [c, d]$ and we have the rectangular grid with $N - 1$ and $M - 1$ cells in row and column respectively. If we enumerate cells along the coordinate axes using two indexes (i_{x_1}, i_{x_2}), we can find trivially the cell indexes for the point

$$i_{x_1} = \left\lceil \frac{x_1 - a}{b - a}(N - 1) \right\rceil + \Psi\left(\frac{x_1 - a}{b - a}(N - 1), N - 1 \right)$$

and

$$i_{x_2} = \left\lceil \frac{x_2 - c}{d - c}(M - 1) \right\rceil + \Psi\left(\frac{x_2 - c}{d - c}(M - 1), M - 1 \right),$$

where

$$\Psi(q, m) = \begin{cases} 1, & \text{if } \{q\} = 0 \text{ and } q < m; \\ 0, & \text{otherwise} \end{cases}$$

and by $\{\}$ we denote fractional part of a number.

Suppose, that we have the following corner values of some function w for the cell $C_{i_{x_2} i_{x_1}}$: $p_{11},\ p_{12},\ p_{21}, p_{22}$ (bottom left, bottom right, top left, top right). Then, we can interpolate those values to find approximate value of target function in the point (x_1, x_2)

$$\begin{aligned} I(x_1, x_2) = (1 - \lambda_{x_2})(1 - \lambda_{x_1})p_{11} + \lambda_{x_2}\lambda_{x_1}p_{22} + \\ + (1 - \lambda_{x_2})\lambda_{x_1}p_{12} + \lambda_{x_2}(1 - \lambda_{x_1})p_{21}, \end{aligned}$$

where

$$\lambda_{x_1} = \begin{cases} \left\{ \frac{x_1 - a}{b - a}(N - 1) \right\}, & \text{if } x_1 < b; \\ 1, & \text{otherwise} \end{cases}$$

and

$$\lambda_{x_2} = \begin{cases} \left\{ \frac{x_2 - c}{d - c}(M-1) \right\}, & \text{if } x_2 < d; \\ 1, & \text{otherwise.} \end{cases}$$

8 Numerical experiment

Let us consider 2D problem (2.1) on a unit square $\Omega = (0,1)^2$ with the following data

$$\mu = 1, \vec{\beta}(x_1, x_2) = (-5(x_2+1), 5(x_1+1))^T, \sigma = 1, f(x_1, x_2) \equiv 5.$$

We used the parameter $\theta = 0.5$ in (7.15) and (7.16) for the simulations, which, in fact, defines the classical CrankNicolson scheme. We set $\Delta t = 0.001$ and executed 200 time iterations. Despite the usual operator splitting, time discretization and finite element errors, we also have in our scheme errors resulting from the interpolation mapping between the substeps of the recurrent scheme. Thus, we also wish to explore how this mapping impacts the error of the scheme. To do so, we ran computations with different intermediate rectangular grid sizes $M \times M$ for $M = 15, 25, 50, 75, 100$.

The entire code is written in Python. We measured the relative error in the L^2 norm with respect to the approximate solution, obtained using the FEniCS library as a reference. Fig. 8.1 demonstrates the concentration evolution over time. Figures 8.2 and 8.3 show the dependence of the relative error on the interpolation grid size: per time step and total. The order of convergence with respect to grid size ranges from 1.0 to 1.8, with an average value of 1.26.

Fig. 8.2 demonstrates a significant dependence of the error on the grid size and thus shows that the interpolation step is very critical in the algorithm. One may also note that at the beginning, the error accumulates a bit and then stabilizes. This reflects the fact that our solution to the parabolic problem stabilizes in time and converges to the solution of the stationary problem. At the same time, this shows that the repeated usage of the interpolation step does not cause instability due to error accumulation over time.

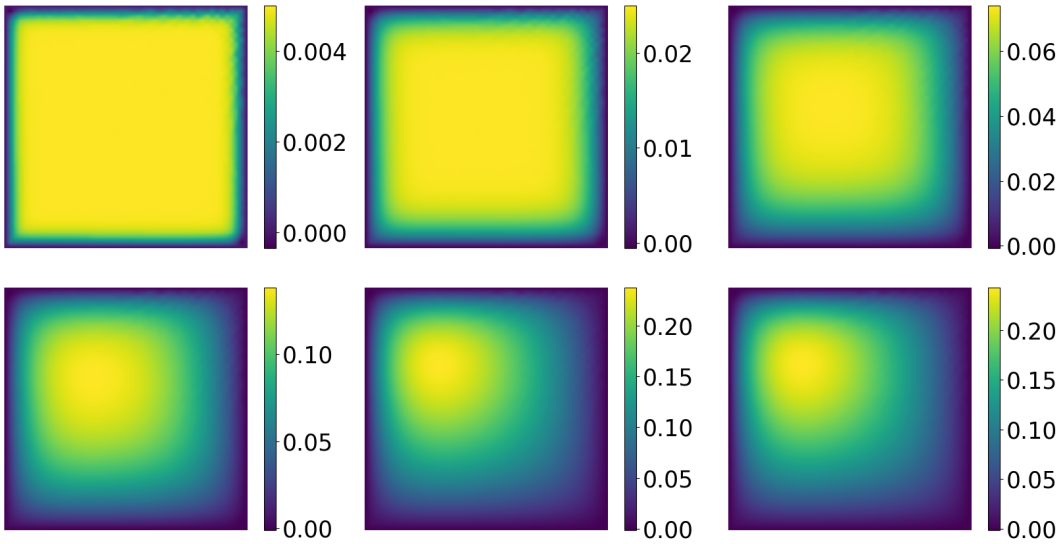

Fig. 8.1: Time evolution of the approximation for time steps 1, 5, 15, 30, 100, 200 (left to right, top to bottom). Intermediate grid size is 50×50.

As a conclusion, we can say that there is a good chance to improve the scheme by enhancing the interpolation step. One possible improvement to the constructed scheme is to eliminate the bilinear interpolation step entirely by directly mapping the intersection points between the two sets of integral curves. Further improvement would also require shifting the finite element nodes

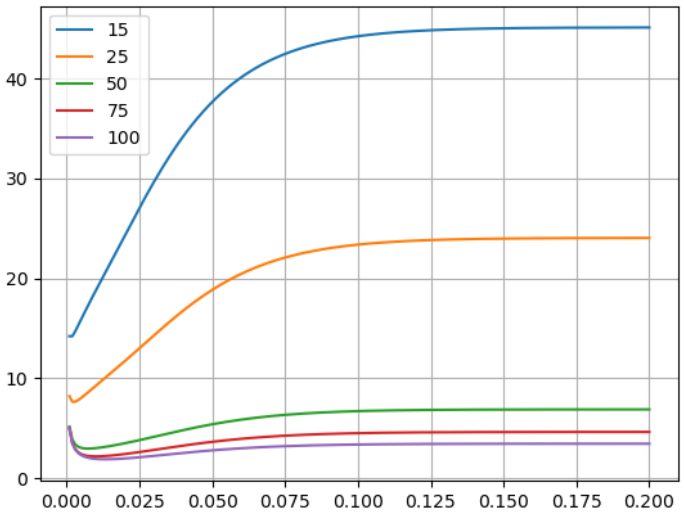

Fig. 8.2: Evolution of the relative error (%) for the approximation over time for different grid sizes M.

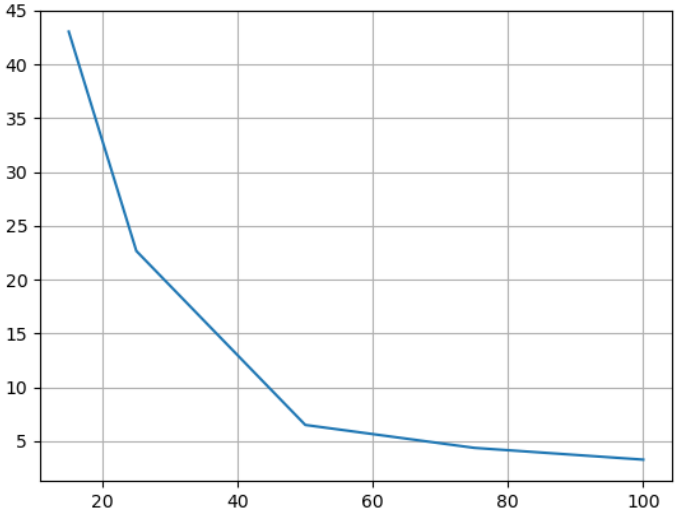

Fig. 8.3: Total (space-time) relative error (%) dependence on the intermediate grid size.

in the 1D problems to the intersection points, thus eliminating any interpolation on the side of the 1D problems defined on curves. It seems reasonable to exploit the fact that the segments of the curves are orthogonal at each intersection point. One idea is to construct a set of "buckets" containing groups of segments with similar "azimuthal angles". This would allow us to quickly identify the segments of the opposite curve group that intersect with a given segment from the other set of orthogonal curves. This idea has not yet been implemented and is also considered for further research and improvement.

9 Discussion on practical applicability to non-rectangular domains

While for testing, we implemented the method only for rectangular domains, it is obvious that, in general, the proposed scheme is applicable to arbitrary domains. The only question is whether it is possible to implement this method efficiently for arbitrary domains.

The major additional problem that arises in the case of arbitrary domains is the problem of efficient detection of intersections between the integral curves of $\vec{\beta}$ (and similarly $\vec{\gamma}$) vector field with

the boundary of the domain. Such intersections are needed to identify the end of the curve.

In general, the mentioned problem can be considered a *line clipping problem*. There are some well-known algorithms for clipping the line segment by a "window", such as the CyrusBeck, Cohen-Sutherland and LiangBarsky algorithms. However, not all of them are applicable for non-rectangular clipping windows. For instance, for the mentioned algorithms, only the CyrusBeck algorithm can be applied when the domain is non-rectangular. There are also new developments in this area, such as a new algorithm from Vaclav Skala [16]. An article reviewing these algorithms is also provided by the same author [15].

Typically, we can expect three possible complexities for such algorithms with respect to clipping a single line segment: $O(N)$, $O(\log N)$ and $O(1)$, where N is the number of vertices in the clipping polygon. The $O(1)$ complexity might be surprising, but it is related to a different algorithmic model. In general, the worst-case $O(N)$ complexity can be seen in a simple search over all possible line segments of a polygon. The $O(\log N)$ complexity can be achieved for convex polygons by leveraging the fact that polygon vertices are typically given in sorted (counter)clockwise order. However, algorithms with both complexities are not suitable for our case, since we will have many segments from integral curves to test against the boundary segment, resulting in a huge overall complexity.

The best complexity we can achieve is $O(1)$, which is the result of special preprocessing before running the test. In short, a typical algorithm of this type uses some auxiliary rectangular grid (as used in previous sections) and generates a list of possible matches between boundary segments and those rectangular cells. This is done as preprocessing and the obtained data can be used to test the intersection many times with different line segments against the same boundary polygon. Since testing if a point is located in a specific rectangle from the grid is $O(1)$, we can simply skip the "real" intersection test in almost all cases (i.e., when the given point is located far away from the boundary). In general, such algorithms also use more sophisticated constructs, such as projective/dual space representation, etc. More details can be found, for example, in [15, 17].

The last type of algorithms is the only one that can be used in combination with proposed algorithm for building the curvilinear grid. An important note is that the complexity of constructing such a preprocessed dataset is $O(N)$.

10 Conclusions

In this article we have developed the splitting method for the advection-diffusion-reaction model, which can be used to transform the two-dimensional problem into a set of smaller one-dimensional problems that can be solved independently. This approach will reduce the number of qubits required when using a quantum-assisted stabilized/adaptive-stabilized scheme for solving the obtained 1D problems and it was specifically designed for that.

Possible improvements to the method are discussed.

Declarations

Conflict of Interest.
The authors declare that there are no conflicts of interest.
Funding.
This research received no any funding.
Author Contributions.
Conceptualization: R.D.; Algorithm: R.D.; Formal analysis: R.D., H.S.; Methodology: R.D., H.S.; Software: R.D.; Validation: R.D., H.S.; Writing original draft: R.D.; Writing – review and editing: R.D., H.S.; Visualization: R.D.; Supervision: H.S.

Additional information

H.S. is a member of the Editorial Board for JANA. The paper was handled by another Editor and has undergone a rigorous peer review process. H.S. was not involved in the journal's peer review, or decisions related to this manuscript.

References

1. Blanes, S., Casas, F., Escorihuela-Tomàs, A.: Runge–Kutta–Nyström symplectic splitting methods of order 8. Appl. Numer. Math. **182**, 14-27 (2022). doi:10.1016/j.apnum.2022.07.010
2. Bujanda, B., Moreta, M.J., Jorge, J.C.: New fractional step Runge–Kutta–Nyström methods up to order three. Appl. Math. Comput. **366**, 124743 (2020). doi:10.1016/j.amc.2019.124743
3. Chorin, A.J.: Numerical solution of the Navier-Stokes equations. Math. Comput. **22**(104), 745-762 (1968)
4. Drebotiy, R., Shynkarenko, H.: Convergence of the directional diffusion splitting method. [Preprint] arXiv:2411.12305 [math.NA] (2024). doi:10.48550/arXiv.2411.12305
5. Drebotiy, R., Shynkarenko, H.: Fully parallel algorithm for solution of advection-reaction Cauchy problem in one finite element regularization scheme. Manufacturing Processes. Actual Problems. **1**, 35-44 (2023)
6. Drebotiy, R., Shynkarenko, H.: Heuristic choice of the regularization parameter for optimal stabilization of the finite element approximations. Mathematical Methods and Physicomechanical Fields. **66**(1–2), 206-221 (2023)
7. Drebotiy, R., Shynkarenko, H.: Modified heuristic criterion for parameter choice for one stabilization scheme of the finite element method. Journal of Applied and Numerical Analysis. **2**, 41-55 (2024)
8. Drebotiy, R., Shynkarenko, H.: Quantum-assisted $h\lambda$-adaptive finite element method. Partial Differential Equations in Applied Mathematics. **13**, 101120 (2025). doi:10.1016/j.pdeapm.2025.101120
9. Geiser, J.: Iterative operator-splitting methods with higher-order time integration methods and applications for parabolic partial differential equations. Journal of Computational and Applied Mathematics. **217**(1), 227–242 (2008). doi:10.1016/j.cam.2007.06.028
10. Harrow, A.W., Hassidim, A., Lloyd, S.: Quantum algorithm for solving linear systems of equations. Physical Review Letters. **103**(15), 150502 (2009). arXiv:0811.3171
11. Lax, P.D.: Functional Analysis. John Wiley & Sons, New York (2002)
12. Marchuk, G.I., Björck, A., Thomée, V.: Handbook of Numerical Methods. Volume 1. Splitting and alternating direction methods. In: Ciarlet, P.G., Lions, J.L. (eds.) Elsevier Sci. Publ. B.V. (North-Holland) (1990)
13. McLachlan, R.I., Quispel, G.R.W.: Splitting methods. Acta Numerica. **11**, 341-434 (2002). doi:10.1017/S0962492902000053
14. Nielsen, M., Chuang, I.: Quantum Computation and Quantum Information: 10th Anniversary Edition. Cambridge University Press, Cambridge (2010). doi:10.1017/CBO9780511976667
15. Skala, V.: A brief survey of clipping and intersection algorithms with a list of references (including triangle-triangle intersections). Informatica. **34**(1), 169-198 (2023). doi:10.15388/23-INFOR508
16. Skala, V.: A new fully projective $\mathcal{O}(\log N)$ line convex polygon intersection algorithm. The Visual Computer. **41**, 1241-1249 (2025). doi:10.1007/s00371-024-03413-3
17. Skala, V.: Line clipping in $\mathbb{E}^2$ with $\mathcal{O}(1)$ processing complexity. Computer Graphics. **20**(4), 523-530 (1996).doi:10.1016/0097-8493(96)00024-6

18. Trushevsky, V.M., Shynkarenko, H.A., Shcherbyna, N.M.: Finite Element Method and Artificial Neural Networks: Theoretical Aspects and Application. Ivan Franko National University of Lviv, Lviv (2014). ISBN 978-617-10-0127-5 (in Ukrainian)
19. Wang, Ha., Wang, Hu., Gao, F., Zhou, P., Zhai, Z.: Literature review on pressure–velocity decoupling algorithms applied to built-environment CFD simulation. Building and Environment. **143**, 671–678 (2018). doi:10.1016/j.buildenv.2018.07.046
20. Westra, D., Heinrich, J., Saxon, J.: The fractional step method applied to simulations of natural convective flows. [Preprint] NASA Technical Reports Server (2002). https://ntrs.nasa.gov/citations/20020091875
21. Zhou, P., Wang, H., Dai, Y., Xue, Y., Huang, C.: On the fast fluid dynamics and fractional step methods to predict the coupled indoor temperature and velocity fields. Building and Environment. **229**, 109959 (2023). doi:10.1016/j.buildenv.2022.109959

Received 08.04.2025
Revised 20.05.2025