On the Numerical Approximation and Optimization Techniques for Solving an Inverse Cauchy Problem of Viscous-Burgers' Equation

Mohammed A. Hilal[1,*], **Faris M. Alwan**[2], **Alaa Adnan Auad**[3]
[1]Baquba Technical Institute, Middle Technical University (MTU), Baghdad, Iraq
[2]Collage of Administration and Economics, University of Baghdad, Iraq
[3]Department of Mathematics, College of Education for pure science, University of Anbar, Iraq
Emails: mohammed_azeez_hilal@mtu.edu.iq; Faris.m.alwan@coadec.uobaghdad.edu.iq; alaa.adnan.auad@uoanbar.edu.iq

Abstract

This paper deals with some inverse problems for nonlinear time-dependent PDEs in one spatial dimension, we investigate an inverse Cauchy problem that is settled by the nonlinear viscous Burgers equation. The viscous Burgers equation is a partial differential equation that is encountered in fluid dynamics studies, particularly in the domain of upward flow. The simplified model of the viscous Burgers equation explains the behavior of incompressible viscous fluid. The inverse Burgers problem belongs to a class of problems called ill-posed problems, which implies that there may be multiple sets of initial and/or boundary conditions that result in the same solution of the Burgers equation.
To obtain robust and reliable solutions, it is essential to use regularization and cross-validation methods. However, it is often difficult to solve analytically, so numerical approaches are developed to overcome this difficulty. Domain decomposition (DDM) was used with alternative iterative methods. We performed a numerical reconstruction of the velocity and normal stress tensor that were vanished on an inaccessible part of the boundary using the over-prescribed noisy data obtained on the other accessible part of the boundary.

Keywords: Burger Equation; Inverse Problem; Cauchy Problem; Operations Research (OR); Boundary Condition

1 Introduction

Inverse problems in applied mathematics, particularly in the fields of fluid mechanics, geophysics, and medical imaging, consist in determining unknown parameters from partial or

indirect measurements. Solving these problems requires efficient and accurate methods, particularly when conditions are complex and data are noisy. Among the common approaches, iterative domain decomposition methods, such as Dirichlet-Neumann, Neumann-Neumann, Robin, and Agoshkov-Lebedev, are widely used to improve the convergence of solutions in multidomain settings. There is an essential and fundamental differential equation in fluid mechanics, known as the Burgers equation. Bateman[1] first derived this equation in 1925, and then Burger[?] in 1948. It has been widely used in the literature to describe turbulent flow in a channel resulting from the interaction of the opposing processes of convection and diffusion. The Burgers equation is the incompressible Navier-Stokes equation without taking into account pressure and strain. It is also used in many areas of daily life such as circulations,[2] shocks[4] and gas dynamics.[5] Analytical and numerical solutions for the Burgers equation have been studied and developed by many researchers. Methods such as the tanh-coth method, the differential transformation method, the Backlar transformation method[6] and the Hopf-Cole transformation [?]and[7] have allowed to obtain analytical solutions, as well as for the first time, Fletcher ,[8] the finite difference method, spectral methods and the variational iteration method .[9] The objective of this study is to analyze the solvability and numerical resolution of a series of inverse problems involving the nonlinear Burgers equation.
Numerical investigations of the inverse Berger equation are used in many fields, such as fluid mechanics, nonlinear waves and meteorology. However, it is essential to use numerical techniques such as the finite element method, the finite difference method or the spectral element method in order to find a numerically stable solution. With these techniques, it is possible to interpret the Burger equation and to solve the resulting system of equations in order to determine its initial and / or boundary conditions. This article explores the application of these iterative methods to solve an inverse diffusion problem. By combining these techniques with acceleration parameters, the objective is to test their effectiveness in terms of convergence speed and solution accuracy. We successively study the resolution of a Dirichlet and Neumann problem through several algorithms, each adapted to a specific type of domain decomposition[10] and.[11]

The numerical results obtained are analyzed in terms of the error between the exact solution and the approximate solution for each method, as a function of the number of iterations. We also present a comparative study of the different acceleration strategies, with emphasis on the optimization of the acceleration parameters, such as the θ parameter, in order to maximize the convergence speed without compromising the accuracy of the solution. In conclusion, this study demonstrates the effectiveness of iterative domain decomposition methods in solving inverse problems, while highlighting the importance of optimizing the acceleration parameters to obtain accurate results in a reduced number of iterations.

The outline of this article is as follows: Section 2 introduces the form of direct Burgers' Equation and its components. ,section 3, Conversion of Burgers' equation into weak form for numerical methods (e.g., finite element, finite difference). In section 4, discussion of boundary conditions (Dirichlet, Neumann, Robin) for solving the problem,In section 5, formulation of the inverse problem: determining boundary conditions from the solution. In section 6,7 we formulate the ICP as an optimization problem. In section 8, splitting the domain into subdomains and solving locally using iterative methods (e.g., alternating iterations). Section 9 study numerical experiments and application of Dirichlet-Neumann, Neumann-Neumann, Robin, and Agoshkov-Lebedev methods for solving the problem.

while section 10 we are study of error behavior across iterations, particularly for inverse problems and comparison of solutions and errors for each method (direct and inverse).. In section 11, we are find numerical results and finding the optimal acceleration parameter θ for faster convergence, section 12 we are introduce discussion our results, Agoshkov-Lebedev method is the most efficient for inverse problems, outperforming other methods. Lastly, section 13 summarize the conclusion that Agoshkov-Lebedev method with optimized parameters offers the best performance.

2 Direct Burger's equation

The one- dimensional unsteady viscous Burger's equation is a fundamental partial differential equation used in various fields such as fluid dynamics and nonlinear wave propagation. It's often written in the form: (Eq. 1)

$$\frac{\partial u}{\partial t} + u\frac{\partial u}{\partial x} = \kappa\frac{\partial^2 u}{\partial x^2} \tag{1}$$

Where: $u(x,t)$ represents the velocity field (is the solution we are looking for) , t is the time, κ is the kinematic viscosity, which determines the strength of the diffusive term. It's a positive constant, and $\kappa\frac{\partial^2 u}{\partial x^2}$ represents diffusion term. and κ is the spatial variable.

3 Solve the Burger's equation using weak formulation.

To solve the Cauchy-Burger equation using the weak form method, the equation must be converted into a suitable form that can be handled using numerical methods such as the finite element method or the finite difference method. The Cauchy-Burger equation is one of the partial differential equations used in modeling physical phenomena such as heat distribution or fluid flow, and it appears in the form of a nonlinear partial differential equation.
We must rewrite the Burger's equation in its weak form in order to solve it , In function spaces less regular than the classical L_2 space, such as Sobolev spaces, the solution can be approximated thanks to the weak form of the equation.

- Governing Equation (Classical Form): Burger's equation with a diffusion term is expressed in classical form (Eq. 2):
- Weak Formulation: First, multiply using a test function v_x , which is usually selected from the same function space as the solution u, multiply both sides of the equation to obtain the weak form, then integrate over the domain $\Omega = [0, L]$. (Eqs. 3).

$$\int_0^T \int_0^L \left(v(x)\frac{\partial u}{\partial t} + v(x)u\frac{\partial u}{\partial x} \right) dxdt = \int_0^T \int_0^L v(x)\nu\frac{\partial^2 u}{\partial x^2}dxdt \tag{2}$$

$$\int_0^T \int_0^L v(x)\frac{\partial u}{\partial t}dxdt = -\int_0^T \int_0^L \frac{\partial v}{\partial t}udx \tag{3}$$

- Implement integration by components: (Eq. 5) Integrate by parts to deal with the term involving the second spatial derivative later. To begin, remember the standard identity for the second derivative's integration:

$$\int_0^T \int_0^L \varphi \left(\frac{\partial u}{\partial t} + u \frac{\partial u}{\partial x} \right) dx\, dt = \int_0^T \int_0^L \nu \varphi \frac{\partial^2 u}{\partial x^2}\, dx\, dt \tag{4}$$

3.1 Integrating the left-hand side:

- First term: $\int_0^L \varphi \frac{\partial u}{\partial t}\, dx$

Using integration by parts in x-direction, and assuming that $u(x,t)$ and $\varphi(x,t)$ vanish at the boundaries (or are periodic, depending on boundary conditions), we get:

$$\int_0^L \varphi \frac{\partial u}{\partial t}\, dx = \frac{d}{dt} \int_0^L \varphi u\, dx$$

- Second term: $\int_0^L \varphi u \frac{\partial u}{\partial x}\, dx$

Again, using integration by parts:

$$\int_0^L \varphi u \frac{\partial u}{\partial x}\, dx = - \int_0^L \frac{\partial \varphi}{\partial x} u^2\, dx + \left[\varphi u^2 \right]_0^L$$

Assuming no boundary contributions or periodic boundaries, the boundary term vanishes. Therefore, this term simplifies to:

$$\int_0^L \frac{\partial \varphi}{\partial x} u^2\, dx$$

3.2 Integrating the right-hand side:

Third term: $\int_0^L \nu \varphi \frac{\partial^2 u}{\partial x^2}\, dx$

Using integration by parts on the second derivative term (Eq. **??**):

$$\int_0^L \varphi \frac{\partial^2 u}{\partial x^2}\, dx = - \int_0^L \frac{\partial \varphi}{\partial x} \frac{\partial u}{\partial x}\, dx + \left[\varphi \frac{\partial u}{\partial x} \right]_0^L \tag{5}$$

Again, assuming no boundary contributions or periodic boundary conditions, this simplifies to:

$$\int_0^L \frac{\partial \varphi}{\partial x} \frac{\partial u}{\partial x}\, dx$$

3.3 Putting everything together:

We now have the weak formulation (Eq. 6):

$$\frac{d}{dt}\int_0^L \varphi u\, dx + \int_0^L \frac{\partial \varphi}{\partial x}\left(\nu\frac{\partial u}{\partial x} - u^2\right) dx = 0 \tag{6}$$

This equation represents the **weak form** of the Burgers' equation. It expresses the evolution of the solution $u(x,t)$ in terms of its interaction with test functions $\varphi(x,t)$, and it can be used to derive numerical methods (such as finite element or finite difference methods) to approximate the solution of the equation.

4 Boundary Conditions: Types and Choices

To solve this equation over a specific domain, you need to specify the boundary conditions at the endpoints of the spatial domain [0, L] . The boundary conditions can vary depending on the physical scenario and assumptions made about the flow.

- Dirichlet Boundary Conditions (Prescribed value of u): This type of boundary condition specifies the value of u at the spatial boundaries.

$$u(0,t) = u_0, \quad u(L,t) = u_L$$

Physical meaning: This implies that the velocity of the fluid (or the quantity modeled by u) is fixed at the boundaries of the domain at all times. For example, in a pipe, this could represent fixed inflow and outflow velocities.
- Example: If you are modeling the velocity of a fluid flowing through a pipe, you might set the inflow and outflow velocities to fixed values.

- Neumann Boundary Conditions (Prescribed derivative of u): This type of boundary condition specifies the gradient (or derivative) of u at the boundaries.

$$\frac{\partial u}{\partial x}(0,t) = 0, \quad \frac{\partial u}{\partial x}(L,t) = 0$$

- Physical meaning: This corresponds to(no flux) at the boundaries, implying that the velocity is constant (no change) in the direction normal to the boundary. This could represent situations where there is no shear or pressure gradient at the boundaries.

- Example: In a pipe with fluid flow, this might represent the case where the fluid velocity at the boundaries is zero (like in a channel with no-slip conditions at the walls).

- Periodic Boundary Conditions: This type of boundary condition is appropriate when the domain is periodic, meaning the solution "wraps around" the boundaries and repeats.

$$u(0,t) = u(L,t), \quad \frac{\partial u}{\partial x}(0,t) = \frac{\partial u}{\partial x}(L,t)$$

- Physical meaning: This implies that the solution is periodic in space. For example, this could model a situation where the system repeats over time, like waves propagating in a loop or a fluid flow in a closed loop (e.g., a circular pipe or ring).

Example: Periodic boundary conditions might be used when modeling a flow that circulates, such as in a recirculating channel or periodic turbulence.

- Mixed Boundary Conditions: In some cases, you might have a combination of Dirichlet and Neumann conditions at the two ends of the domain:

$$u(0,t) = u_0, \quad \frac{\partial u}{\partial x}(L,t) = 0$$

- Physical meaning: This would mean that u is fixed at one boundary (e.g., the left boundary), while the derivative is zero at the other boundary (e.g., no flux condition at the right boundary).
Example: This might be used when you want to model a flow with a specified inlet velocity at one end, while there is no flux or gradient of velocity at the other end.

4.1 Initial Conditions

In addition to boundary conditions, you also need to specify the initial condition to complete the problem. The initial condition defines the state of the system at t = 0. For the Burgers' equation, an example initial condition could be:

$$u(x,0) = f(x)$$

Where f(x) is some function that defines the initial profile of u across the spatial domain. The initial condition could be:

- Dirichlet: Specifies the velocity at the boundaries.

$$u(0,t) = u_0, \quad u_{(L,t)} = u_L$$

- Neumann: Specifies the flux (derivative of the velocity) at the boundaries. $\frac{\partial u}{\partial x}(0,t) = 0$, $\frac{\partial u}{\partial x}(L,t) = 0$
- Periodic: The solution is assumed to repeat across boundaries. u(0,t) = u(L,t), $\frac{\partial u}{\partial x}(0,t) = \frac{\partial u}{\partial x}(L,t)$
- Mixed: A combination of Dirichlet and Neumann conditions at opposite boundaries. $u(0,t) = u_0$, $\frac{\partial u}{\partial x}(L,t) = 0$

4.2 Physical Considerations

For a fluid flow through a channel or pipe, typical boundary conditions would be Dirichlet conditions at the inlet and outlet to specify the velocity (e.g., $u(0,t)$ = u_0 and $u_(L,t) = u_L$.)
For a closed system or when modeling waves propagating, periodic boundary conditions might be more appropriate. If you want to model a no-slip condition at the boundaries of a pipe, Neumann conditions (no flux) could be used.
Choosing the right boundary conditions depends on the physical problem being modeled and the assumptions you make about the flow or system behavior at the boundaries.

5 Inverse Cauchy problem for the Burgers' equation (,[1213])

This paper focuses on the inverse system of Burgers' equations and seeks to determine the boundary conditions in this system for the Burgers equation as follows: Let us consider:

$$u_t = \frac{\partial u}{\partial t}, \quad u_x = \frac{\partial u}{\partial x}, \quad u_{xx} = \frac{\partial^2 u}{\partial x^2}$$

$$\nu_t = \frac{\partial \nu}{\partial t}, \quad \nu_x = \frac{\partial \nu}{\partial x}, \quad \nu_{xx} = \frac{\partial^2 \nu}{\partial x^2}$$

$$I.P \begin{cases} u_t - u_{xx} + \lambda u u_x + \alpha(uv)_x = 0, \quad 0 < x < 1, \quad 0 \le t \le T, \\ v_t - v_{xx} + \lambda v v_x + \beta(uv)_x = 0, \quad 0 < x < 1, \quad 0 \le t \le T. \\ \text{with the initial conditions:} \\ u(x,0) = f_1(x), \quad v(x,0) = f_2(x), \quad 0 < x < 1, \\ \text{and the boundary conditions:} \\ u(0,t) = p_1(t), \quad v(0,t) = p_2(t), \quad 0 \le t \le T, \\ u(1,t) = q_1(t), \quad v(1,t) = q_2(t), \quad 0 \le t \le T, \\ u(x^*,t) = g_1(t), \quad v(x^*,t) = g_2(x), \quad 0 < x^* < 1, \quad 0 \le t \le T. \end{cases} \tag{7}$$

The overspecified data: $u(x^*,t) = g1(t), \quad v(x^*,t) = g2(x), \quad 0 < x^* < 1, \quad 0 \le t \le T$
Where λ is the real constant, α and β arbitrary constants depend on the system parameters such as the Peclet number stokes the velocity of particles due to gravity and the Brownian diffusivity. Also, T represents the final time, $\omega = \{(x,t) : x \in [0,1] = \Omega, \quad t \in [0,T]\}, f1(x), f2(x), p1(t), p2(t)$ and $q1(t), q2(t)$ are given continuous functions. The boundary conditions p1(t), and p2(t) are unknown and are to be determined from overspecified data $u(x^*,t) = g1(t), v(x^*,t) = g2(x)$. We seek the functions $u(x,t), v(x,t)$ and$p1(t), p2(t)$. For two unknown boundary conditions $p1(t), p2(t)$. In other words, we will consider the following problem: given initial and boundary conditions on a part of a model domain, as well as the physical parameters inside the model domain in the inverse Burgers equation, we need to recover the missing physical parameters, which are unknown in the class of initial functions.
We focus mainly on the numerical treatment of parameter identification problems from the inverse Burgers equation, such as the numerical parameter identification methods using the spatial finite element discretization and domain decomposition algorithms that we introduced in the inverse problem. Inverse problems are generally much more vulnerable to

numerical instability than the solution of direct problems, and even to the ultra-sensitivity of the problem function to perturbations, which implies that a small noise can give a solution whose input-output relation is strongly altered. Noise (perturbations) can be explained by errors induced in a source or a boundary condition, which are always present in reality. This point of view is particularly relevant for numerical procedures aimed at solving poorly tuned problems.

6 Operations Research Techniques to Solve the Inverse Cauchy Problem

In Operations Research (OR), we often use optimization methods to solve inverse problems. The main approach is to formulate the ICP as an optimization problem and solve it using standard OR techniques like gradient descent, conjugate gradient, or least squares optimization. Additionally, regularization methods (such as Tikhonov regularization) are crucial to stabilize the ill-posedness.[?]

7 Optimization Algorithm

We can employ common gradient-based optimization algorithms to address the optimization problem. Our goal is to iteratively $g(x)$ to minimize the objective function $J_{reg}(g)$. Several algorithms that are frequently utilized include:[?]

- Gradient Descent: Updates the solution g(x) iteratively by moving in the direction of the negative gradient. $g^{(k+1)} = g^{(k)} - \alpha \nabla J_{reg}(g^{(k)})$

  where α is the step size and $\nabla J_{reg}(g^{(k)})$ is the gradient of the objective function.

- Conjugate Gradient Method: An effective technique for resolving sparse, large linear equation systems that come up during optimization.

- Levenberg-Marquardt Algorithm:An approach that is widely used for nonlinear least-squares problems and strikes a balance between Gauss-Newton and gradient descent.

The Operations Research framework for solving the Inverse Cauchy Problem for the Burgers' equation can be outlined as follows:
1. Model Formulation:
- The forward problem is defined as the Burgers' equation and initial condition.
- Construct the inverse problem as an optimization problem with the goal of reducing the difference between calculated and observed data.
2. Numerical Solution:
- To solve the Burgers' equation forward in time, start with an assumed initial condition and work your way through it using finite element or other numerical techniques.
3. Optimization:
- By changing the initial condition g(x) to correspond with the observed data, optimization

algorithms can be used to solve the inverse problem.
- Use regularization to keep the solution stable and avoid overfitting to data that is noisy.
We can effectively and efficiently solve large-scale inverse problems, like the ICP for Burgers' equation, by combining the domain decomposition technique with optimization methods for inverse problem solving. Accurate recovery of the initial conditions is guaranteed by optimization algorithms like gradient descent or conjugate gradient, and domain decomposition enables parallel computation, which lessens the overall computational load. By using regularization, the solution is further stabilized and overfitting to noisy data is avoided. When combined, these techniques provide a strong foundation for addressing challenging inverse issues in operations research, especially when dealing with physical systems that are described by nonlinear partial differential equations.

8 Domain decomposition method ,[10] ?

To solve the set of inverse problems (8), we are chose the domain decomposition approach which has been developed by dividing the solution domain into several subdomains and then solving the problem locally in each of these subdomains. Subsequently, the local solutions are combined to obtain the global solution of the problem. The domain decomposition method solves the inverse Cauchy problem of the Viscous-Burgers equation by following these step by step approach to the problem. It is possible to divide the domain into several subdomains, each section being considered as a subdomain. The Viscous-Burgers equation can be solved locally in each subdomain using one of the available numerical methods, such as the finite difference method or the finite element method. After collecting the local solutions in each subdomain, it is possible to combine them in order to obtain the global solution of the problem. This step may involve solving an adjustment problem in order to guarantee the continuity of the global solution. This approach can help solve complex problems by reducing the problem size and employing efficient numerical methods for each domain.
The domain decomposition method can be easily accelerated using Krylov subspace methods (as an instance, the conjugate gradient method for symmetric positive definite matrices ,[14] GMRES ,[15] BiCGSTAB for non-symmetric ones [17]). The domain decomposition method is then employed as Krylov technique preconditioners. The BiCGSTAB method was utilized, which was preconditioned by the additive Schwarz method with or without a coarse grid.
By combining optimization methods for solving inverse problems with the domain decomposition technique, we can efficiently and effectively solve large-scale inverse problems, such as the ICP for Burgers' equation. Optimization algorithms, such as gradient descent or conjugate gradient, ensure that the initial conditions are recovered accurately, while domain decomposition allows for parallel computation, reducing the overall computational burden. The use of regularization further stabilizes the solution, preventing overfitting to noisy data. Together, these methods offer a powerful framework for tackling complex inverse problems in Operations Research, particularly in physical systems described by nonlinear partial differential equations.

This paper examines the application of the alternating iteration domain decomposition method based on the updated boundary conditions of the subdomains already solved at each iteration.
Problem Setup: We have the following equations:

- For

$$u(x,t) = u_t - u_{xx} + \lambda u u_x + \alpha(uv)_x = 0, \quad 0 < x < 1, \quad 0 \le t \le T$$

- For

$$v(x,t) = v_t - v_{xx} + \lambda v v_x + \beta(uv)_x = 0, \quad 0 < x < 1, \quad 0 \le t \le T$$

- Assume we divide the domain (0,1) into two subdomains:
  - Subdomain 1: $\Omega_1(0, x^*)$
  - Subdomain 2: $\Omega_2(0, x^*)$
  In Ω_1

  $u_t - u_{xx} + \lambda u u_x + \alpha(uv)_x = 0, \quad 0 \le x \le x^*, v_t - uv_{xx} + \lambda v v_x + \beta(uv)_x = 0, \quad 0 \le x \le x^*.$

  In Ω_2

  $u_t - u_{xx} + \lambda u u_x + \alpha(uv)_x = 0, \quad x^* \le x \le 1, v_t - uv_{xx} + \lambda v v_x + \beta(uv)_x = 0, \quad x^* \le x \le 1.$

- Boundary and interface conditions:
  - At the boundaries:

  $u(0,t) = p_1(t), \quad v(0,t) = p_2(t), \quad 0 \le t \le T, u(1,t) = q_1(t), \quad v(1,t) = q_2(t), \quad 0 \le t \le T.$

  - At the interface

$$x = x^* : u(x^*, t) = g_1(t), \quad v(x^*, t) = g_2(t),$$

  Initial conditions:

$$u(x,0) = f_1(x), \quad v(x,0) = f_2(x), \quad 0 < x < 1.$$

- Solving subdomain problems:
  For each subdomain, use a suitable numerical method such as Finite Element Method to discretize the equations.

- Discretization:
  - Let Δx be the spatial step and Δt be the time step.
  - Use explicit finite elements methods for u and v :

$$u_i^{n+1} = u_i^n + \Delta t \left(u_{xx} - \lambda u u_x - \alpha(uv)_x\right)$$
$$v_i^{n+1} = v_i^n + \Delta t \left(v_{xx} - \beta v v_x - (uv)_x\right)$$

- Iterative Coupling:
  - Initialize the solution in each subdomain using the initial conditions.
  - Solve the equations in both subdomains, using the interface conditions to update the solutions iteratively:
  - At each time step, compute $u(x^*, t)$ and $v(x^*, t)$from the solution in both subdomains.
  - Update u and v both subdomains using the values obtained at the interface.

9 Numerical experiments for square domain

To solve the Burger-viscous problem let us re-write our problem as adomain decomposition with Dirichlet to Neumann conditions, let us consider our domain as a rectangular like the one depicted below. (see Fig. 1),
such that $\Omega =]-1,1[\times]-1,1[$ .with boundary $\partial\Omega = \Gamma_1 \cup \Gamma_2 \cup \Gamma_3 \cup \Gamma_4$ our problem is to

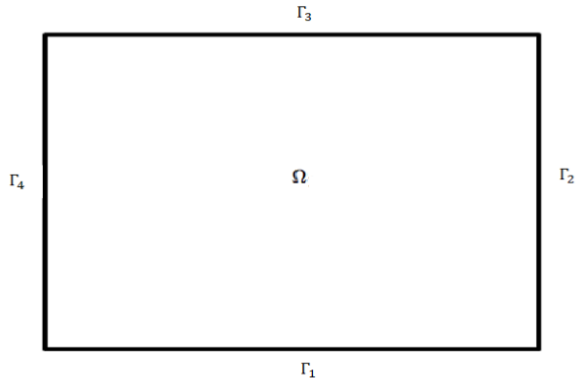


Figure 1: Domain.

find u solution of the following problem:

$$\begin{array}{rcl} -\Delta u^{(k+1)} & = & f \quad \text{in} \quad \Omega_1 \\ u^{(k+1)} & = & u_e \quad \text{on} \quad \Gamma_3 \quad \text{and} \quad \Gamma_2 \\ \frac{\partial u^{(k+1)}}{\partial n} & = & \frac{\partial u_e}{\partial n} \quad \text{on} \quad \Gamma_1 \\ u^{(k+1)} & = & \lambda^{(k)} \quad \text{on} \quad \Gamma_4 \end{array} \tag{8}$$

To solve this problem let us re-write our problem as adomain decomposition with divided our domain Ω into two domains $\Omega_1 \cup \Omega_2$ such that(see Fig. 2): Roache Such that:
$\partial\Omega = \Gamma_1 \cup \Gamma_2 \cup \Gamma_3 \cup \Gamma_4 \cup \Gamma_5 \cup \Gamma_6 \cup \Gamma_7$ where
$\Gamma_1 = \{(x,y) \in \partial\Omega_1, \quad x = t, \quad y = 0\}$, $\Gamma_2 = \{(x,y) \in \partial\Omega_1, \quad x = 1, \quad y = t\}$,
$\Gamma_3 = \{(x,y) \in \partial\Omega_1, \quad x = t, \quad y = 1\}$, $\Gamma_4 = \{(x,y) \in \partial\Omega_1, \quad x = 0, \quad y = t\}$,
$\Gamma_5 = \{(x,y) \in \partial\Omega_2, \quad x = -1, \quad y = 1\}$, $\Gamma_6 = \{(x,y) \in \partial\Omega_2, \quad x = -1, \quad y = t\}$,
$\Gamma_7 = \{(x,y) \in \partial\Omega_2, \quad x = -t, \quad y = 0\}$.

So, we will write the equation in simplify formula as the following:

9.1 The Dirichlet-Neumann method

This method is summarized by the following algorithm :[18] for $\lambda^{(0)}$ given, we resolve for each $k \geq 0$, see (Eq. 9)

$$\begin{cases} -\Delta u_1^{(k+1)} = f \quad \text{in} \quad \Omega_1 \\ u_1^{(k+1)} = u_e \quad \text{on} \quad \Gamma_3 \quad \text{and} \quad \Gamma_2 \\ \frac{\partial u_1^{(k+1)}}{\partial n} = \frac{\partial u_e}{\partial n} \quad \text{on} \quad \Gamma_1 \\ u_1^{(k+1)} = \lambda^{(k)} \quad \text{on} \quad \Gamma_4 \end{cases} \qquad \begin{cases} -\Delta u_2^{(k+1)} = f \quad \text{in} \quad \Omega_2 \\ u_2^{(k+1)} = u_e \quad \text{on} \quad \Gamma_5 \\ \frac{\partial u_2^{(k+1)}}{\partial n} = \frac{\partial u_e}{\partial n} \quad \text{on} \quad \Gamma_6 \quad \text{and} \quad \Gamma_7 \\ \frac{\partial u_2^{(k+1)}}{\partial n} = \frac{\partial u_1^{(k+1)}}{\partial n} \quad \text{on} \quad \Gamma_4 \end{cases} \tag{9}$$

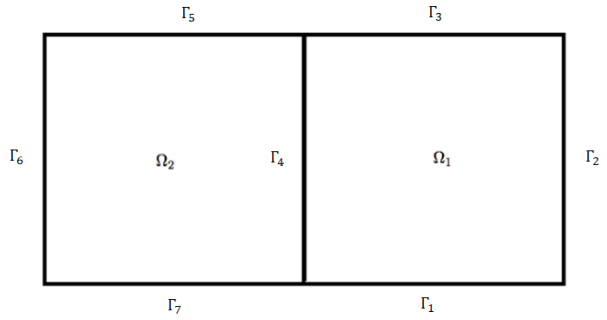


Figure 2: Domain Decomposition

with $\lambda^{(k+1)} = \theta u_2^{(k+1)}|\Gamma_4 + (1-\theta)\lambda^{(k)}$

9.2 The Neumann-Neumann Method

This method is summarized by the following algorithm :[19] for $\lambda^{(0)}$ given, we solve for each $k \geq 0$, (Eqs.(10, 11))

$$\begin{cases} -\Delta u_1^{(k+1)} = f \quad \text{in} \quad \Omega_1 \\ u_1^{(k+1)} = u_e \quad \text{on} \quad \Gamma_3 \\ \frac{\partial u_1^{(k+1)}}{\partial n} = \frac{\partial u_e}{\partial n} \quad \text{on} \quad \Gamma_1 \quad \text{and} \quad \Gamma_2 \\ u_1^{(k+1)} = \lambda^{(k)} \quad \text{on} \quad \Gamma_4 \end{cases} \qquad \begin{cases} -\Delta u_2^{(k+1)} = f \quad \text{in} \quad \Omega_2 \\ u_2^{(k+1)} = u_e \quad \text{on} \quad \Gamma_5 \quad \text{and} \quad \Gamma_6 \\ \frac{\partial u_1^{(k+1)}}{\partial n} = \frac{\partial u_e}{\partial n} \quad \text{on} \quad \Gamma_7 \\ u_2^{(k+1)} = \lambda^{(k)} \quad \text{on} \quad \Gamma_4 \end{cases} \tag{10}$$

$$\begin{cases} -\Delta \psi_1^{(k+1)} = f \quad \text{in} \quad \Omega_1 \\ \psi_1^{(k+1)} = u_e \quad \text{on} \quad \Gamma_3 \\ \frac{\partial \psi_1^{(k+1)}}{\partial n} = \frac{\partial u_e}{\partial n} \quad \text{on} \quad \Gamma_1 \quad \text{and} \quad \Gamma_2 \\ \frac{\partial \psi_1^{(k+1)}}{\partial n} = \frac{\partial u_1^{(k+1)}}{\partial n} + \frac{\partial u_2^{(k+1)}}{\partial n} \quad \text{on} \quad \Gamma_4 \end{cases} \qquad \begin{cases} -\Delta \psi_2^{(k+1)} = f \quad \text{in} \quad \Omega_2 \\ \psi_2^{(k+1)} = u_e \quad \text{on} \quad \Gamma_5 \quad \text{and} \quad \Gamma_6 \\ \frac{\partial \psi_2^{(k+1)}}{\partial n} = \frac{\partial u_e}{\partial n} \quad \text{on} \quad \Gamma_7 \\ \frac{\partial \psi_2^{(k+1)}}{\partial n} = \frac{\partial u_1^{(k+1)}}{\partial n} + \frac{\partial u_2^{(k+1)}}{\partial n} \quad \text{on} \quad \Gamma_4 \end{cases} \tag{11}$$

where $\lambda^{(k+1)}$= $\lambda^(k) - \theta(\sigma_1 u_2^{(k+1)}|4 - \sigma_2 \psi_2^{(k+1)}|4)$ with θ an acceleration parameter. We take in the sequence $\sigma_1 = \sigma_2 = 1$.

9.3 The Robin Method

This method is summarized in the following algorithm :[22] for $u_2^{(0)}$ given, we solve for each $k \geq 0$, (Eq. 12)

$$\begin{cases} -\Delta u_1^{(k+1)} = f \quad \text{in} \quad \Omega_1 \\ u_1^{(k+1)} = u_e \quad \text{in} \quad \Gamma_3 \\ \frac{\partial u_1^{(k+1)}}{\partial n} = \frac{\partial u_e}{\partial n} \quad \text{on} \quad \Gamma_1 \text{and} \quad \Gamma_2 \\ \frac{u_1^{(k+1)}}{\partial n} + \gamma_1 u_1^{(k+1)} = \frac{u_2^{(k)}}{\partial n} + \\ \qquad\qquad \gamma_1 u_2^{(k)} \quad \text{on} \quad \Gamma_4 \end{cases} \qquad \begin{cases} -\Delta u_2^{(k+1)} = f \quad \text{in} \quad \Omega_2 \\ u_2^{(k+1)} = u_e \quad \text{on} \quad \Gamma_5 \quad \text{and} \quad \Gamma_6 \\ \frac{\partial u_2^{(k+1)}}{\partial n} = \frac{\partial u_e}{\partial n} \quad \text{on} \quad \Gamma_7 \\ \frac{u_2^{(k+1)}}{\partial n} + \gamma_2 u_2^{(k+1)} = \frac{u_1^{(k+1)}}{\partial n} + \\ \qquad\qquad \gamma_2 u_1^{(k+1)} \quad \text{on} \quad \Gamma_4 \end{cases} \tag{12}$$

γ_1 and γ_2 are positive acceleration parameters of the problem satisfying $\gamma_1 + \gamma_2 > 0$. We set them to $\gamma_1 = 10$ and $\gamma_2 = 0.1$.

9.4 The Agoshkov-Lebedev Method

This method is a generalization of many other methods, and notably that of Robin seen just above when we set $p_k = \gamma_1$ and $q_k = \frac{1}{\gamma_2}$. α_k and β_k are acceleration parameters (Eq. 13). This method consists in solving, for $u_2^{(0)}$ and $u_1^{(0)}$ given, for each $k \geq 0$, [23]

$$\begin{cases} -\Delta u_1^{(k+\frac{1}{2})} = f \quad \text{in} \quad \Omega_1 \\ u_1^{(k+\frac{1}{2})} = u_e \quad \text{on} \quad \Gamma_3 \\ \frac{\partial u_1^{(k+\frac{1}{2})}}{\partial n} = \frac{\partial u_e}{\partial n} \quad \text{on} \quad \Gamma_1 \quad \text{and} \quad \Gamma_2 \\ \frac{\partial u_1^{(k+\frac{1}{2})}}{\partial n} + p_k u_1^{(k+\frac{1}{2})} = \frac{\partial u_2^{(k)}}{\partial n} + \\ \qquad\qquad p_k u_2^{(k)} \quad \text{on} \quad \Gamma_4 \end{cases} \qquad \begin{cases} -\Delta u_2^{(k+\frac{1}{2})} = f \quad \text{in} \quad \Omega_1 \\ u_2^{(k+\frac{1}{2})} = u_e \quad \text{on} \quad \Gamma_5 \quad \text{and} \quad \Gamma_6 \\ \frac{\partial u_2^{(k+\frac{1}{2})}}{\partial n} = \frac{\partial u_e}{\partial n} \quad \text{on} \quad \Gamma_7 \\ \frac{-q_k \partial u_2^{(k+\frac{1}{2})}}{\partial n} + u_2^{(k+\frac{1}{2})} = -q_k \frac{\partial u_1^{(k+1)}}{\partial n} + \\ \qquad\qquad u_1^{(k+1)} \quad \text{on} \quad \Gamma_4 \end{cases} \tag{13}$$

$$u_1^{(k+1)} = u_1^{(k)} + \alpha_{k+1}(u_1^{(k+\frac{1}{2})} - u_1^{(k)}) \quad \text{in} \quad \Omega_1$$

$$u_2^{(k+1)} = u_2^{(k)} + \beta_{k+1}(u_2^{(k+\frac{1}{2})} - u_2^{(k)}) \quad \text{in} \quad \Omega_2$$

10 Inverse Problem

By "folding" Ω_2 onto Ω_1 in the previous algorithms, we obtain the algorithms to solve the inverse problem. We take the same values, for the parameters, as in the part on domain decomposition.[24] and .[25]

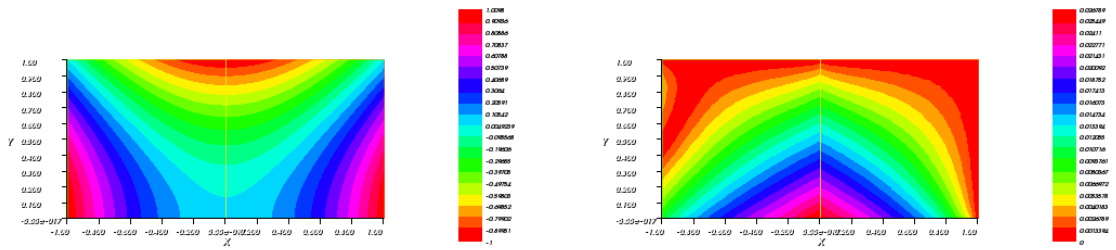
10.1 Dirichlet-Neumann Method

For given $\lambda^{(0)}$, we solve for each $k \geq 0$, (Eq.(14))

$$\begin{cases} -\Delta u^{(2k)} = f \quad \text{in} \quad \Omega_1 \\ u^{(2k)} = u_e \quad \text{on} \quad \Gamma_3 \\ \frac{\partial u^{(2k)}}{\partial n} = \frac{\partial u_e}{\partial n} \quad \text{on} \quad \Gamma_1 \quad \text{in} \quad \Gamma_2 \\ u^{(2k)} = \lambda^{(k)} \quad \text{in} \quad \Gamma_4 \end{cases} \qquad \begin{cases} -\Delta u^{(2k+1)} = f \quad \text{on} \quad \Omega_1 \\ u^{(2k+1)} = u_e \quad \text{in} \quad \Gamma_3 \quad \text{in} \quad \Gamma_2 \\ \frac{\partial u^{(2k+1)}}{\partial n} = \frac{\partial u_e}{\partial n} \quad \text{in} \quad \Gamma_1 \\ \frac{\partial u^{(2k+1)}}{\partial n} = \frac{\partial u^{(2k)}}{\partial n} \quad \text{in} \quad \Gamma_4 \end{cases} \tag{14}$$

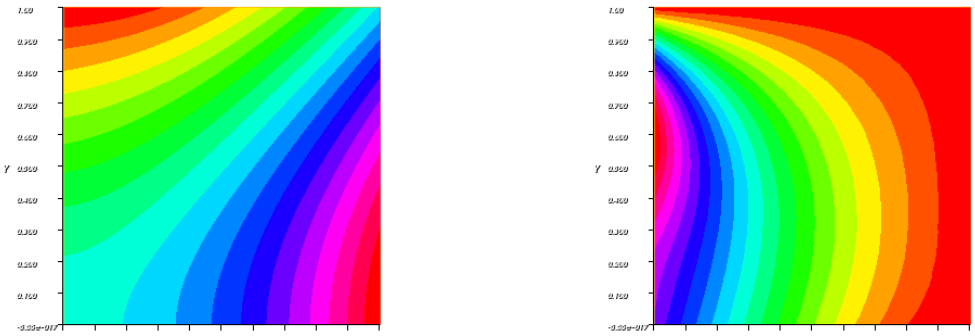
where $\lambda^{(k+1)} = \theta u^{(2k+1)}|4 + (1-\theta)\lambda^{(k)}$ and $\theta > 0$ is a parameter of acceleration.
(see Fig. 3,4,5)

It should be noted that this technique is not necessarily suitable, as long as one does not make assumptions about the parameter θ or Ω_1 and Ω_2. However, if it is convergent, the rate of convergence does not depend on the weight of the mesh. See[19] for a proof of convergence.



(a) Approximate solution for Dirichlet-Neumann with domain decomposition-Direct problem (b) Error for Dirichlet-Neumann with domain decomposition-Direct problem

Figure 3: Dirichelt- Numann Condition For Direct Problem



(a) Approximate solution for Dirichlet-Neumann with domain decomposition-Inverse problem (b) Error for Dirichlet-Neumann with domain decomposition-Inverse problem

Figure 4: Dirichelt- Numann Condition For Inverse Problem

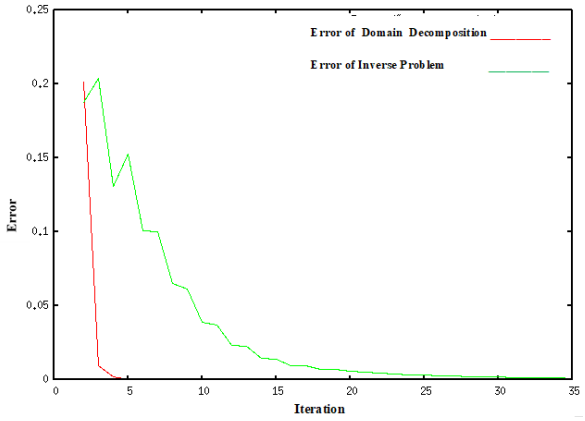


Figure 5: Error in the Dirichlet-Neumann case as a function of the iteration number.

10.2 Neumann-Neumann Method

According to Bourgat et al.,[19] the Neumann-Neumann method operates as follows (Eqs.(14, 15)): Set $\lambda^{(0)}$ to its initial value and solve for each $k \geq 0$ (see Fig. 6,7,8)

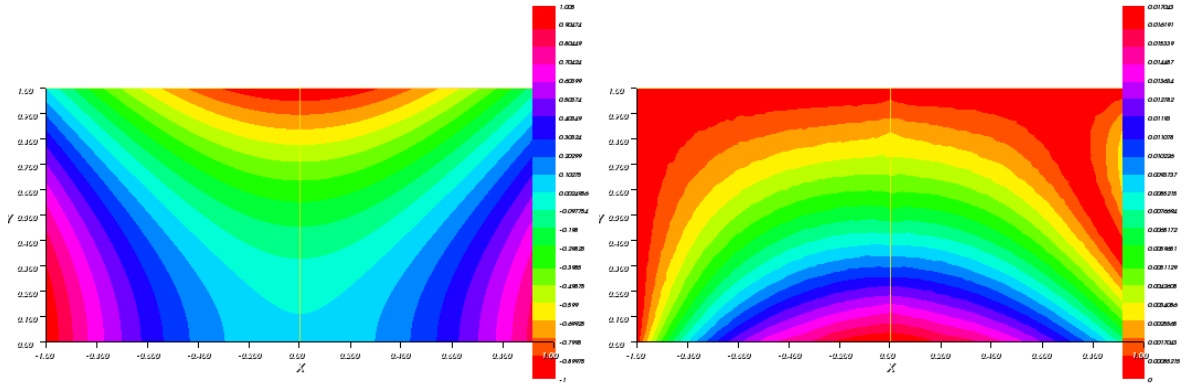
$$\begin{cases} -\Delta u^{(2k)} = f \quad \text{on} \quad \Omega_1 \\ u^{(2k)} = u_e \quad \text{in} \quad \Gamma_3 \\ \frac{\partial u^{(2k)}}{\partial n} = \frac{\partial u_e}{\partial n} \quad \text{in} \quad \Gamma_1 \quad \text{and} \quad \Gamma_2 \\ u^{(2k)} = \lambda^{(k)} \quad \text{in} \quad \Gamma_4 \end{cases} \qquad \begin{cases} -\Delta u^{(2k+1)} = f \quad \text{on} \quad \Omega_1 \\ u^{(2k+1)} = u_e \quad \text{in} \quad \Gamma_2 \quad \text{and} \quad \Gamma_3 \\ \frac{\partial u^{(2k+1)}}{\partial n} = \frac{\partial u_e}{\partial n} \quad \text{in} \quad \Gamma_1 \\ u^{(2k+1)} = \lambda^{(k)} \quad \text{in} \quad \Gamma_4 \end{cases} \tag{15}$$

$$\begin{cases} -\Delta \psi^{(2k)} = f \quad \text{on} \quad \Omega_1 \\ \psi^{(2k)} = u_e \quad \text{in} \quad \Gamma_3 \\ \frac{\partial \psi^{(2k)}}{\partial n} = \frac{\partial u_e}{\partial n} \quad \text{in} \quad \Gamma_1 \quad \text{and} \quad \Gamma_2 \\ \frac{\partial \psi^{(2k)}}{\partial n} = \frac{\partial u^{(2k)}}{\partial n} - \frac{\partial u^{(2k+1)}}{\partial n} \quad \text{in} \quad \Gamma_4 \end{cases} \qquad \begin{cases} -\Delta \psi^{(2k+1)} = f \quad \text{on} \quad \Omega_1 \\ \psi^{(2k+1)} = u_e \quad \text{in} \quad \Gamma_2 \quad \text{and} \quad \Gamma_3 \\ \frac{\partial \psi^{(2k+1)}}{\partial n} = \frac{\partial u_e}{\partial n} \quad \text{in} \quad \Gamma_1 \\ \frac{\partial \psi^{(2k+1)}}{\partial n} = \frac{\partial u^{(2k)}}{\partial n} - \frac{\partial u^{(2k+1)}}{\partial n} \quad \text{in} \quad \Gamma_4 \end{cases} \tag{16}$$

Considering $i = 1, 2$, with

$$\lambda^{n+1} := \lambda^n - \theta(\sigma_1 \psi_1^{n+1}|_\Gamma - \sigma_2 \psi_2^{n+1}|_\Gamma)$$

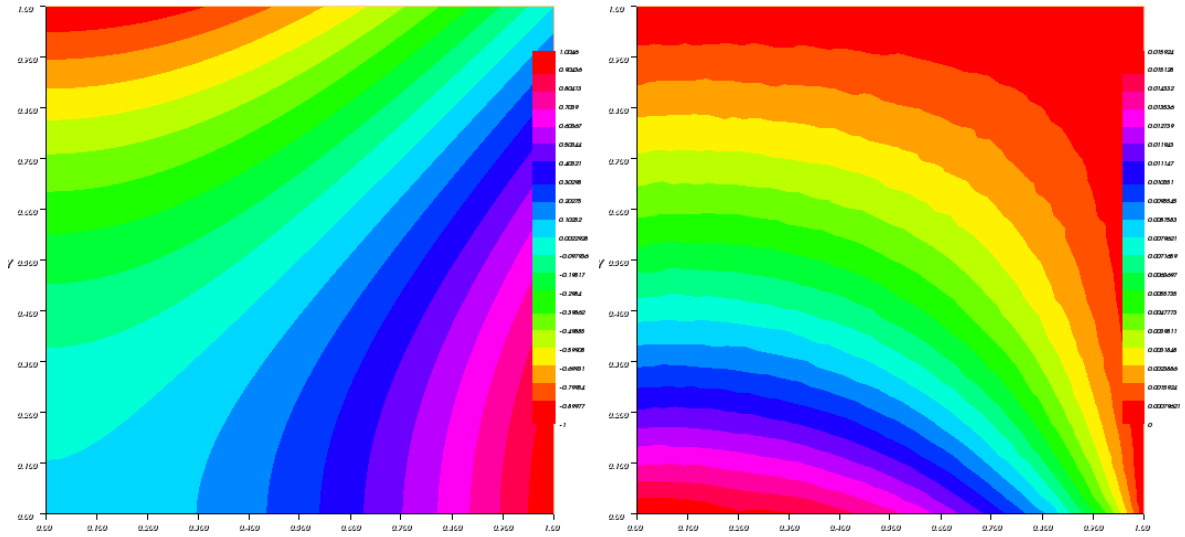
, in which $\sigma_1 > 0$ are averaging coefficients and $\theta > 0$ is an acceleration parameter In addition to being guaranteed to converge, the rate of convergence in the case of a discretization with finite elements is independent of the mesh size.



(a) Approximate solution for Neumann-Neumann with domain decomposition-Direct Problem

(b) Error for Neumann-Neumann with domain decomposition-Direct Problem

Figure 6: Numann-Numann Condition For Direct Problem



(a) Approximate solution for Neumann-Neumann with domain decomposition-Inverse problem

(b) Error for Neumann-Neumann with domain decomposition-Inverse problem

Figure 7: Numann-Numann Condition For Inverse Problem

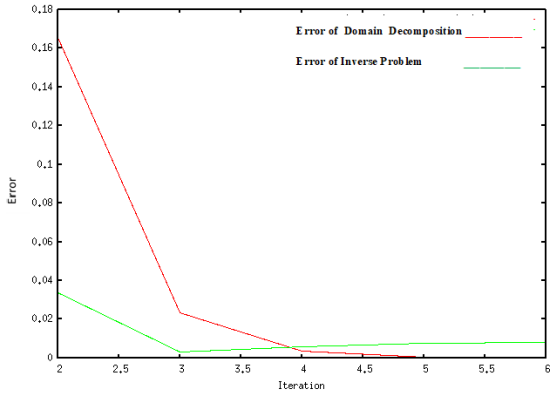


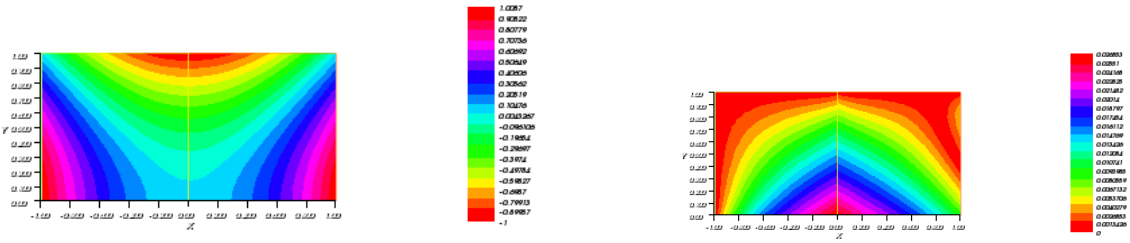
Figure 8: Error in the Neumann-Neumann case as a function of the iteration number.

10.3 Robin Method

For given $u^{(1)}$, we solve for each $k \geq 1$, (Eq.(17))

$$\begin{cases} -\Delta u^{(2k)} = f \quad \text{on} \quad \Omega_1 \\ u^{(2k)} = u_e \quad \text{in} \quad \Gamma_3 \\ \frac{\partial u^{(2k)}}{\partial n} = \frac{\partial u_e}{\partial n} \quad \text{in} \quad \Gamma_1 \quad \text{and} \quad \Gamma_2 \\ \frac{u^{(2k)}}{\partial n} + \gamma_1 u^{(2k)} = \frac{u^{(2k-1)}}{\partial n} + \\ \qquad \gamma_1 u^{(2k-1)} \quad \text{in} \quad \Gamma_4 \end{cases} \qquad \begin{cases} -\Delta u^{(2k+1)} = f \quad \text{on} \quad \Omega_1 \\ u^{(2k+1)} = u_e \quad \text{in} \quad \Gamma_2 \quad \text{and} \quad \Gamma_3 \\ \frac{\partial u^{(2k+1)}}{\partial n} = \frac{\partial u_e}{\partial n} \quad \text{in} \quad \Gamma_1 \\ \frac{u^{(2k+1)}}{\partial n} - \gamma_2 u^{(2k+1)} = \frac{u^{(2k)}}{\partial n} + \\ \qquad \gamma_2 u^{(2k)} \quad \text{in} \quad \Gamma_4 \end{cases} \tag{17}$$

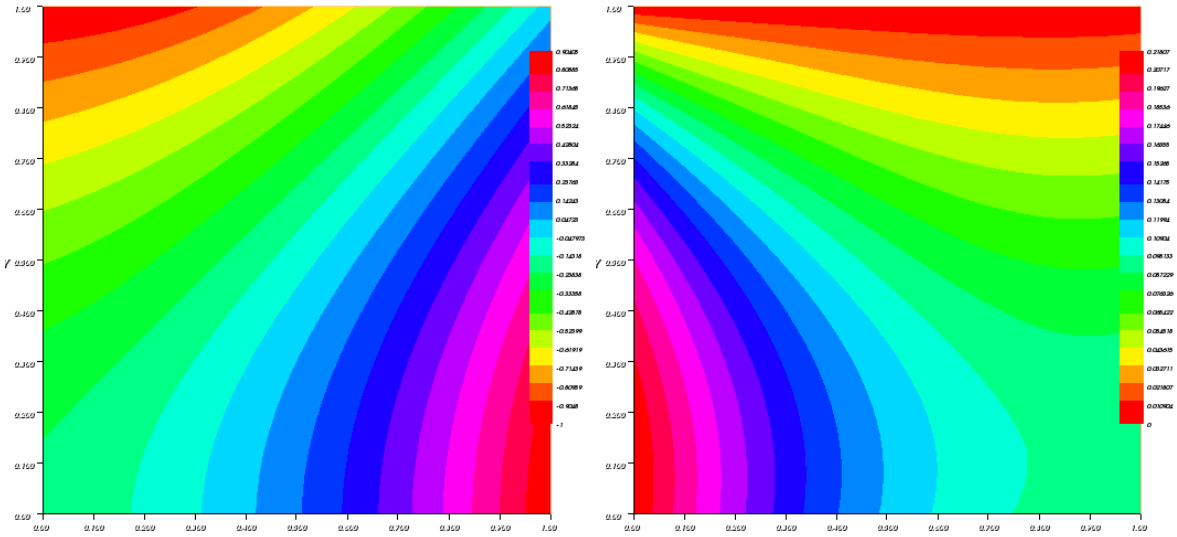
where the acceleration parameters γ_1 and γ_2 are non-negative and satisfy the condition $\gamma_1 + \gamma_2 > 1$ (see Fig. 9,10,11)



(a) Approximate solution for Robin with domain decomposition-Direct problem (b) Error for Robin with domain decomposition

Figure 9: Robin Condition For Direct Problem

We notice that the error at each iteration for domain decompositions has a good appearance. This is not the case for the inverse problem: we notice in fact a ”distance” from the solution calculated in the case of Robin and Neumann-Neumann from a certain rank. Although we don’t know the rate of convergence or estimates of the error reduction factor at each iteration, the Robin method is guaranteed to converge.[22]



(a) Approximate solution for Robin with domain decomposition-Inverse problem

(b) Error for Robin for the inverse problem

Figure 10: Robin Condition For Inverse Problem

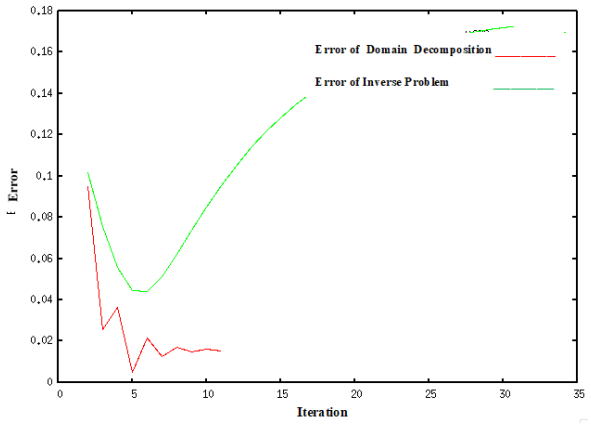


Figure 11: Error in Robin case as a function of the iteration.

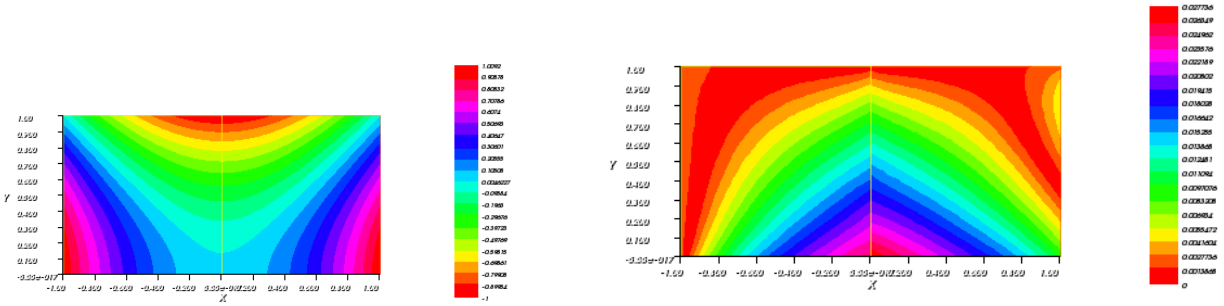
10.4 Agoshkov-Lebedev Algorithm

Agoshkov and Lebedev proposed the non-overlapping domain decomposition algorithm by.[?] This method consists in solving, for $u^{(0)}$ and $u^{(1)}$ given, for each $k \geq 1$, Eq.(18), (see Fig. 12,13,14)

$$\begin{cases} -\Delta u^{(2k+\frac{1}{2})} = f \quad \text{on} \quad \Omega_1 \\ u^{(2k+\frac{1}{2})} = u_e \quad \text{in} \quad \Gamma_3 \\ \dfrac{\partial u^{(2k+\frac{1}{2})}}{\partial n} = \dfrac{\partial u_e}{\partial n} \quad \text{in} \quad \Gamma_1 \quad \text{and} \quad \Gamma_2 \\ \dfrac{\partial u^{(2k+\frac{1}{2})}}{\partial n} + p_{2k} u^{(2k+\frac{1}{2})} = \dfrac{\partial u^{(2k-1)}}{\partial n} + \\ \qquad\qquad p_{2k} u^{(2k-1)} \quad \text{in} \quad \Gamma_4 \end{cases} \qquad \begin{cases} -\Delta u^{(2k+\frac{3}{2})} = f \quad \text{on} \quad \Omega_1 \\ u^{(2k+\frac{3}{2})} = u_e \quad \text{in} \quad \Gamma_3 \quad \text{and} \quad \Gamma_2 \\ \dfrac{\partial u^{(2k+\frac{3}{2})}}{\partial n} = \dfrac{\partial u_e}{\partial n} \quad \text{in} \quad \Gamma_1 \\ \dfrac{-q_{2k+1} \partial u^{(2k+\frac{3}{2})}}{\partial n} + u^{(2k+\frac{3}{2})} = -q_{2k+1} \dfrac{\partial u^{(2k)}}{\partial n} + \\ \qquad\qquad u^{(2k)} \quad \text{in} \quad \Gamma_4 \end{cases}$$

$$u^{(2k)} = u^{(2k-2)} + \alpha_{2k}(u^{(2k+\frac{1}{2})} - u^{(2k-2)}), \qquad u^{(2k+1)} = u^{(2k-1)} + \beta_{2k+1}(u^{(2k+\frac{3}{2})} - u^{(2k-1)}) \tag{18}$$

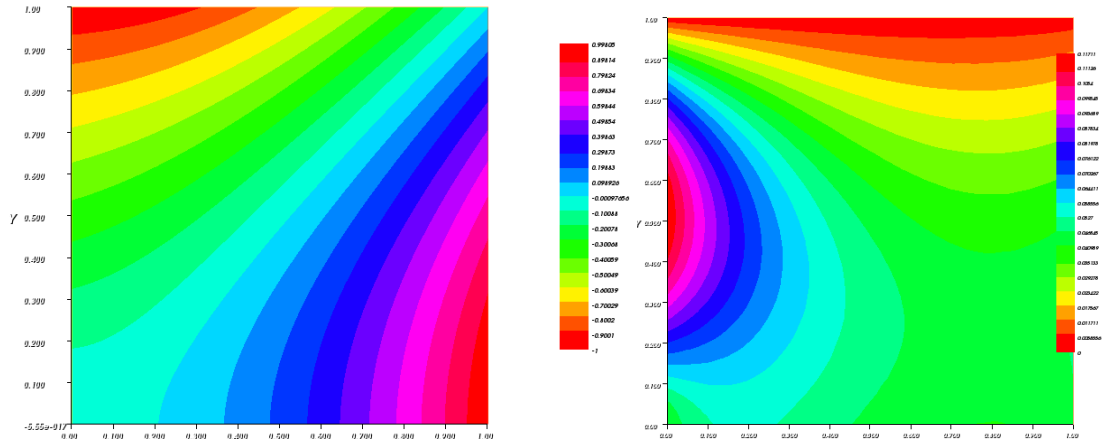
where $p_n, q_n \geq 0$ and $\alpha_{n+1}, \beta_{n+1} \in \mathrm{R}$ are free parameters. This algorithm is a generalization of many other methods, as the already mentioned Robin method Eq.(17), which is obtained by setting $p_n = \gamma_1, q_n = 1/\gamma_2$ and $\alpha_n = \beta_n = 1$ in Eq.(18).
Similarly,one can also obtain the Dirichlet-Neumann method Eq.(14) by taking $p_n = q_n = 0$, $\alpha_n = \beta_n = 1$, and by noting that the roles of Ω_1 and Ω_2 are reversed.



(a) Approximate solution for Agoshkov-Lebedev -Direct problem

(b) Error for Agoshkov-Lebedev -Direct problem

Figure 12: Agoshkov-Lebedev For Direct Problem



(a) Approximate solution for Agoshkov-Lebedev -Inverse problem

(b) Error for Agoshkov-Lebedev -Inverse problem

Figure 13: Agoshkov-Lebedev For Inverse Problem

11 Numerical Results

In this part, we take $u_e(x, y) = x^2 - y^2$. We therefore set $f = 0$. The initializations ($u^{(0)}$ or $\lambda^{(0)}$ for example) are all placed at $-y$. First, we present the approximate solution and then the error for all cases Then, we carry out the study of the error (norm $L^2(\Gamma_4)$ of the difference between the calculated solution and u_e) as a function of the iteration number for

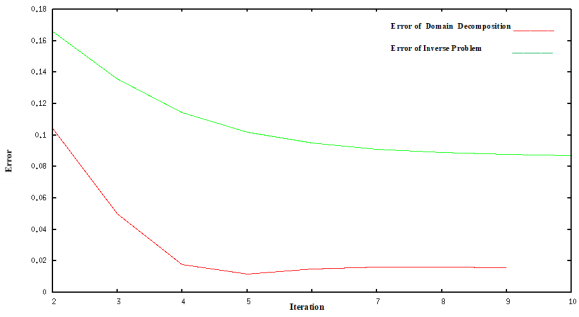
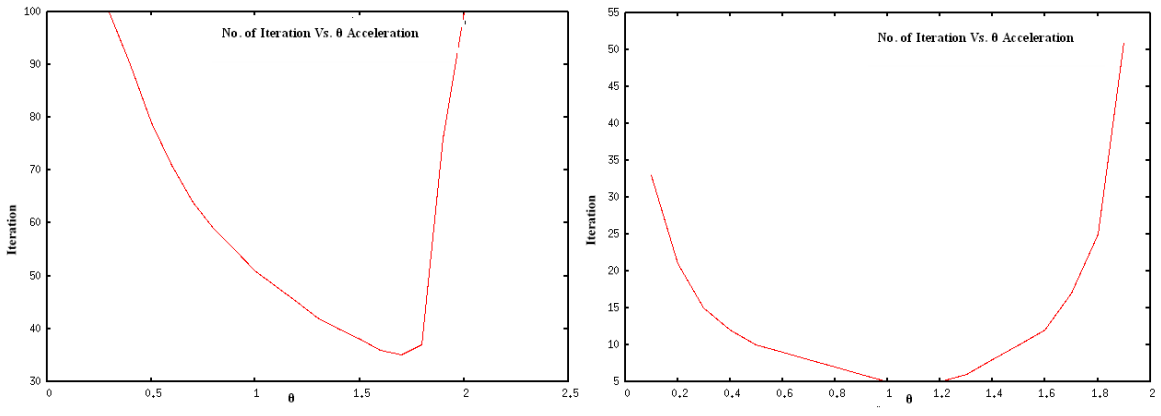


Figure 14: Error in the Agoshkov-Lebedev case as a function of the iteration number.

all the algorithms (domain decomposition included).
Finally we make an approximation of the optimal acceleration parameter θ in the case of the inverse problem.

11.1 Calculation of the optimal parameter

To solve the ICP optimally, it is crucial to select appropriate approximation parameters and optimization strategies. These parameters guide the iterative optimization process to minimize the discrepancy between observed data and model predictions. In this section, we present the number of iterations required as a function of the acceleration parameter θ .[27] The Agoshkov-Lebedev algorithm is a generalization of the latter p_k and q_k deduced from γ_1 and γ_2 with two acceleration parameters α_k and β_k.
We want to know the acceleration obtained with respect to Robin by varying these parameters, so we do not perform this test for Robin. Furthermore, to simplify, we restrict ourselves to the case $\alpha_k = \beta_k = \theta$. se Fig. (15,16)



(a) Iterations Vers. θ for Dirichlet-Neumann (b) Iterations Vers. θ for Neumann-Neumann

Figure 15: Iterations Vers. θ

Therefore we deduce that $\theta_{optimal} \simeq 1.7$ for the Dirichlet-Neumann algorithm, $\theta_{optimal} \simeq 1.15$ for the Neumann-Neumann algorithm and $\theta_{optimal} \simeq 0.5$ for the Agoshkov-Lebedev algorithm. We note that, in the Agoshkov-Lebedev case, another local minimum appears unlike in the other cases. Finally, when $\alpha_k = \beta_k = \theta = 0.5$ in this case, we have an answer in 9 iterations. While Robin returns an answer in 31 iterations. We therefore have an acceleration of convergence.

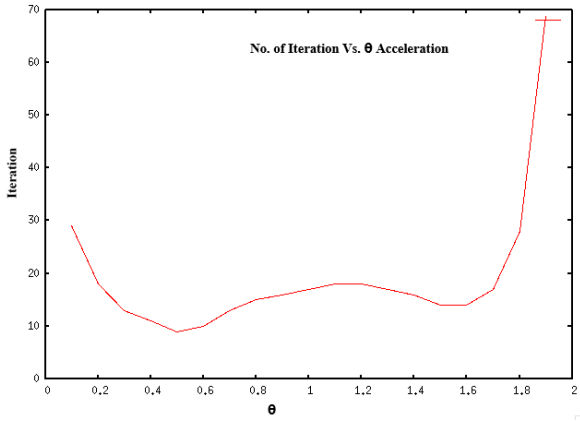


Figure 16: Iterations Vers. θ for Agoshkov-Lebedev .

12 Discussion

In this study, we studied four iterative methods for solving inverse problems of Burger equation by using domain decomposition method. This method are based on four classical boundaries Dirichlet-Neumann, Neumann-Neumann, Robin and Agoshkov-Lebedev approaches. The aim was to analyze their efficiency in terms of convergence and to compare the numerical results obtained.

- Dirichlet-Neumann method: The Dirichlet-Neumann method has shown good performance in terms of convergence for domain decomposition problems. The approximate solution converges quickly to the exact solution, with an error that decreases at each iteration. However, for the inverse problem, the error tends to increase after a certain number of iterations, suggesting that this method might be less efficient for inverse problems, especially when the boundary conditions are complex.

- Neumann-Neumann method: Similar to the Dirichlet-Neumann, the Neumann-Neumann method performs well for domain decomposition problems, with a reduction in error over iterations. However, for inverse problems, the error also shows a tendency to grow from a certain point. This phenomenon is particularly marked for methods that use Neumann-type conditions, where the error propagation may be more sensitive to changes in boundary conditions.

- Robin method: The Robin method, although effective for some inverse problems, has a slower convergence rate compared to other methods. The study of errors has shown that, in the case of the inverse problem, the error tends to stabilize after a certain number of iterations, but this stabilization is slower than for other methods. The acceleration parameters $y1$ and $y2$ have a notable influence on the convergence, but even with these optimized parameters, the number of iterations remains relatively high, which can make the method less competitive for complex inverse problems.

- Agoshkov-Lebedev method: The Agoshkov-Lebedev method, which is a generalization of Robin's method, has shown very good performance, especially with the acceleration parameters α_k and β_k. This method has made it possible to significantly reduce the number of iterations required to obtain a satisfactory solution, especially by using

optimal values to these parameters. In particular, in the case where $\alpha_k = \beta_k = 0.5$, the method converged in only 9 iterations, compared to 31 iterations for Robin's method. This shows the increased efficiency of the Agoshkov-Lebedev algorithm compared to others, especially for inverse problems.

13 Conclusion

In conclusion, we can state that the tested domain decomposition methods for solving Burger equation, in particular Dirichlet-Neumann and Neumann-Neumann, are efficient for solving standard diffusion of Burger problems. However, for inverse problems, the Agoshkov-Lebedev method stands out for its ability to accelerate convergence, in particular thanks to the use of optimal acceleration parameters. While Robin's method is competitive for domain decomposition problems, it remains less efficient for inverse problems due to its slower convergence rate.

References

[1] Bateman, H. (1915). Some recent researches on the motion of fluids. Monthly Weather Review, 43(4), 163-170.

[2] Velasco, R. M., and Saavedra, P. (2007). A first order model in traffic flow. Physica D: Nonlinear Phenomena, 228(2), 153-158.

[3] Auad, A. A., & AbdulJabbar, R. S. (2018, May). On direct theorems for best polynomial approximation. In Journal of Physics: Conference Series (Vol. 1003, No. 1, p. 012054). IOP Publishing.

[4] Watanabe, S., Ishiwata, S., Kawamura, K., and Oh, H. G. (1997). Higher order solution of nonlinear waves. II. Shock wave described by Burgers equation. Journal of the Physical Society of Japan, 66(4), 984-987.

[5] Albeverio, S., Korshunova, A., and Rozanova, O. (2013). A probabilistic model associated with the pressureless gas dynamics. Bulletin des sciences mathematiques, 137(7), 902-922.

[6] Srivastava, V. K., Awasthi, M. K., and Tamsir, M. (2013). A fully implicit Finite-difference solution to one dimensional Coupled Nonlinear Burgers' equations. Int. J. Math. Sci, 7(4), 23.

[7] Cole, J. D. (1951). On a quasi-linear parabolic equation occurring in aerodynamics. Quarterly of applied mathematics, 9(3), 225-236.

[8] Fletcher, C. A. J. (1983). A comparison of finite element and finite difference solutions of the one-and two-dimensional Burgers' equations. Journal of Computational Physics, 51(1), 159-188.

[9] Wazwaz, A. M. (2002). Partial differential equations. CRC Press.

[10] Hilal, M. A. (2016). Domain decomposition like methods for solving an electrocardiography inverse problem (Doctoral dissertation, Nantes).

[11] Nachaoui, A., Nachaoui, M., Chakib, A., and Hilal, M. A. (2021). Some novel numerical techniques for an inverse Cauchy problem. Journal of Computational and Applied Mathematics, 381, 113030.

[12] J. Hadamard. Lectures on Cauchy's problem in linear partial differential equations. Dover Publications, New York, 1953.

[13] Zayeni, H., Abda, A. B., Delvare, F., and Khayat, F. Fading regularization MFS algorithm for the Cauchy problem associated with the two-dimensional Stokes equations.

[14] Abbasbandy, S., Jafarian, A., and Ezzati, R. (2005). Conjugate gradient method for fuzzy symmetric positive definite system of linear equations. Applied mathematics and computation, 171(2), 1184-1191.

[15] Saad, Y., and Schultz, M. H. (1986). GMRES: A generalized minimal residual algorithm for solving nonsymmetric linear systems. SIAM Journal on scientific and statistical computing, 7(3), 856-869.

[16] J. N. Ismail et al. Enhancing Decision Accuracy in DEMATEL using Bonferroni Mean Aggregation under Pythagorean Neutrosophic Environment. Journal of Fuzzy Extension & Applications (JFEA), 4(4), 281 - 298, 2023.

[17] Toh, K. C., and Phoon, K. K. (2008). Comparison between iterative solution of symmetric and non-symmetric forms of Biot's FEM equations using the generalized Jacobi preconditioner. International journal for numerical and analytical methods in geomechanics, 32(9), 1131-1146.

[18] C. Börgers. The Neumann–Dirichlet domain decomposition method with inexact solvers on the subdomains. Numer. Math., 55:123–136, 1989.

[19] J.-F. Bourgat, Roland Glowinski, Patrick Le Tallec, and Marina Vidrascu. Variational formulation and algorithm for trace operator in domain decomposition calculations. In Domain decomposition methods (Los Angeles, CA, 1988), pages 3–16. SIAM, Philadelphia, PA, 1989.

[20] L.D. Marini and A. Quarteroni. A relaxation procedure for domain decomposition methods using finite elements. Numerische Mathematik, 55(5):575–598, 1989.

[21] A. Al-Quran, F. Al-Sharqi, A. U. Rahman and Z. M. Rodzi, The q-rung orthopair fuzzy-valued neutrosophic sets: Axiomatic properties, aggregation operators and applications. AIMS Mathematics, 9(2), 5038-5070, 2024.

[22] P. L. Lions. On the Schwarz alternating method. III. A variant for nonoverlapping subdomains. In Third International Symposium on Domain Decomposition Methods for Partial Differential Equations (Houston, TX, 1989), pages 202–223. SIAM, Philadelphia, PA, 1990.

[23] V. I. Agoshkov and V. I. Lebedev. Poincaré-Steklov operators and methods of partition of the domain in variational problems. In Computational processes and systems, No. 2, pages 173–227. "Nauka", Moscow, 1985.

[24] V. Dolean, F. Nataf, and G. Rapin. New constructions of domain decomposition methods for systems of pdes. Comptes Rendus Mathematique, 340(9):693– 696, 2005.

[25] V. Dolean, F. Nataf, and G. Rapin. Deriving a new domain decomposition method for the Stokes equations using the Smith factorization. Mathematics of Computation, 78:789–814, 2009.

[26] F. Al-Sharqi, A. Al-Quran and Z. M. Rodzi, Multi-Attribute Group Decision-Making Based on Aggregation Operator and Score Function of Bipolar Neutrosophic Hypersoft Environment, Neutrosophic Sets and Systems, 61(1), 465-492, 2023.

[27] M. Jourhmane and A. Nachaoui. An alternating method for an inverse Cauchy problem. Numer. Algorithms, 21(1-4):247–260, 1999. Numerical methods for partial differential equations (Marrakech, 1998).

[28] M. U. Romdhini et. al., Signless Laplacian Energyof Interval-Valued Fuzzy Graph and its Applications, Sains Malaysiana 52(7), 2127-2137, 2023