



Universidad Nacional de Asunción  
Facultad Politécnica

# A Comparison of Preconditioned Iterative Solvers for High-Order Mimetic Difference Approximations to the Poisson Equation

Gustavo Enrique Espínola Mena

Tesis presentada a la Facultad Politécnica, Universidad Nacional de Asunción, como requisito para la obtención del Grado de Master en Ciencias de la Computación.

San Lorenzo - Paraguay  
Octubre, 2025



Universidad Nacional de Asunción

Facultad Politécnica

# **A Comparison of Preconditioned Iterative Solvers for High-Order Mimetic Difference Approximations to the Poisson Equation**

**Gustavo Enrique Espínola Mena**

Orientadores:

Prof. Juan C. Cabral, D.Sc

Prof. Jhabriel D. Varela, Ph.D.

Tesis presentada a la Facultad Politécnica, Universidad Nacional de Asunción, como requisito para la obtención del Grado de Master en Ciencias de la Computación.

San Lorenzo - Paraguay

Octubre, 2025

Gustavo Enrique Espínola Mena.

A Comparison of Preconditioned Iterative Solvers for High-Order Mimetic Difference Approximations to the Poisson Equation/

Gustavo Enrique Espínola Mena. Orientadores: Juan Carlos Cabral, DSc, Jhabriel Varela, PhD. - - San Lorenzo : FPUNA, 2025.

X-XXXI; 92p; il.:30cm.

Incluye bibliografía y apéndices (CD)

Trabajo de Maestría (Ciencias de la Computación). – UNA. Facultad Politécnica, 2025.

1. High-order mimetic operators. 2. Preconditioned Krylov subspace methods. 3. Sparse linear systems. I. Cabral. Juan C. II Jhabriel D. Varela. III. Título.

SCDD 004

Es77c

Hoja de Aprobación de Tesis

**A COMPARISON OF PRECONDITIONED ITERATIVE  
SOLVERS FOR HIGH-ORDER MIMETIC DIFFERENCE  
APPROXIMATIONS TO THE POISSON EQUATION**

**Gustavo Enrique Espínola Mena**

Tesis de Maestría aprobada el 27 de octubre de 2025 por los siguientes miembros del  
Jurado de Defensa:

Prof. Dr. Miguel Dumett (San Diego State University, EEUU)

Prof. Dr. Inocencio Ortiz (Polytechnic University Taiwan-Paraguay)

Prof. Dr. Christian Schaerer (FP-UNA)

Prof. Dr. Juan Carlos Cabral (FP-UNA), orientador

Prof. Dr. Jhabriel Varela (FP-UNA), co-orientador

**Prof. Dr. José Colbes Sanabria**

Coordinador Académico

Maestría en Ciencias de la Computación

Facultad Politécnica

Universidad Nacional de Asunción

**Prof. Dr. Juan Carlos Cabral**

Orientador

*To my family and friends, for the emotional support.  
To Mauricio, for the valuable conversations in Ap. 511.  
To Marina, to make my world brighter.*

*Real knowledge is to know the extent of one's ignorance.*  
Confucius (551 - 479 BCE)

*The beauty of Mathematics only shows itself to more patient followers.*  
Maryam Mirzakhani (1977 - 2017)

## **Acknowledgements**

To the people of the Republic of Paraguay, through the Fund for Excellence in Education and Investigation (FEEI), for their efforts in investing in programs and projects that promote educational excellence and the promotion of research for development.

To the National Council for Science and Technology (CONACYT), its PRO-CIENCIA II Program, and the Incentives for the Training of Researchers in National Postgraduates sub-program.

To the Polytechnic School at the National University of Asunción (FPUNA) for its valuable academic offerings to the society.

To the Nucleus for Investigation and Technologic Development (NIDTEC) for the freedom and space for creativity to thrive.

To Professors Juan C. Cabral, Jhabriel D. Varela, and Christian E. Schaerer, my eternal gratitude for their teachings and comments during this work.

Gustavo.

# **A Comparison of Preconditioned Iterative Solvers for High-Order Mimetic Difference Approximations to the Poisson Equation**

*Autor:* Gustavo Enrique Espínola Mena

*Orientadores:*

Prof. Juan C. Cabral, D.Sc.

Prof. Jhabriel D. Varela, Ph.D.

## **Abstract**

Mimetic difference approximations are of increasing interest in scientific computing for their ability to preserve key properties of continuous problems, such as conservation laws, while maintaining a consistent order of accuracy in the interior of the domain and near the boundary. However, their application leads to sparse, locally dense, and potentially ill-conditioned linear systems that are challenging to solve. Using the two-dimensional Poisson equation as the model problem, this thesis evaluates the computational performance of several preconditioned Krylov subspace methods for solving the resulting linear systems. Numerical experiments show that combining a Jacobi preconditioner with augmented GMRES( $m$ ) variants significantly reduces the execution times and number of iterations. The results highlight the potential of augmented iterative methods with preconditioning as a robust alternative for large-scale simulations in science and engineering that use mimetic discretizations.

**Keywords:** high-order mimetic operators, preconditioned Krylov subspace methods, sparse linear systems.

# Table of Contents

|                                                            |             |
|------------------------------------------------------------|-------------|
| <b>List of Tables</b>                                      | <b>vi</b>   |
| <b>List of Figures</b>                                     | <b>viii</b> |
| <b>1 Introduction</b>                                      | <b>1</b>    |
| 1.1 Objectives . . . . .                                   | 2           |
| 1.2 Originality and Contribution . . . . .                 | 2           |
| 1.3 Organization of the Book . . . . .                     | 3           |
| 1.4 Notation . . . . .                                     | 3           |
| <b>2 Mimetic Discretization Methods</b>                    | <b>5</b>    |
| 2.1 Poisson Equation . . . . .                             | 7           |
| 2.2 Finite Difference Methods . . . . .                    | 7           |
| 2.2.1 1D Approximation of Poisson's Equation . . . . .     | 9           |
| 2.2.2 1D Staggered grid . . . . .                          | 10          |
| 2.2.3 Support Operator Method . . . . .                    | 11          |
| 2.3 Mimetic Operators . . . . .                            | 15          |
| 2.3.1 Building Mimetic Operators . . . . .                 | 17          |
| 2.3.2 2D Staggered Grid . . . . .                          | 21          |
| 2.3.3 2D Mimetic Operators . . . . .                       | 23          |
| 2.4 MOLE: Mimetic Operators Library Enhanced . . . . .     | 24          |
| <b>3 Krylov Methods for</b>                                |             |
| <b>Nonsymmetric Linear Systems</b>                         | <b>27</b>   |
| 3.1 Iterative Methods for Solving Linear Systems . . . . . | 28          |
| 3.2 Control Approach to Iterative Methods . . . . .        | 29          |
| 3.3 Basic Iterative Methods . . . . .                      | 30          |
| 3.3.1 Jacobi Iteration Method . . . . .                    | 31          |
| 3.3.2 Gauss-Seidel Iterative Method . . . . .              | 31          |

|          |                                                               |           |
|----------|---------------------------------------------------------------|-----------|
| 3.4      | Krylov Subspace Methods . . . . .                             | 31        |
| 3.5      | GMRES Method . . . . .                                        | 32        |
| 3.6      | Restarting Version of GMRES, or GMRES( $m$ ) . . . . .        | 34        |
| 3.7      | Strategies for GMRES( $m$ ) . . . . .                         | 35        |
| 3.7.1    | LGMRES( $m, l$ ) . . . . .                                    | 36        |
| 3.7.2    | GMRES-E( $m, d$ ) . . . . .                                   | 37        |
| 3.7.3    | PD-GMRES( $m_j$ ) . . . . .                                   | 39        |
| 3.8      | The KrySBAS Library . . . . .                                 | 39        |
| 3.9      | Preconditioning . . . . .                                     | 42        |
| 3.9.1    | Iteration Matrices and Preconditioning . . . . .              | 42        |
| 3.9.2    | Left Preconditioning . . . . .                                | 43        |
| 3.9.3    | Incomplete $LU$ Preconditioning . . . . .                     | 44        |
| <b>4</b> | <b>Experiments, Results and Discussion</b>                    | <b>45</b> |
| 4.1      | Experimental Setup . . . . .                                  | 45        |
| 4.2      | Problems . . . . .                                            | 46        |
| 4.3      | Numerical Convergence Analysis . . . . .                      | 49        |
| 4.4      | Comparative Analysis of Iterative Methods . . . . .           | 50        |
| 4.5      | Effectiveness in Preconditioning . . . . .                    | 54        |
| 4.6      | Adapting the Restart Parameter $m$ . . . . .                  | 58        |
| <b>5</b> | <b>Conclusions and Future Work</b>                            | <b>60</b> |
|          | <b>References</b>                                             | <b>62</b> |
| <b>A</b> | <b>Notes on Numerical Analysis</b>                            | <b>67</b> |
| A.1      | Numerical Error Estimation . . . . .                          | 67        |
| A.2      | Eigenvalues, Eigenvectors and Spectral Radius . . . . .       | 68        |
| A.3      | Norms of Vectors and Matrices . . . . .                       | 69        |
| A.3.1    | Norm of a Vector . . . . .                                    | 69        |
| A.3.2    | Norm of a Matrix . . . . .                                    | 69        |
| A.4      | Perturbation Analysis: Condition Number of a Matrix . . . . . | 70        |
| <b>B</b> | <b>Presentation in National and International Conferences</b> | <b>73</b> |
| B.1      | Presentations in International Conferences . . . . .          | 73        |
| B.2      | Presentations in National Confereneces . . . . .              | 76        |

# List of Tables

|     |                                                                                                                                                                                                                                                                                                                                                                              |    |
|-----|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|----|
| 4.1 | Parameters considered for each method. . . . .                                                                                                                                                                                                                                                                                                                               | 46 |
| 4.2 | List of test problems in the Poisson problem with different orders of accuracy ( $k$ ), discretization sizes ( $m_x = m_y$ in the studied case) and classic preconditioners. %nnz( $L$ ): percentage of nonzero elements of matrix $L$ , condest( $L$ ): condition number of matrix $L$ , OOM: "Out of Memory" exception. > 1 h: the preconditioning time exceeded 1 hour. . | 49 |
| 4.3 | Numerical convergence analysis for Problem A. The error $E_\infty$ is measured in the infinity norm, and the estimated order of convergence is calculated using Equation (4.2) for successive grid refinements. . . . .                                                                                                                                                      | 51 |
| 4.4 | Metrics for iterative methods, Problem A: average computational time in seconds and number of iterations required for $\ \mathbf{r}_j\ /\ \mathbf{r}_0\  < 10^{-12}$ . For cases that do not reach convergence before 1000 restarts, the method is stopped and the time is denoted as NC. Lowest execution times and number of iterations are highlighted in green. . . . .  | 53 |
| 4.5 | Metrics for iterative methods, Problem B: average computational time in seconds and number of iterations required for $\ \mathbf{r}_j\ /\ \mathbf{r}_0\  < 10^{-12}$ . Lowest execution times and number of iterations are highlighted in green.                                                                                                                             | 54 |
| 4.6 | Comparative summary of numerical methods with different preconditioners. Order of accuracy: $k = 2$ . Average computational time in seconds and number of iterations required for $\ \mathbf{r}_j\ /\ \mathbf{r}_0\  < 10^{-12}$ . The lowest execution times per $m_x$ are highlighted in green. . . . .                                                                    | 55 |
| 4.7 | Comparative summary of numerical methods with different preconditioners. Order of accuracy: $k = 4$ . Average computational time in seconds and number of iterations required for $\ \mathbf{r}_j\ /\ \mathbf{r}_0\  < 10^{-12}$ . The lowest execution times per $m_x$ are highlighted in green. . . . .                                                                    | 55 |

|      |                                                                                                                                                                                                                                                                                                                                                    |    |
|------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|----|
| 4.8  | Comparative summary of numerical methods with different preconditioners. Order of accuracy: $k = 6$ . Average computational time in seconds and number of iterations required for $\ \mathbf{r}_j\ /\ \mathbf{r}_0\  < 10^{-12}$ . The lowest execution times per $m_x$ are highlighted in green. . . . .                                          | 56 |
| 4.9  | Comparative summary of numerical methods with different preconditioners. Order of accuracy: $k = 8$ . Average computational time in seconds and number of iterations required for $\ \mathbf{r}_j\ /\ \mathbf{r}_0\  < 10^{-12}$ . The lowest execution times per $m_x$ are highlighted in green. . . . .                                          | 56 |
| 4.10 | Comparative summary of numerical methods for fine-discretization problems ( $m_x = 256, 512$ ): no preconditioning vs. Jacobi preconditioning. Average computational time in seconds and number of iterations required for $\ \mathbf{r}_j\ /\ \mathbf{r}_0\  < 10^{-12}$ . The lowest execution times per $m_x$ are highlighted in green. . . . . | 57 |

# List of Figures

|     |                                                                                                                                                                                                                                                                                                  |    |
|-----|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|----|
| 2.1 | 1D uniform grid with $m_x = 5$ cells. . . . .                                                                                                                                                                                                                                                    | 8  |
| 2.2 | 1D staggered grid with $m_x = 5$ cells. Adapted from <a href="#">Dumett and Castillo (2024)</a> , page 3. . . . .                                                                                                                                                                                | 10 |
| 2.3 | 2D uniform staggered grid . . . . .                                                                                                                                                                                                                                                              | 22 |
| 2.4 | Evaluation scheme for the $x$ -component of the gradient of a scalar function $u$ at the midpoint of the edge between two adjacent cells of a 2D staggered grid. Adapted from <a href="#">Runyan (2011)</a> , page 26. . . . .                                                                   | 22 |
| 2.5 | Evaluation scheme for the divergence of a flux at the cell center of a 2D cell. Adapted from <a href="#">Runyan (2011)</a> , page 25. . . . .                                                                                                                                                    | 23 |
| 2.6 | Numerical solution of the minimal_poisson2D.m example from MOLE. Adapted from <a href="#">Corbino et al. (2024)</a> . . . . .                                                                                                                                                                    | 26 |
| 3.1 | Partition of matrix $A$ into $D$ , $L$ , and $U$ . Adapted from <a href="#">Saad (2003)</a> , page 106. . . . .                                                                                                                                                                                  | 29 |
| 3.2 | Iterative method for the linear system $A\mathbf{x} = \mathbf{b}$ as feedback control loop. Adapted from <a href="#">Bhaya and Kaszkurewicz (2006)</a> , page 73. . . . .                                                                                                                        | 30 |
| 4.1 | Numerical solution of the 2D Poisson equation, Problem A. . . . .                                                                                                                                                                                                                                | 47 |
| 4.2 | Numerical solution of the 2D Poisson equation, Problem B. . . . .                                                                                                                                                                                                                                | 48 |
| 4.3 | Log-log plot of the error norm $E_\infty$ vs. the cell size $h$ . The slopes of the dotted lines enclose the estimated orders of accuracy. . . . .                                                                                                                                               | 51 |
| 4.4 | Convergence curves for the built-in MATLAB®/Octave function GMRES(30) for Problem A with $k = 2$ and different discretization sizes $h = 1/m_x$ . The plot shows the relative residual norm versus the number of restart cycles. For $m_x = 512$ , GMRES(30) does not reach convergence. . . . . | 52 |
| 4.5 | Comparison of execution times (in seconds) between no preconditioning and Jacobi preconditioning, for the selected problems listed in Table 4.10. . . . .                                                                                                                                        | 57 |
| 4.6 | Relative residual norms for Problem A with $k = 4$ and $m_x = 128$ : no preconditioner, Jacobi, Gauss-Seidel and ILU, plus LGMRES. . . . .                                                                                                                                                       | 58 |

|     |                                                                                                                                                                                                                                                                                                                                 |    |
|-----|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|----|
| 4.7 | Evolution of the restart parameter $m$ and the relative residual norm for PD-GMRES( $m_j$ ) with $m_{min} = 1$ , $m_{max} = 100$ , $\alpha_P = 1$ , and $\alpha_D = 1$ , problem A with $k = 4$ and $m_x = 32, 64, 128, 256$ . . . . .                                                                                          | 59 |
| A.1 | A sketch of the trade-off between truncation and roundoff errors in the execution of a numerical method. As step-size gets smaller, the point of diminishing returns indicates that the roundoff errors begin to cancel the benefits of that step-size reduction. Adapted from <a href="#">Chapra (2011)</a> , page 79. . . . . | 68 |
| B.1 | Poster presentation shown at the CNMAC 2023. . . . .                                                                                                                                                                                                                                                                            | 74 |
| B.2 | Poster presentation shown at the CNMAC 2024. . . . .                                                                                                                                                                                                                                                                            | 75 |
| B.3 | Poster presentation shown at the RED-AMARU event: Mimetic Operators and Iterative Methods for the Resolution of Flow Equations. . . .                                                                                                                                                                                           | 77 |
| B.4 | Poster presentation shown at the RED-AMARU event: Krylov Subspace-Based Adaptive Solvers. . . . .                                                                                                                                                                                                                               | 78 |

# Chapter 1

## Introduction

Physics describes real-world phenomena using the language of mathematics, usually in the form of partial differential equations (PDEs). Although analytical solutions of these PDEs might not always be available, one can approximate solutions to these equations using numerical methods. The discretization of PDEs is a crucial step in this process, as it transforms the continuous problem into a discrete one that can be solved using computational techniques.

Mimetic discretizations of differential operators from vector calculus such as gradient, divergence, Laplacian, and curl (Castillo and Grone, 2003; Corbino and Castillo, 2020) can be built to satisfy the discrete analogues of the continuous conservation properties (conservation of mass, energy, etc.). A pivotal aspect of mimetic discretization research has been the development of software libraries and tools that facilitate the implementation and application of these methods. The Mimetic Operators Library Enhanced (MOLE) represents a significant leap in this direction by offering a repository of mimetic operators for community use (Corbino et al., 2024).

The computational solution of the sparse linear systems of equations that arise from this mimetic discretization is of particular interest. Due to the potentially large size and sparsity of these systems, direct solvers are often impractical, hence, iterative methods are the preferred computational tool. These methods enable efficient approximations to a specified precision. Among them, Krylov subspace-based algorithms, such as the generalized minimal residual method (GMRES) proposed by Saad and Schultz (1986), are widely adopted for their robustness and effectiveness in handling non-symmetric and ill-conditioned matrices.

However, iterative methods might suffer slow convergence; therefore, to overcome this problem, the application of preconditioning techniques is usually required. The effectiveness of a preconditioner depends on its ability to transform the original

linear system into one that is easier for the iterative method to solve, ideally reducing the number of iterations required to reach the approximate solution.

Selecting suitable preconditioners and iterative methods is therefore crucial for achieving efficient and robust numerical solutions. This thesis explores combinations of classical preconditioning techniques—such as Jacobi, Gauss-Seidel, SOR, and ILU—with various Krylov subspace-based iterative methods, in a search to determine the practical discretization limits for the mimetic scheme and to assess the efficacy of these numerical methods in solving the resulting linear systems. While the restarted GMRES, denoted as  $\text{GMRES}(m)$ , is a common choice for the sparse, non-symmetric systems arising from mimetic discretizations, it can struggle with convergence for large problems. In a new MATLAB®/Octave library called *KrySBAS* (Krylov Subspace Based Adaptive Solvers), several variants of the  $\text{GMRES}(m)$  method were implemented; GMRES-E (Morgan, 1995), LGMRES (Baker et al., 2005), and PD-GMRES (Nuñez et al., 2018).

## 1.1 Objectives

The main objective of this book is to analyze of the performance of mimetic discretization methods applied to the Poisson equation using different preconditioned iterative solvers. The specific objectives are:

- To numerically verify that the implemented mimetic operators for the two-dimensional Poisson equation achieve their theoretical orders of convergence.
- To collaborate in the design and implementation of the open-source MATLAB®/-Octave library KrySBAS.
- To compare the performance of the LGMRES, GMRES-E, and PD-GMRES methods against the standard  $\text{GMRES}(m)$ , evaluating their efficiency in terms of iteration count and execution time on the discretized problems.
- To quantify the impact of classical preconditioners —Jacobi, Gauss-Seidel, and ILU—on the convergence of the iterative solvers across a range of problem sizes and discretization orders.

## 1.2 Originality and Contribution

The originality of this work stems from its focused investigation of the practical performance of the mimetic discretization framework, a PDE numerical solving

technique explicitly designed to maintain discrete analogues of vector calculus identities (Dumett and Castillo, 2023). Also, the thesis addresses the question of how to efficiently solve the large, sparse, and often challenging linear systems generated by high-order mimetic discretizations. The core novelty lies in the comparative analysis of advanced and adaptive Krylov subspace methods (specifically, variants of GMRES) when applied to these systems, an area that has received less attention than the development of the discretization schemes themselves.

The primary contribution of this work is a performance analysis of the mimetic method for the Poisson equation, including a comparison of different preconditioning and iterative solving techniques. This analysis serves as a practical guide for selecting efficient solution strategies for similar problems. A significant complementary contribution is the development of *KrySBAS*, an open-source software library for MATLAB®/Octave that provides a well-structured and accessible implementation of several GMRES variants, a tangible tool that facilitates reproducible scientific research in the field. Finally, this work contributes to the body of knowledge by empirically establishing the practical limits of the mimetic scheme for various discretization sizes. It could highlight the effectiveness of specific preconditioner-solver pairs and identifies bottlenecks that might be addressed in future works.

## 1.3 Organization of the Book

The book is organized as follows: Chapter 2 presents the Poisson equation, together with the mimetic methods and staggered grids for 1D discretization and their generalization to 2D grids. Chapter 3 introduces Numerical Linear Algebra tools required for the discrete problem, including an overview of the employed iterative methods and preconditioners. Chapter 4 describes the setup for the numerical experiments, including the verification of convergence orders of the mimetic operators, and the evaluation of the performance on several Krylov-subspace iterative methods and classic preconditioners. Finally, in Chapter 5 some conclusions and future research directions are presented.

## 1.4 Notation

Throughout this book, we adopt the following notation. Continuous scalar fields are represented by lowercase italic letters, such as  $u$  or  $f$ . Continuous vector fields are denoted by italic letters with an arrow above, for example,  $\vec{F}$  or  $\vec{v}$ , except the unit

vectors  $\hat{\mathbf{i}}$ ,  $\hat{\mathbf{j}}$ , and  $\hat{\mathbf{k}}$  from the Cartesian coordinate system, which are denoted by lowercase, boldfaced letters. In the discrete setting, values stored at specific grid points are denoted by lowercase italic letters with subscripts, such as  $u_i$  or  $u_{i+1/2}$ . Computational vectors, which contain these discrete values, are represented by lowercase, boldfaced letters, for instance,  $\mathbf{u}$ . Matrices are denoted by uppercase italic letters, such as  $A$ ,  $D$ , or  $G$ .

# Chapter 2

## Mimetic Discretization Methods

When dealing with physical quantities like pressure, temperature, mass concentration or electric charge, it is usual to model them either as a continuous distribution, or to discretize that distribution at a finite set of points in space (Castillo and Miranda, 2013). Mathematically ideal objects can be modeled as follows:

1. Points can model tiny particles,
2. 1-D lines, whether straight or curved, can be used to model thin wires, capillary blood vessels, etc.,
3. 2-D surfaces provide good representations of thin plates, membranes, or sheets,
4. 3-D solids and fluids can be used to describe the physical domain in 3-D space.

It is common to first consider physical quantities as real-valued functions (fields), continuously distributed throughout the domain under consideration. There can be different kinds of mathematical fields, defined by the nature of the physical quantities:

1. Scalar-valued functions (scalar fields), such as pressure, mass density, electric charge density, temperature, and gravitational or electrostatic potentials.
2. Vector-valued functions (vector fields), including gravitational, electric or magnetic fields, gradients of potentials, velocity fields describing fluid motion, etc.
3. Tensor-valued functions (tensor fields), such as the stress tensor defined for elastic bodies.

Vector calculus provides mathematical tools to describe natural phenomena in terms of scalar and vector fields (Arfken et al., 2011). A physically-motivated example is the *flux*, which quantifies the rate of flow of a physical quantity (such as mass,

heat, or electrical charge) through a given surface. This rate of flow is mathematically described by a vector field (the flux density vector) that represents the magnitude and direction of the flow at each point in space.

For the next mathematical statements, consider a region  $\Omega \subset \mathbb{R}^d$ , where  $d = 1, 2$  or  $3$ , with boundary  $\partial\Omega$ . The total flux of a vector field  $\vec{F}$  across  $\partial\Omega$  is defined by the integral

$$\Phi = \int_{\partial\Omega} \vec{F} \cdot \vec{n} \, dS, \quad (2.1)$$

where  $\vec{n}$  is the outward normal unit vector to  $\partial\Omega$ .

Often, we want to quantify the change in the flux  $\vec{F}$  over a spatial region. To this aim, we use the *divergence* of a vector field  $\vec{F}$ , denoted  $\nabla \cdot \vec{F}$ , which is defined in Cartesian coordinates, for  $\vec{F} = F_x \hat{\mathbf{i}} + F_y \hat{\mathbf{j}} + F_z \hat{\mathbf{k}}$ , as

$$\nabla \cdot \vec{F} = \frac{\partial F_x}{\partial x} + \frac{\partial F_y}{\partial y} + \frac{\partial F_z}{\partial z}. \quad (2.2)$$

The *Gauss Divergence Theorem* states that the integral of the divergence of  $\vec{F}$  over the region  $\Omega$  is equal to the flux of  $\vec{F}$  across  $\partial\Omega$ :

$$\int_{\Omega} \nabla \cdot \vec{F} \, dV = \oint_{\partial\Omega} \vec{F} \cdot \vec{n} \, dS. \quad (2.3)$$

The *gradient* of a scalar field  $f$ , denoted by  $\nabla f$ , is a vector field that points in the direction of greatest rate of increase of  $f$ , and whose magnitude is that rate of increase. In Cartesian coordinates, it is defined by

$$\nabla f = \frac{\partial f}{\partial x} \hat{\mathbf{i}} + \frac{\partial f}{\partial y} \hat{\mathbf{j}} + \frac{\partial f}{\partial z} \hat{\mathbf{k}}. \quad (2.4)$$

The next identity gives a relation between the divergence and gradient operators. Let the divergence be applied to the product  $f \vec{v}$ , where  $f$  is a scalar field (e.g., mass density),  $\vec{v}$  is a vector field (e.g., velocity), then

$$\begin{aligned} \nabla \cdot (f \vec{v}) &= \frac{\partial}{\partial x}(f v_x) + \frac{\partial}{\partial y}(f v_y) + \frac{\partial}{\partial z}(f v_z), \\ &= \frac{\partial f}{\partial x} v_x + f \frac{\partial v_x}{\partial x} + \frac{\partial f}{\partial y} v_y + f \frac{\partial v_y}{\partial y} + \frac{\partial f}{\partial z} v_z + f \frac{\partial v_z}{\partial z}, \\ &= f \nabla \cdot \vec{v} + \vec{v} \cdot \nabla f. \end{aligned} \quad (2.5)$$

The extended form of Gauss' divergence theorem states that, if  $\vec{F} = f \vec{v}$ , then

$$\int_{\Omega} f \nabla \cdot \vec{v} dV + \int_{\Omega} \vec{v} \cdot \nabla f dV = \oint_{\partial\Omega} f \vec{v} \cdot \vec{n} dS. \quad (2.6)$$

Gradient and divergence operators appear naturally in many PDEs. Arguably, the simplest of them is the Poisson equation, which is introduced next.

## 2.1 Poisson Equation

The *Poisson equation* is an elliptical PDE, ubiquitous in many areas of science and engineering, and therefore a usual starting point when analyzing continuum mathematical models. Some of the common uses include –but are not limited to– heat transfer, diffusion processes, and electrostatics. In these kind of numerical simulations, most of the computational resources are dedicated to its solution (Schaerer et al., 2004). Analytical solutions are often limited to simple geometries and boundary conditions and may not exist for cases of practical interest involving complex geometries. Therefore, numerical approximations to the Poisson equation are of significant interest.

If  $\Omega$  is an open and bounded domain whose boundary  $\partial\Omega$  is sufficiently smooth, the Poisson equation describes a *potential function*  $u : \Omega \rightarrow \mathbb{R}$  that satisfies

$$\nabla \cdot (-\nabla u) = f \quad \text{in } \Omega, \quad (2.7)$$

$$\alpha u + \beta \nabla u \cdot \vec{n} = g \quad \text{on } \partial\Omega. \quad (2.8)$$

where  $f : \Omega \rightarrow \mathbb{R}$  is a *source function* or *forcing function*. In the case  $f = 0$ , the equation reduces to the *Laplace equation*. Equation (2.8) imposes a *Robin boundary condition*, representing a weighted combination of the value of the desired function  $u$  and the value of its normal derivative at the boundary  $\partial\Omega$ . The Robin condition is a general form that encompasses two other common types of boundary conditions: when  $\alpha = 1, \beta = 0$ , it simplifies to the *Dirichlet boundary condition*, which prescribes the value of  $u$  on the boundary, and when  $\alpha = 0, \beta = 1$ , it becomes the *Neumann boundary condition*, which prescribes the value of the normal derivative of  $u$  on the boundary.

## 2.2 Finite Difference Methods

For the numerical solution of PDEs arising from several fields in the computational sciences, the usual spatial discretization techniques include the finite volume, finite element, and finite difference methods. Among these, the *Finite Difference Method*

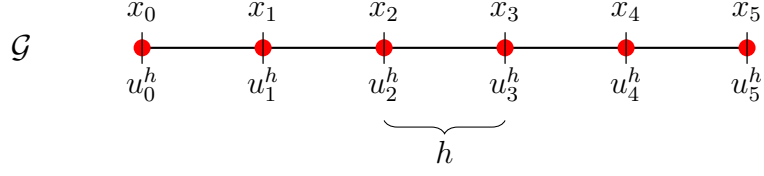


Figure 2.1: 1D uniform grid with  $m_x = 5$  cells.

(FDM) is often the most straightforward to implement. FDM is a class of numerical techniques for solving PDEs by approximating derivatives with linear combinations of discrete function values.

These methods seek an unknown continuous function (e.g.,  $u$  for the Poisson equation) on a discrete set of points, the *grid*  $\mathcal{G}$ , which partitions the continuous domain  $\Omega$ . The numerical solution is a finite set of numbers that approximates the unknown function over  $\mathcal{G}$ . For a one-dimensional domain  $\Omega = [0, 1]$ , a uniform grid is characterized by a parameter  $h$ , the *step size*. The domain is partitioned into  $m_x$  subintervals  $[x_{i-1}, x_i]$  or *cells*, each of size  $h = 1/m_x$ . This creates  $m_x + 1$  evenly spaced points  $x_i = ih$  for  $i = 0, 1, \dots, m_x$ , as shown in Figure 2.1.

This discretization process returns a vector of discrete values representing the numerical approximation of  $u$ :

$$\mathbf{u}^h = [u_0^h, u_1^h, u_2^h, \dots, u_{m_x}^h]^T. \quad (2.9)$$

Each component of this vector,  $u_i^h$ , is an approximation of the continuous solution at a specific grid point, i.e.,  $u_i^h \approx u(\vec{x}_i)$  for  $\vec{x}_i \in \mathcal{G}$ . As  $h \rightarrow 0$ ,  $\mathcal{G}$  becomes a finer partition of  $\Omega$ . Consequently, the dimension of the solution vector  $\mathbf{u}^h$  increases, leading to higher computational costs.

A crucial property of any numerical method is *convergence*, which ensures that the numerical solution approaches the exact solution as the grid is refined. A method is convergent if

$$\lim_{h \rightarrow 0} \|\mathbf{u}^h - \mathbf{u}^*\| = 0, \quad (2.10)$$

where  $\mathbf{u}^*$  is the vector of exact solution values at the grid points  $\mathcal{G}$ . The difference between the numerical and exact solutions,  $\mathbf{u}^h - \mathbf{u}^*$ , is measured using a suitable vector norm, such as the Euclidean or maximum norm (see Appendix A for definitions).

The methods presented in this work, known as *mimetic operators*, fall into the category of finite difference methods. They receive the name from the fact that they "mimic" key identities from vector calculus, allowing them to preserve physical conservation laws at the discrete level.

### 2.2.1 1D Approximation of Poisson's Equation

An application of FDM (Demmel, 1997; Saad, 2003) is the approximation of the one-dimensional Poisson equation

$$-\frac{d^2u}{dx^2} = f(x), \text{ on } 0 < x < 1. \quad (2.11)$$

where  $f(x)$  is a known source term and  $u(x)$  is the unknown function that we want to compute. Suppose the boundary conditions are of the Dirichlet-type:

$$u(0) = u_0, \quad u(1) = u_{m_x}. \quad (2.12)$$

The discretization of the second-order derivative in (2.11) is done using centered finite differences:

$$\left. -\frac{d^2u}{dx^2} \right|_{x=x_i} \cong \frac{-u_{i+1}^h + 2u_i^h - u_{i-1}^h}{h^2}, \quad i = 1, \dots, m_x - 1. \quad (2.13)$$

The discretized form of the Poisson equation can be written in matrix form as

$$\underbrace{\begin{bmatrix} 2 & -1 & 0 & \cdots & 0 \\ -1 & 2 & -1 & \cdots & 0 \\ 0 & -1 & 2 & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \cdots & 2 \end{bmatrix}}_L \underbrace{\begin{bmatrix} u_1^h \\ u_2^h \\ u_3^h \\ \vdots \\ u_{m_x-1}^h \end{bmatrix}}_{\mathbf{u}^h} = h^2 \underbrace{\begin{bmatrix} f_1 + u_0 \\ f_2 \\ f_3 \\ \vdots \\ f_{m_x-1} + u_{m_x} \end{bmatrix}}_{\mathbf{b}} \quad (2.14)$$

The discretization results in a linear system of equations

$$L\mathbf{u}^h = \mathbf{b}, \quad (2.15)$$

where  $L$  is symmetric positive definite, for which efficient algorithms already exist, such as the tridiagonal matrix algorithm (TDMA) or Thomas algorithm (Thomas, 1949), and the conjugate gradient method (Hestenes et al., 1952). The methods that will be studied in the next sections, however, generate nonsymmetric systems, and the appropriate methods for solving these problems are not fully known.

The resulting matrix of coefficients,  $L$ , has condition number  $\kappa \approx \frac{4}{\pi^2}(m_x + 1)^2$  (Strang, 2006), which is dependent of the number of discretization points. This means that, the better we approximate  $-u'' = f$  by increasing  $m_x$ , the harder it is to compute the approximate solution. At a certain crossover point, the condition number becomes

so large that the solution is no longer accurate. This is a well-known problem in numerical analysis, and it is called *ill-conditioning*. See Appendix A for more details about the condition number of a matrix.

### 2.2.2 1D Staggered grid

This subsection presents a variant of FDM that is more akin to the physical properties of a problem. Given a certain number of discretization cells, each one can be interpreted as a subdivision of the domain, with its own inner points and local boundaries. A staggered grid adds grid points that are superimposed on the original grid: these new points are the *cell centers*, while the original ones are the *nodal points*. The grid is called *staggered* because it "staggeres" or arranges the variables of interest at different locations within the grid cells. To be precise, potentials are evaluated at the cell centers, whereas fluxes are evaluated at the nodal points. On a one-dimensional staggered grid  $\Omega = [0, 1]$ , shown in Figure 2.2, consider  $m_x$  cells with step size  $h_x = 1/m_x$ . The following sets are defined:

$$\mathcal{C} = \left\{ \left( i + \frac{1}{2} \right) h_x, 0 \leq i \leq (m_x - 1) \right\} \quad (\text{set of centers}) \quad (2.16)$$

$$\mathcal{N} = \{ i h_x, 0 \leq i \leq m_x \} \quad (\text{set of nodes}) \quad (2.17)$$

$$\hat{\mathcal{C}} = \mathcal{C} \cup \{0, 1\} \quad (\text{centers and boundaries}) \quad (2.18)$$

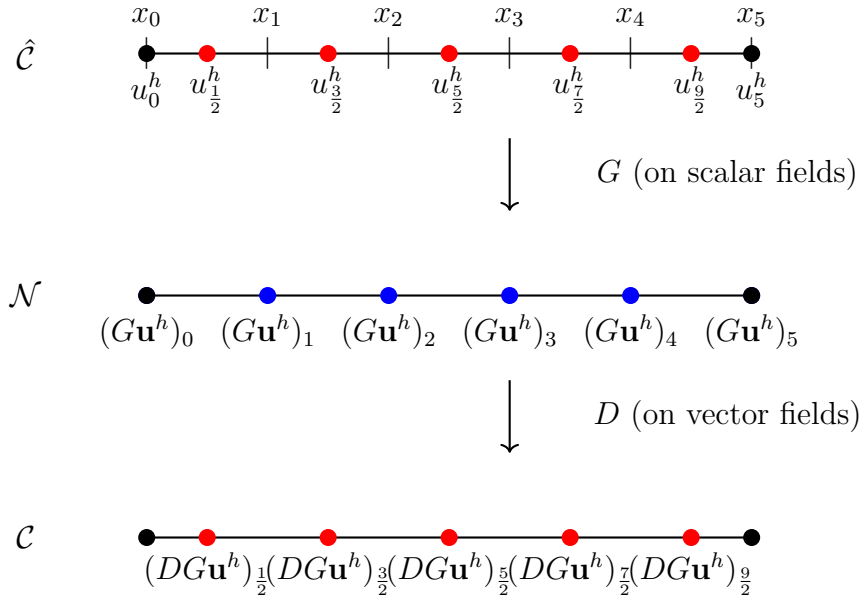


Figure 2.2: 1D staggered grid with  $m_x = 5$  cells. Adapted from [Dumett and Castillo \(2024\)](#), page 3.

Define the *projection* of the scalar function  $u$  as the computational vector  $\mathbf{u} = [u_0 \ u_{1/2} \ u_{3/2} \ \cdots \ u_{m-1/2} \ u_m]^T$  that records the scalar values at the cell centers plus external boundary, whereas the *projection* of a vector function  $\vec{v}$  is defined as the computational vector  $\mathbf{v} = [v_0 \ v_1 \ v_2 \ \cdots \ v_{m_x-1} \ v_{m_x}]^T$  that records the vector components at the nodes.

### 2.2.3 Support Operator Method

The simplest finite difference method for building discrete divergence and gradient operators on a staggered grid is the support operator method (SOM) (Shashkov and Steinberg, 1995; Shashkov, 1995), which was designed to satisfy certain integral properties. In one dimension, the operators must satisfy the next properties.

- The derivative of a constant function is zero:  $\frac{d}{dx}c = 0$  for all  $c$ ,
- The Fundamental Theorem of Calculus: if a function  $u$  is continuous in an interval  $[0, 1]$  then

$$\int_0^1 \frac{d}{dx}u(x)dx = u(1) - u(0), \quad (2.19)$$

- The 1D Gauss Divergence Theorem (integration by parts) as written in Equation 2.6:

$$\int_0^1 u(x) \frac{d}{dx}v(x) dx + \int_0^1 v(x) \frac{d}{dx}u(x) dx = u(1)v(1) - u(0)v(0).$$

In this way, the discretization of the divergence operator is given by centered finite differences on the inner points and forward/backward finite differences on the boundary, as shown in the equations below:

$$(D\mathbf{v})_{i+\frac{1}{2}} = \frac{v_{i+1} - v_i}{h_x} \quad i = 0, 1, \dots, m_x - 1, \quad (2.20)$$

$$(G\mathbf{u})_i = \frac{u_{i+\frac{1}{2}} - u_{i-\frac{1}{2}}}{h_x} \quad i = 1, 2, \dots, m_x - 1. \quad (2.21)$$

For boundary evaluations of  $G\mathbf{u}$  at  $x_0$  and  $x_{m_x}$ , the definitions of  $u_0$  and  $u_{m_x}$  are needed. The simplest way to define these values is to use the first-order finite difference

approximation of the gradient operator. This is given by

$$(G\mathbf{u})_0 = \frac{u_{\frac{1}{2}} - u_0}{\frac{h_x}{2}}, \quad (2.22)$$

$$(G\mathbf{u})_{m_x} = \frac{u_{m_x} - u_{m_x - \frac{1}{2}}}{\frac{h_x}{2}}. \quad (2.23)$$

If  $\mathbf{v} = [v_0, v_1, v_2, \dots, v_{m_x}]^T \in \mathbb{R}^{m_x+1}$ , then  $D\mathbf{v} \in \mathbb{R}^{m_x}$  represents the approximation, at the centers of the cells, of  $\nabla \cdot \vec{v}$ . In matricial form, it takes the form:

$$\begin{bmatrix} (D\mathbf{v})_{\frac{1}{2}} \\ (D\mathbf{v})_{\frac{3}{2}} \\ \vdots \\ \vdots \\ (D\mathbf{v})_{m_x - \frac{1}{2}} \end{bmatrix} = \frac{1}{h_x} \underbrace{\begin{bmatrix} -1 & 1 & & & \\ & -1 & 1 & & \\ & & \dots & \dots & \\ & & & \dots & \dots \\ & & & & -1 & 1 \end{bmatrix}}_{D \in \mathbb{R}^{m_x \times (m_x+1)}} \underbrace{\begin{bmatrix} v_0 \\ v_1 \\ \vdots \\ \vdots \\ v_{m_x} \end{bmatrix}}_{\mathbf{v}}. \quad (2.24)$$

If  $\mathbf{u} = [u_0, u_{\frac{1}{2}}, u_{\frac{3}{2}}, \dots, u_{m_x - \frac{1}{2}}, u_{m_x}]^T \in \mathbb{R}^{m_x+2}$ , then  $G\mathbf{u} \in \mathbb{R}^{m_x+1}$  represents the approximation of  $\nabla u$  at the nodes,

$$\begin{bmatrix} (G\mathbf{u})_0 \\ (G\mathbf{u})_1 \\ \vdots \\ (G\mathbf{u})_{m_x-1} \\ (G\mathbf{u})_{m_x} \end{bmatrix} = \frac{1}{h_x} \underbrace{\begin{bmatrix} -2 & 2 & & & \\ & -1 & 1 & & \\ & & \dots & \dots & \\ & & & \dots & \dots \\ & & & & -1 & 1 \\ & & & & & -2 & 2 \end{bmatrix}}_{G \in \mathbb{R}^{(m_x+1) \times (m_x+2)}} \underbrace{\begin{bmatrix} u_0 \\ u_{\frac{1}{2}} \\ \vdots \\ \vdots \\ u_{m_x - \frac{1}{2}} \\ u_{m_x} \end{bmatrix}}_{\mathbf{u}}. \quad (2.25)$$

Let  $\mathbf{1} = [1, 1, \dots, 1]^T$  be a vector of ones. It is easy to check that  $D \cdot (c\mathbf{1})$  and  $G \cdot (c\mathbf{1})$  are equal to the zero vector  $\mathbf{0}$  for any constant  $c \in \mathbb{R}$ .

The second condition (Fundamental Theorem of Calculus) requires an appropriate choice for the approximation of the integral on the discretized grid. In the case of SOM, the integral uses the values at the cell centers and the naive numerical integration rule:

$$\begin{aligned} \int_0^1 \frac{d}{dx} v(x) dx &= \sum_{i=0}^{m_x-1} \frac{v_{i+1} - v_i}{h_x} \cdot h_x \\ &= \langle D\mathbf{v}, h_x \mathbf{1} \rangle \\ &= v_{m_x} - v_0, \end{aligned} \quad (2.26)$$

or, in matricial form:

$$\begin{aligned}
\langle D\mathbf{v}, h_x \mathbb{1} \rangle &= h \mathbb{1}^T \cdot D\mathbf{v} \\
&= h_x \begin{bmatrix} 1 & 1 & \cdots & 1 & 1 \end{bmatrix} \frac{1}{h_x} \begin{bmatrix} -1 & 1 & & & \\ & -1 & 1 & & \\ & & \cdots & \cdots & \\ & & & \cdots & \cdots \\ & & & & -1 & 1 \end{bmatrix} \begin{bmatrix} v_0 \\ v_1 \\ \vdots \\ \vdots \\ v_{m_x} \end{bmatrix} \\
&= \begin{bmatrix} -1 & 0 & \cdots & 0 & 1 \end{bmatrix} \begin{bmatrix} v_0 \\ v_1 \\ \vdots \\ \vdots \\ v_{m_x} \end{bmatrix} \\
&= v_{m_x} - v_0,
\end{aligned} \tag{2.27}$$

where the angular brackets represent the classic inner product notation  $\langle \mathbf{u}, \mathbf{v} \rangle = \mathbf{v}^T \mathbf{u} = \sum_i u_i v_i$ . The operation  $\mathbb{1}^T D$  represents the sum of rows of  $D$ , which returns the vector  $[-1 \ 0 \ \cdots \ 0 \ 1]$ . For the gradient operator, the Trapezoid rule of integration satisfies the condition:

$$\begin{aligned}
\int_0^1 \frac{d}{dx} u(x) \, dx &= \sum_{i=0}^{m_x-1} \frac{(G\mathbf{u})_{i+1} + (G\mathbf{u})_i}{2} \cdot h_x \\
&= \left( \frac{1}{2}(G\mathbf{u})_0 + \sum_{i=1}^{m_x-1} (G\mathbf{u})_i + \frac{1}{2}(G\mathbf{u})_{m_x} \right) \cdot h_x \\
&= \langle G\mathbf{u}, h_x \mathbb{1} \rangle_P,
\end{aligned} \tag{2.28}$$

where  $P$  is a diagonal matrix:

$$P = \text{diag} \left( \frac{1}{2}, 1, 1, \dots, 1, \frac{1}{2} \right) \in \mathbb{R}^{(m_x+1) \times (m_x+1)}. \tag{2.29}$$

Notice that the *weighted inner product*  $\langle \mathbf{u}, \mathbf{v} \rangle_A = \mathbf{v}^T A \mathbf{u}$  is a valid inner product for

any positive definite matrix  $A$ . In matricial form:

$$\begin{aligned}
\langle G\mathbf{u}, h_x \mathbb{1} \rangle_P &= h_x \begin{bmatrix} 1 & 1 & \cdots & 1 & 1 \end{bmatrix} \frac{1}{h_x} \begin{bmatrix} \frac{1}{2} & & & & \\ & 1 & & & \\ & & \ddots & & \\ & & & 1 & \\ & & & & \frac{1}{2} \end{bmatrix} \begin{bmatrix} (G\mathbf{u})_0 \\ (G\mathbf{u})_1 \\ \vdots \\ (G\mathbf{u})_{m_x-1} \\ (G\mathbf{u})_{m_x} \end{bmatrix} \\
&= \begin{bmatrix} \frac{1}{2} & 1 & \cdots & 1 & \frac{1}{2} \end{bmatrix} \begin{bmatrix} -2 & 2 & & & \\ & -1 & 1 & & \\ & & \cdots & \cdots & \\ & & & \cdots & \cdots \\ & & & & -1 & 1 \\ & & & & & -2 & 2 \end{bmatrix} \begin{bmatrix} u_0 \\ u_{\frac{1}{2}} \\ \vdots \\ \vdots \\ u_{m_x-\frac{1}{2}} \\ u_{m_x} \end{bmatrix} \\
&= \begin{bmatrix} -1 & 0 & \cdots & 0 & 1 \end{bmatrix} \begin{bmatrix} u_0 \\ u_{\frac{1}{2}} \\ \vdots \\ \vdots \\ u_{m_x-\frac{1}{2}} \\ u_{m_x} \end{bmatrix} \\
&= u_{m_x} - u_0.
\end{aligned} \tag{2.30}$$

The third mimetic condition requires that

$$\int_0^1 u(x) \frac{d}{dx} v(x) dx = \langle D\mathbf{v}, \mathbf{u} \rangle, \tag{2.31}$$

however,  $D\mathbf{v} \in \mathbb{R}^{m_x}$  and  $\mathbf{u} \in \mathbb{R}^{m_x+2}$ , and since they are vectors from different sizes, this can not be immediately done. A new operator  $\hat{D} \in \mathbb{R}^{(m_x+2) \times (m_x+1)}$  with padded zeros on boundary points can overcome this difficulty:

$$\hat{D} = \begin{bmatrix} -- & 0 & -- \\ & D & \\ -- & 0 & -- \end{bmatrix} \tag{2.32}$$

Since  $\hat{D}v \in \mathbb{R}^{(m_x+2)}$ , it follows that

$$\begin{aligned}
\int_0^1 u(x) \frac{d}{dx} v(x) dx + \int_0^1 v(x) \frac{d}{dx} u(x) dx &= \langle D\mathbf{v}, h_x \mathbf{u} \rangle + \langle \mathbf{v}, h_x G \mathbf{u} \rangle_P \\
&= h_x \mathbf{u}^T D \mathbf{v} + h_x (G \mathbf{u})^T P \mathbf{v} \\
&= h_x \mathbf{u}^T D \mathbf{v} + h_x \mathbf{u}^T G^T P \mathbf{v} \\
&= h_x (\mathbf{u}^T D \mathbf{v} + \mathbf{u}^T G^T P \mathbf{v}) \\
&= h_x (\mathbf{u}^T (D \mathbf{v} + G^T P \mathbf{v})) \\
&= h_x \langle D \mathbf{v} + G^T P \mathbf{v}, \mathbf{u} \rangle \\
&= h_x \langle (D + G^T P) \mathbf{v}, \mathbf{u} \rangle \\
&= \langle B \mathbf{v}, \mathbf{u} \rangle.
\end{aligned} \tag{2.33}$$

where the resulting matrix

$$B = h_x (D + G^T P) \tag{2.34}$$

is called the *boundary operator*.

## 2.3 Mimetic Operators

The Support Operator Method does not exhibit the same order of accuracy in the domain as near the boundaries. However, it represent an important starting point for the introduction of mimetic discretization methods, which are able to overcome this mentioned problem. The *mimetic operators* are discrete, matricial analogs of vector calculus operators gradient ( $G \equiv \nabla$ ), divergence ( $D \equiv \nabla \cdot$ ), Laplacian ( $L \equiv \nabla^2$ ) and curl ( $C \equiv \nabla \times$ ), designed to preserve the properties of the continuum differential operators. They are constructed to adhere to a discrete version of the identities from the continuum mathematical model. They are capable of achieving the same high order of accuracy in every point of the domain and can implement boundary conditions inside their matrix structure.

This area of study has evolved from the seminal work of [Castillo and Grone \(2003\)](#), which developed matrix techniques that allow the construction of discrete versions of the divergence and gradient operators with the same order of approximation at the boundary and at the inner points. In their approach, the *Castillo-Grone (CG) mimetic operators* have a set of free parameters that need to be selected. More recently, [Corbino and Castillo \(2020\)](#) developed a slightly different, more intuitive approach for the construction of these matrices, here called *Corbino-Castillo (CC) mimetic operators*, without the need for free parameters and with improvements in accuracy when

compared to the CG counterparts. Since PDEs are described in terms of continuum differential operators, the mimetic discretization method has found several applications in image processing (Bazan et al., 2011), electromagnetism (Runyan, 2011), and fluid dynamics (Brzenski and Castillo, 2023), among others.

The mimetic operators follows similar properties to the original continuous counterparts:

- the gradient of a constant scalar function is zero:  $G\mathbf{u}_{const} = 0$ ,
- the divergence of a constant vector field is zero:  $D\mathbf{v}_{const} = 0$ ,
- the curl of the gradient of any scalar field is zero:  $CG\mathbf{u} = 0$ ,
- the divergence of the curl of a vector field is zero:  $DC\mathbf{v} = 0$ ,
- the divergence of the gradient is equal to the laplacian of any scalar field:  $DG\mathbf{u} = L\mathbf{u}$ , and
- the Gauss divergence theorem by 2.6 is fulfilled.

Like SOM, mimetic operators are defined on staggered grids. In this type of grids, scalar variables are stored in the centers of the cells; while vector components are placed on edges (or faces, in 3D).

The main idea behind their construction is to find approximations that, in the discrete realm, satisfy the identities from vector calculus like the extended Gauss Divergence theorem for a scalar function  $f$  and a vector function  $\vec{v}$  on a domain  $\Omega$  with boundary  $\partial\Omega$ .

The discrete version of the Equation (2.6) requires that

$$h_x \langle D\mathbf{v}, \mathbf{u} \rangle + h_x \langle \mathbf{v}, G\mathbf{u} \rangle = u_{m_x} v_{m_x} - u_0 v_0, \quad (2.35)$$

where  $\mathbf{u}, \mathbf{v}$  are the projections of  $u, \vec{v}$  to the finite grids  $\hat{\mathcal{C}}$  and  $\mathcal{N}$  respectively, and the angular brackets mean that the integrals are approximated utilizing a discrete inner product. This is not possible to achieve with the classical inner product, unless special weighted inner products are used for the quadratures (Castillo and Grone, 2003):

$$h_x \langle D\mathbf{v}, \mathbf{u} \rangle_Q + h_x \langle \mathbf{v}, G\mathbf{u} \rangle_P = \langle B\mathbf{v}, \mathbf{u} \rangle. \quad (2.36)$$

Here,  $P$  and  $Q$  are diagonal matrices with positive weights. That is,  $P$  and  $Q$  are the quadrature matrices corresponding to the weighted inner products of the scalar

and vector functions. The resulting mimetic boundary operator can be derived from Equation (2.36) to be:

$$h_x(QD + G^T P) = B. \quad (2.37)$$

The matrix structure of either  $D$  and  $G$  can be written (Corbino and Castillo, 2020) in terms of smaller matrices or blocks, as

$$\begin{bmatrix} A & & \\ & S & \\ & & A' \end{bmatrix}, \quad (2.38)$$

where  $A, A' \in \mathbb{R}^{b \times \hat{b}}$  are the upper left and lower right blocks related to the most exterior points of the domain, with  $A' = -P_L^T A P_R$  for some permutation matrices  $P_L, P_R$  (appropriately sized, flipped identity matrices),  $S \in \mathbb{R}^{(r-2b) \times (r-2b)}$  is the  $k$ -diagonal matrix that stores the stencils describing the operator behavior at the inner points of the domain.

The value of  $k$  is the desired order of accuracy of the discretization method,  $r$  is the number of rows of the discrete operator ( $r \geq 2k + 1$ ),  $b$  is the number of rows of the blocks  $A, A'$ , and  $\hat{b}$  is the number of columns of the blocks  $A, A'$ . The structure is nonsymmetric, which already makes it different from the matrix of coefficients  $L$  from Section 2.2.1. For CG mimetic operators,  $\hat{b} = \frac{3k}{2}$  returns a parameter-dependent family of operators (Castillo and Grone, 2003), whereas for CC mimetic operators,  $\hat{b} = k + 1$  returns a parameter-free operator (Corbino and Castillo, 2020).

### 2.3.1 Building Mimetic Operators

The CC and CG matricial operators are constructed using centered finite difference stencils for the interior points of the grid, and modified stencils at, and close to, the boundary points. For a given even order of accuracy  $k$ , a vector  $\mathbf{s} = [s_1, s_2, \dots, s_k]$  fills the main diagonal bands of the block  $S$  from Equation (2.38). To build the upper left block  $A$  and lower right block  $A'$  that handle the boundaries, the vectors  $\mathbf{s}_A, \mathbf{s}_{A'}$  are modifications from  $\mathbf{s}$  to have length  $\hat{b}$ .

Take for example  $k = 4$ . For an interior nodal point  $x_i$  of a staggered grid, the derivative  $u'(x_i)$  can be approximated with a local order of accuracy  $O(h_x^4)$  by the linear combination of weights  $\sigma = [\sigma_1, \sigma_2, \sigma_3, \sigma_4]$ :

$$u'(x_i) \approx \sigma_1 u(x_{i-3/2}) + \sigma_2 u(x_{i-1/2}) + \sigma_3 u(x_{i+1/2}) + \sigma_4 u(x_{i+3/2}) \quad (2.39)$$

or, in matrix form:

$$u'(x_i) = \begin{bmatrix} \sigma_1 & \sigma_2 & \sigma_3 & \sigma_4 \end{bmatrix} \begin{bmatrix} u(x_{i-3/2}) \\ u(x_{i-1/2}) \\ u(x_{i+1/2}) \\ u(x_{i+3/2}) \end{bmatrix} \quad (2.40)$$

Assuming a function  $f$  that is sufficiently smooth, and expanding the Taylor series of  $f$  with a fourth-order local truncation error at each center point, one gets:

$$\begin{aligned} u(x_{i-3/2}) &= u(x_i) - \frac{3}{2}h_x u'(x_i) + \left(\frac{3}{2}h_x\right)^2 \frac{u''(x_i)}{2!} - \left(\frac{3}{2}h_x\right)^3 \frac{u'''(x_i)}{3!} + \left(\frac{3}{2}h_x\right)^4 \frac{u^{(4)}(x_i)}{4!} + O(h_x^5) \\ u(x_{i-1/2}) &= u(x_i) - \frac{1}{2}h_x u'(x_i) + \left(\frac{1}{2}h_x\right)^2 \frac{u''(x_i)}{2!} - \left(\frac{1}{2}h_x\right)^3 \frac{u'''(x_i)}{3!} + \left(\frac{1}{2}h_x\right)^4 \frac{u^{(4)}(x_i)}{4!} + O(h_x^5) \\ u(x_{i+1/2}) &= u(x_i) + \frac{1}{2}h_x u'(x_i) + \left(\frac{1}{2}h_x\right)^2 \frac{u''(x_i)}{2!} + \left(\frac{1}{2}h_x\right)^3 \frac{u'''(x_i)}{3!} + \left(\frac{1}{2}h_x\right)^4 \frac{u^{(4)}(x_i)}{4!} + O(h_x^5) \\ u(x_{i+3/2}) &= u(x_i) + \frac{3}{2}h_x u'(x_i) + \left(\frac{3}{2}h_x\right)^2 \frac{u''(x_i)}{2!} + \left(\frac{3}{2}h_x\right)^3 \frac{u'''(x_i)}{3!} + \left(\frac{3}{2}h_x\right)^4 \frac{u^{(4)}(x_i)}{4!} + O(h_x^5) \end{aligned}$$

or, in matrix form,

$$\begin{bmatrix} u(x_{i-3/2}) \\ u(x_{i-1/2}) \\ u(x_{i+1/2}) \\ u(x_{i+3/2}) \end{bmatrix} = \begin{bmatrix} 1 & \left(-\frac{3}{2}\right) & \left(-\frac{3}{2}\right)^2 & \left(-\frac{3}{2}\right)^3 & \left(-\frac{3}{2}\right)^4 \\ 1 & \left(-\frac{1}{2}\right) & \left(-\frac{1}{2}\right)^2 & \left(-\frac{1}{2}\right)^3 & \left(-\frac{1}{2}\right)^4 \\ 1 & \left(\frac{1}{2}\right) & \left(\frac{1}{2}\right)^2 & \left(\frac{1}{2}\right)^3 & \left(\frac{1}{2}\right)^4 \\ 1 & \left(\frac{3}{2}\right) & \left(\frac{3}{2}\right)^2 & \left(\frac{3}{2}\right)^3 & \left(\frac{3}{2}\right)^4 \end{bmatrix} \begin{bmatrix} u(x_i) \\ h_x u'(x_i) \\ \frac{h_x^2}{2} u''(x_i) \\ \frac{h_x^3}{3!} u'''(x_i) \\ \frac{h_x^4}{4!} u^{(4)}(x_i) \end{bmatrix} + O(h^5)\mathbb{1} \quad (2.41)$$

Equations (2.39) and (2.41) are rewritten as

$$\begin{bmatrix} \sigma_1 & \sigma_2 & \sigma_3 & \sigma_4 \end{bmatrix} \begin{bmatrix} 1 & \left(-\frac{3}{2}\right) & \left(-\frac{3}{2}\right)^2 & \left(-\frac{3}{2}\right)^3 & \left(-\frac{3}{2}\right)^4 \\ 1 & \left(-\frac{1}{2}\right) & \left(-\frac{1}{2}\right)^2 & \left(-\frac{1}{2}\right)^3 & \left(-\frac{1}{2}\right)^4 \\ 1 & \left(\frac{1}{2}\right) & \left(\frac{1}{2}\right)^2 & \left(\frac{1}{2}\right)^3 & \left(\frac{1}{2}\right)^4 \\ 1 & \left(\frac{3}{2}\right) & \left(\frac{3}{2}\right)^2 & \left(\frac{3}{2}\right)^3 & \left(\frac{3}{2}\right)^4 \end{bmatrix} = \begin{bmatrix} 0 & 1/h_x & 0 & 0 & 0 \end{bmatrix} \quad (2.42)$$

or, after introducing the change of variable  $s_j = h_x \sigma_j$  and transposing the matrices,

$$\underbrace{\begin{bmatrix} 1 & 1 & 1 & 1 \\ -\frac{3}{2} & -\frac{1}{2} & \frac{1}{2} & \frac{3}{2} \\ \left(-\frac{3}{2}\right)^2 & \left(-\frac{1}{2}\right)^2 & \left(\frac{1}{2}\right)^2 & \left(\frac{3}{2}\right)^2 \\ \left(-\frac{3}{2}\right)^3 & \left(-\frac{1}{2}\right)^3 & \left(\frac{1}{2}\right)^3 & \left(\frac{3}{2}\right)^3 \end{bmatrix}}_V \begin{bmatrix} s_1 \\ s_2 \\ s_3 \\ s_4 \end{bmatrix} = \begin{bmatrix} 0 \\ 1 \\ 0 \\ 0 \end{bmatrix}, \quad (2.43)$$

which has the unique solution

$$\begin{bmatrix} s_1 \\ s_2 \\ s_3 \\ s_4 \end{bmatrix} = \begin{bmatrix} \frac{1}{24} \\ -\frac{9}{8} \\ \frac{9}{8} \\ -\frac{1}{24} \end{bmatrix}.$$

The matrix  $V$  is known as the *Vandermonde matrix* and is determined by the staggered grid and the desired order of accuracy (Castillo and Grone, 2003; Runyan, 2011). The entries of  $V$  are generated from the relative distances between the point where the derivative  $u'(x_i)$  is approximated and the points where the function is sampled:  $u(x_{i-3/2})$ ,  $u(x_{i-1/2})$ ,  $u(x_{i+1/2})$ ,  $u(x_{i+3/2})$ . The relative distances are stored in the row vector  $\mathbf{d} = \left[-\frac{3}{2}, -\frac{1}{2}, \frac{1}{2}, \frac{3}{2}\right]$ .

When approximating the derivative at a point near the boundary, such as  $u'(x_{3/2})$ , a different stencil is required to maintain the desired order of accuracy. This results in a different Vandermonde system. The length of this boundary vector  $\mathbf{s}_A$  or  $\mathbf{s}_{A'}$  depends on the chosen mimetic method. For the CG approach, the stencil length is  $\frac{3}{2}k$ , while for the CC approach, it is  $k + 1$ .

For example, the generator of the Vandermonde matrix for the first row of  $A$  in the CG mimetic divergence operator is  $\mathbf{d} = \left[-\frac{1}{2}, \frac{1}{2}, \frac{3}{2}, \frac{5}{2}, \frac{7}{2}, \frac{9}{2}\right]$ , and the corresponding linear system for  $\mathbf{s}_A$  is

$$\begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 \\ -\frac{1}{2} & \frac{1}{2} & \frac{3}{2} & \frac{5}{2} & \frac{7}{2} & \frac{9}{2} \\ \left(-\frac{1}{2}\right)^2 & \left(\frac{1}{2}\right)^2 & \left(\frac{3}{2}\right)^2 & \left(\frac{5}{2}\right)^2 & \left(\frac{7}{2}\right)^2 & \left(\frac{9}{2}\right)^2 \\ \left(-\frac{1}{2}\right)^3 & \left(\frac{1}{2}\right)^3 & \left(\frac{3}{2}\right)^3 & \left(\frac{5}{2}\right)^3 & \left(\frac{7}{2}\right)^3 & \left(\frac{9}{2}\right)^3 \\ \left(-\frac{1}{2}\right)^4 & \left(\frac{1}{2}\right)^4 & \left(\frac{3}{2}\right)^4 & \left(\frac{5}{2}\right)^4 & \left(\frac{7}{2}\right)^4 & \left(\frac{9}{2}\right)^4 \end{bmatrix} \begin{bmatrix} s_1 \\ s_2 \\ s_3 \\ s_4 \\ s_5 \\ s_6 \end{bmatrix} = \begin{bmatrix} 0 \\ 1 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}. \quad (2.44)$$

In general, the Vandermonde matrix for the boundary stencils in the CG method is of size  $(k + 1) \times \frac{3}{2}k$  (Runyan, 2011), while in the CC method, it is  $(k + 1) \times (k + 1)$ . For  $k = 2$ , the CG Vandermonde matrix is  $3 \times 3$ , which yields a unique solution for the corresponding weights. However, for  $k = 4$ , the system in Equation 2.44 has size  $5 \times 6$  and is underdetermined. This leads to a family of solutions for the boundary entries that depends on free parameters  $\alpha_i$  ( $i = 1, 2, 3, 4$ ), as seen in Equation (2.45):

$$A(\alpha) = \begin{bmatrix} -\frac{11}{12} & \frac{17}{24} & \frac{3}{8} & -\frac{5}{24} & \frac{1}{24} & 0 \\ \frac{1}{24} & -\frac{9}{8} & \frac{9}{8} & -\frac{1}{24} & 0 & 0 \\ 0 & \frac{1}{24} & -\frac{9}{8} & \frac{9}{8} & -\frac{1}{24} & 0 \\ 0 & 0 & \frac{1}{24} & -\frac{9}{8} & \frac{9}{8} & -\frac{1}{24} \end{bmatrix} + \begin{bmatrix} \alpha_1 \\ \alpha_2 \\ \alpha_3 \\ \alpha_4 \end{bmatrix} \begin{bmatrix} -1 & 5 & -10 & 10 & -5 & 1 \end{bmatrix}. \quad (2.45)$$

If the same problem is solved for CC mimetic operators, a square  $(k+1) \times (k+1)$  Vandermonde matrix is generated. For  $k = 4$ , this is a  $5 \times 5$  matrix, which yields a unique solution for the weights and thus a unique  $A$ :

$$A = \begin{bmatrix} -\frac{11}{12} & \frac{17}{24} & \frac{3}{8} & -\frac{5}{24} & \frac{1}{24} & 0 \\ \frac{1}{24} & -\frac{9}{8} & \frac{9}{8} & -\frac{1}{24} & 0 & 0 \\ 0 & \frac{1}{24} & -\frac{9}{8} & \frac{9}{8} & -\frac{1}{24} & 0 \\ 0 & 0 & \frac{1}{24} & -\frac{9}{8} & \frac{9}{8} & -\frac{1}{24} \end{bmatrix}. \quad (2.46)$$

This property of matrix uniqueness represents a considerable simplification of the mimetic discretization process. For this reason, this thesis only uses CC mimetic operators. A complete list of CC divergence and gradient operators—and the respective generators for the Vandermonde matrices—for  $k = 2, 4, 6$  and  $8$  can be found in (Dumett and Castillo, 2023), and some of them are shown thereafter. For  $k = 2$ , the one-dimensional CC mimetic divergence operator is given by

$$D = \frac{1}{h_x} \begin{bmatrix} 0 & 0 & \cdots & 0 \\ -1 & 1 & & \\ & \ddots & \ddots & \\ & & -1 & 1 \\ 0 & 0 & \cdots & 0 \end{bmatrix}_{(m_x+2) \times (m_x+1)}, \quad (2.47)$$

and the mimetic gradient operator is given by

$$G = \frac{1}{h_x} \begin{bmatrix} -\frac{8}{3} & 3 & -\frac{1}{3} & & & \\ & -1 & 1 & & & \\ & & \ddots & \ddots & & \\ & & & -1 & 1 & \\ & & & \frac{1}{3} & -3 & \frac{8}{3} \end{bmatrix}_{(m_x+1) \times (m_x+2)} \quad (2.48)$$

For  $k = 4$ , the divergence matrix is given by

$$D = \frac{1}{h_x} \begin{bmatrix} 0 & \dots & & & & & \\ \frac{-11}{12} & \frac{17}{24} & \frac{3}{8} & \frac{-5}{24} & \frac{1}{24} & 0 & \dots \\ \frac{1}{24} & \frac{-9}{8} & \frac{9}{8} & \frac{-1}{24} & 0 & \dots & \\ 0 & \ddots & \ddots & \ddots & \ddots & \ddots & \end{bmatrix}_{(m_x+2) \times (m_x+1)}, \quad (2.49)$$

and the gradient operator is given by

$$G = \frac{1}{h_x} \begin{bmatrix} \frac{-352}{105} & \frac{35}{8} & \frac{-35}{24} & \frac{21}{40} & \frac{-5}{56} & 0 & \dots \\ \frac{16}{105} & \frac{-31}{24} & \frac{29}{24} & \frac{-3}{40} & \frac{1}{168} & 0 & \dots \\ 0 & \frac{1}{24} & \frac{-9}{8} & \frac{9}{8} & \frac{-1}{24} & 0 & \dots \\ \vdots & 0 & \ddots & \ddots & \ddots & \ddots & \end{bmatrix}_{(m_x+1) \times (m_x+2)}. \quad (2.50)$$

### 2.3.2 2D Staggered Grid

In a two-dimensional, rectangular staggered grid, scalar values are defined at the centers of the cells, and vector components are defined at horizontal middle cell faces and vertical middle cell faces, respectively. On the boundary, midpoints and corners are reserved for the scalar values. Figure 2.3 illustrates a 2D staggered grid, where  $m_x$  and  $m_y$  represents the number of cells in the  $x$ - and  $y$ - direction respectively.

On the 2D grid, the gradient operator is built between two cells along their common edge. In the case of Figure 2.4, the line connecting the cell-centers of two adjacent cells is orthogonal to their common edge, so *the component of the gradient* which is normal to the edge is computed (second-order approximation) as

$$(G\mathbf{u})_{i,j+\frac{1}{2}} = \frac{1}{h_x} \left( u_{i+\frac{1}{2},j+\frac{1}{2}} - u_{i-\frac{1}{2},j+\frac{1}{2}} \right). \quad (2.51)$$

In order to approximate the divergence in a discretized cell, see Figure 2.5, so

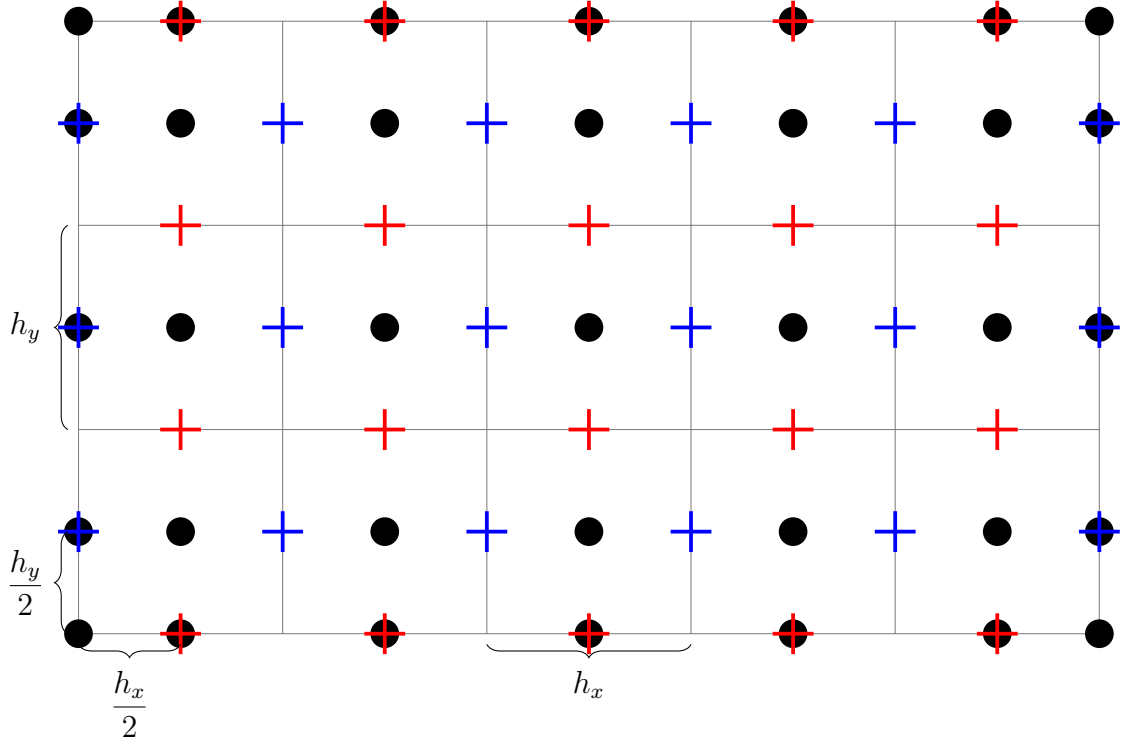


Figure 2.3: A 2D uniform staggered grid with  $m_x = 5$  horizontal cells and  $m_y = 3$  vertical cells. Adapted from [Dumett and Castillo \(2023\)](#), page 5.

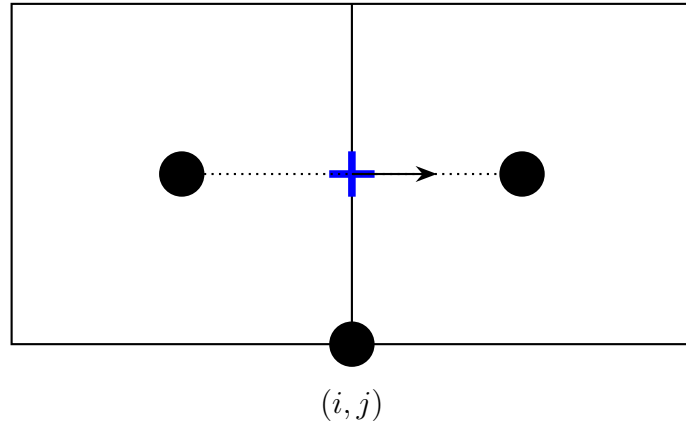


Figure 2.4: Evaluation scheme for the  $x$ -component of the gradient of a scalar function  $u$  at the midpoint of the edge between two adjacent cells of a 2D staggered grid. Adapted from [Runyan \(2011\)](#), page 26.

that it satisfies Equation (2.3.2):

$$\nabla \cdot \vec{v} = \lim_{V \rightarrow 0} \frac{1}{V} \oint_{\partial V} \vec{v} \cdot \vec{n} \, dA,$$

the flux through the edges of the cell is divided by the "volume" (area in 2D) of the

cell:

$$\nabla \cdot \vec{v} \approx \frac{1}{h_x h_y} \sum_{edges} (\vec{v}_{edge} \cdot \vec{n} h_*) \Big|_{edge}$$

where  $h_*$  means either  $h_x$  or  $h_y$  depending on the edge. It can be seen that the divergence is explained in terms of the components of the vectors that are orthogonal to the edges. By second-order approximation:

$$(D\mathbf{v})_{i+\frac{1}{2},j+\frac{1}{2}} = \frac{1}{h^2} \left( \mathbf{v}_{i+\frac{1}{2},j} \cdot (-h) + \mathbf{v}_{i+1,j+\frac{1}{2}} \cdot h + \mathbf{v}_{i+\frac{1}{2},j+1} \cdot h + \mathbf{v}_{i,j+\frac{1}{2}} \cdot (-h) \right) \quad (2.52)$$

$$= -\frac{1}{h} \left( \mathbf{v}_{i+1,j+\frac{1}{2}} - \mathbf{v}_{i,j+\frac{1}{2}} \right) + \frac{1}{h} \left( \mathbf{v}_{i+\frac{1}{2},j+1} - \mathbf{v}_{i+\frac{1}{2},j} \right). \quad (2.53)$$

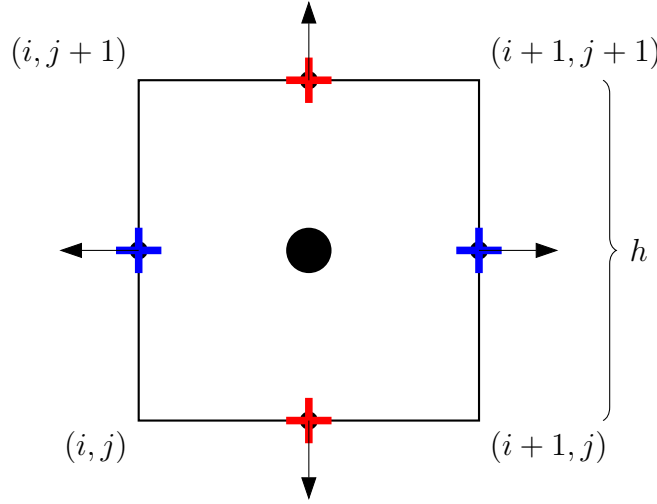


Figure 2.5: Evaluation scheme for the divergence of a flux at the cell center of a 2D cell. Adapted from [Runyan \(2011\)](#), page 25.

If  $\mathbf{v}$  contains information about the computation of the gradient mimetic operator  $G$  of some scalar function  $u$  upon which the divergence operator  $D$  will have to act, then the discrete Laplacian  $L$  of  $\mathbf{u}$  will be the product of the divergence and the gradient matrix operators:

$$L = DG. \quad (2.54)$$

### 2.3.3 2D Mimetic Operators

This section presents a direct way of representing mimetic operators in two-dimensions with the 1D mimetic operators and Kronecker products ([Castillo and Miranda, 2013](#); [Corbino and Castillo, 2020](#)). Let  $\hat{I}_m$  be an augmented identity matrix of size  $(m+2) \times m$  padded with zeros in the first and last rows. If  $m_x$  is the number of

cells in the  $x$ -direction,  $m_y$  is the number of cells in the  $y$ -direction,  $G_x$ ,  $G_y$  are the one-dimensional mimetic gradient operators in the  $x$ - and  $y$ - directions, respectively. Then, the two-dimensional mimetic gradient operator  $G$  is given by

$$G_{xy} = \begin{bmatrix} S_x \\ S_y \end{bmatrix},$$

where  $S_x = \hat{I}_{m_y}^T \otimes G_x$  and  $S_y = G_y \otimes \hat{I}_{m_x}^T$  and  $\otimes$  is the Kronecker product.

The two-dimensional mimetic divergence operator is

$$D_{xy} = \begin{bmatrix} S_x & S_y \end{bmatrix},$$

where  $S_x = \hat{I}_{m_y}^T \otimes D_x$  and  $S_y = D_y \otimes \hat{I}_{m_x}^T$ .

Three-dimensional mimetic operators can be constructed in a similar way, with the addition of the  $z$ -direction. Scalar-valued functions reside in cell centers, and vector components are placed along the cell faces. The reader is referred to [Castillo and Miranda \(2013\)](#), [Corbino and Castillo \(2020\)](#) for the construction of 3D mimetic operators.

## 2.4 MOLE: Mimetic Operators Library Enhanced

For a more straightforward design of numerical experiments on the mentioned operators, the MOLE (Mimetic Operators Library Enhanced) is used. MOLE is a free, open source library written in MATLAB®/Octave and C++ that implements high-order mimetic operators to solve PDEs. It provides discrete analogs of the most common vector calculus operators: Gradient, Divergence, Laplacian and Curl. These operators (highly sparse matrices) act on staggered grids (uniform, non-uniform, curvilinear) and satisfy local and global conservation laws. MOLE can be downloaded from <https://github.com/csirc-sdsu/mole>.

An example of the MOLE usage is the `minimal_poisson2D.m` snippet, shown in Listing 2.1. The script returns the numerical approximation of the 2D Poisson equation (2.56) with Dirichlet boundary condition

$$f(x, y) = \begin{cases} 100 & \text{if } y = 0, \\ 0 & \text{otherwise.} \end{cases} \quad (2.55)$$

Lines 8 to 10 permit the user to select the resolution of the rectangular grid in the horizontal (`m`) and vertical (`n`) directions, and the order of accuracy `k` of the

mimetic operator. Line 12 invoke the 2D mimetic Laplacian operator  $L$  with the call to the function `lap2D`, and line 13 adds the Dirichlet boundary condition, a special case of Robin boundary condition and hence the usage of function `robinBC2D`, to the matrix structure. The right-hand side  $\mathbf{b}$  is set in lines 15 to 17. The derived linear system of equations

$$L\mathbf{u}^h = \mathbf{b} \quad (2.56)$$

is solved in line 19 with the *backslash* `\` operator, and the `uh` vector is again reshaped back to a 2D grid for its visualization with lines 22 to 27.

```

1 % 2D Staggering example using a 2D Mimetic laplacian
2
3 clc
4 close all
5
6 addpath('.../src/matlab')
7
8 k = 2; % Order of accuracy
9 m = 5; % Vertical resolution
10 n = 6; % Horizontal resolution
11
12 L = lap2D(k, m, 1, n, 1); % 2D Mimetic laplacian operator
13 L = L + robinBC2D(k, m, 1, n, 1, 1, 0); % Dirichlet BC
14
15 b = zeros(m+2, n+2);
16 b(1, :) = 100; % Known value at the bottom boundary
17 b = reshape(b, [], 1);
18
19 uh = L\b;
20 uh = reshape(uh, m+2, n+2);
21
22 imagesc(uh)
23 title('2D Poisson''s equation')
24 xlabel('m')
25 ylabel('n')
26 set(gca, 'YDir', 'Normal')
27 colorbar

```

Listing 2.1: MOLE sample code for MATLAB®/Octave, adapted from [Corbino et al. \(2024\)](#).

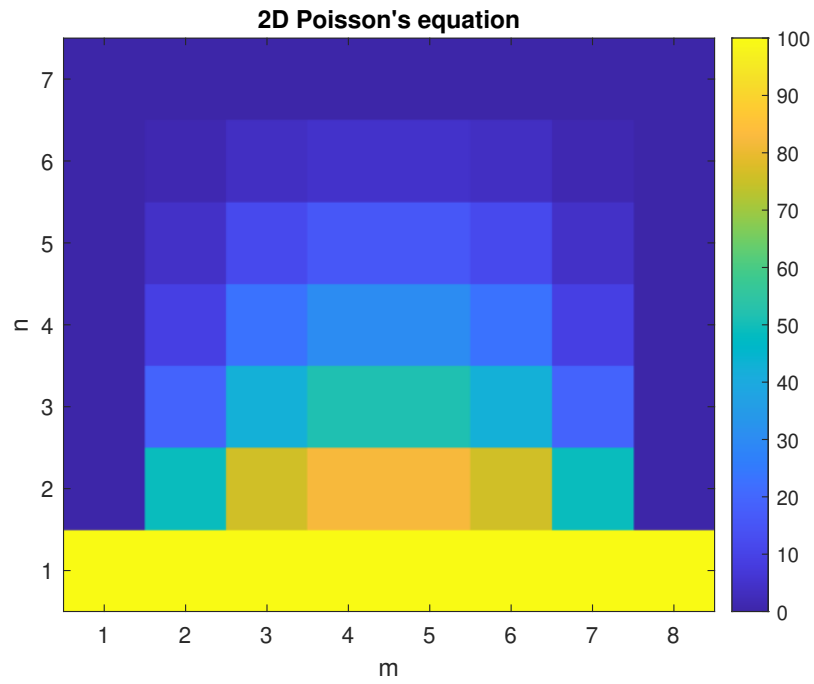


Figure 2.6: Numerical solution of the `minimal_poisson2D.m` example from MOLE. Adapted from [Corbino et al. \(2024\)](#).

# Chapter 3

## Krylov Methods for Nonsymmetric Linear Systems

In Chapter 2, the discretization of the Poisson equation resulted in a *linear system of equations* denoted as  $L\mathbf{u}^h = \mathbf{b}$ , where  $L$  is the discrete Laplacian operator,  $\mathbf{u}^h$  is the vector of unknown potential values, and  $\mathbf{b}$  is the right-hand side vector. For the general discussion of methods in this chapter, we will rather adopt the standard notation of numerical linear algebra

$$A\mathbf{x} = \mathbf{b}, \tag{3.1}$$

where  $A \in \mathbb{R}^{n \times n}$  corresponds to the coefficient matrix  $L$  and  $\mathbf{x} \in \mathbb{R}^n$  to the approximated solution vector  $\mathbf{u}$ .

Assuming that  $A$  is nonsingular, equation (3.1) has a unique solution  $\mathbf{x}^* = A^{-1}\mathbf{b}$ . The goal is to find an approximation to  $\mathbf{x}^*$  without explicitly computing  $A^{-1}$ . A well-known method for solving the linear system is Gaussian elimination. In general, this method requires storing all  $n^2$  entries of matrix  $A$  and performing  $O(n^3)$  arithmetic operations (additions, subtractions, multiplications, and divisions) (Greenbaum, 1997). However, matrices that arise in practice from the discretization of PDEs often possess special properties: they are frequently sparse, meaning that they have a large number of zero entries. In this case, it is more efficient to store only the nonzero entries and use iterative methods to solve the linear system. Also, sparsity can often be used to great advantage in matrix-vector multiplication. If a square  $n \times n$  matrix has only  $\ell$  nonzero entries per row ( $\ell \ll n$ ), then the product of this matrix will need just  $\ell n$  arithmetic operations, compared to the  $2n^2$  operations that would be required for a dense matrix-vector multiplication (Bhaya and Kaszkurewicz, 2006). This leads to the important question: can a system of linear equations be solved (or an approximate

solution be obtained) using only matrix-vector multiplications?

### 3.1 Iterative Methods for Solving Linear Systems

This section introduces a family of algorithms that generate a sequence of approximations to the solution of a linear system (Barrett et al. (1994), Saad (2003)). The main idea behind iterative methods is that the algorithm starts with an initial guess for the solution and then iteratively refines that guess until it converges to the true solution. A general iterative method, also called a *simple iterative method* (Greenbaum, 1997), for the solution of  $A\mathbf{x} = \mathbf{b}$  can be described by a recurrence relation of the form:

$$\begin{aligned}\mathbf{x}_{k+1} &= \mathbf{x}_k + K(\mathbf{b} - A\mathbf{x}_k), \\ &= \mathbf{x}_k + K\mathbf{r}_k,\end{aligned}\tag{3.2}$$

where  $K \in \mathbb{C}^{n \times n}$ , called *feedback gain matrix* (Schaerer and Kaszkurewicz, 2001; Bhaya and Kaszkurewicz, 2006), is a matrix that determines the direction and magnitude of the correction to be applied to the current approximation  $\mathbf{x}_k$ , and  $\mathbf{r}_k = \mathbf{b} - A\mathbf{x}_k$  is the residual vector at iteration  $k$ .

The choice of  $K$  determines the iterative method (Barrett et al., 1994). If  $K$  does not change at each iteration, the method is a *stationary iterative method*. These methods are easier to understand but are not always effective. If  $K$  (and other parameters) change throughout the iterations, then *non-stationary iterative methods* emerge.

Each method differs in how the matrix  $K$  is constructed. Take, for example, a classical splitting of  $A$ : let  $D$ ,  $(-L)$ , and  $(-U)$  be the diagonal, strictly lower triangular, and strictly upper triangular parts of  $A$ , respectively. The choice of the minus signs is conventional (Saad, 2003):

$$A = (-L) + D + (-U),\tag{3.3}$$

(see Figure 3.1). This partition gives rise to the most basic stationary iterative methods, which are described below.

- If  $K = D^{-1}$ , then (3.2) represents the *Jacobi* iterative method,
- If  $K = \left(\frac{1}{\omega}D - L\right)^{-1}$ , where  $0 < \omega < 2$ , then (3.2) is the *successive overrelaxation (SOR) method*. When  $\omega = 1$ , this becomes the *Gauss-Seidel* method.

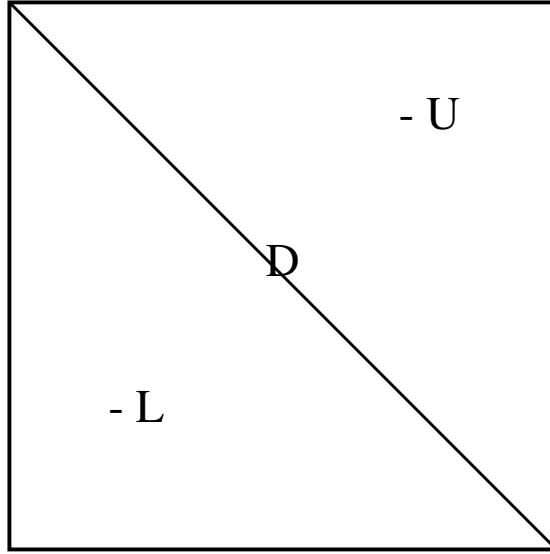


Figure 3.1: Partition of matrix  $A$  into  $D$ ,  $L$ , and  $U$ . Adapted from [Saad \(2003\)](#), page 106.

Later, it will be explained that only the matrices  $K$  are needed as preconditioners, not as parts of the iterative methods from which they originate. In this work, the chosen iterative approach is the Krylov subspace method, a non-stationary iterative method that is useful because it exploits the sparsity of matrices, such as those arising from the discretization of PDEs.

## 3.2 Control Approach to Iterative Methods

Control theory, specifically *state-space control*, states that any general iterative method for solving the linear system of equations 3.1 is a discrete-time *dynamical system* in a standard feedback control configuration ([Bhaya and Kaszkurewicz, 2006](#)) of the form:

$$\mathbf{x}_{k+1} = F\mathbf{x}_k + G\mathbf{u}_k, \quad (3.4)$$

$$\mathbf{y}_k = H\mathbf{x}_k + J\mathbf{u}_k, \quad (3.5)$$

where  $\mathbf{x}_k$ —the approximate solution returned by the iterative method—is the state variable,  $\mathbf{u}_k$  is the input, and  $\mathbf{y}_k$  is the output. Their feedback interconnections—a *closed-loop system*—are designed to achieve basic properties, such as stability, where a disturbed but bounded input leads to a bounded output. When compared with the simple iteration method of Equation (3.2), one can see that  $F = I$  (an identity matrix),

$G = K$ ,  $\mathbf{u}_k = \mathbf{r}_k = \mathbf{b} - A\mathbf{x}_k$ ,  $H = A$ , and  $J = 0$ . The choice of  $K$  is crucial for the convergence of the iterative method, as the following theorem states:

**Theorem 1.** *Let  $A \in \mathbb{R}^{n \times n}$  be a nonsingular matrix, and let  $K$  be a matrix such that the spectral radius of the iteration matrix  $I - AK$  is less than 1. Then the iterative method defined by (3.2) converges to the unique solution of the linear system  $A\mathbf{x} = \mathbf{b}$  for any initial guess  $\mathbf{x}_0$ .*

*Proof.* Observe that the residual  $\mathbf{r}_k$  can be obtained as follows:

$$\begin{aligned} A\mathbf{x}_{k+1} &= A\mathbf{x}_k + AK\mathbf{r}_k \\ \Rightarrow \mathbf{b} - A\mathbf{x}_{k+1} &= \mathbf{b} - A\mathbf{x}_k - AK\mathbf{r}_k \\ \Rightarrow \mathbf{r}_{k+1} &= (I - AK)\mathbf{r}_k \end{aligned} \tag{3.6}$$

The convergence of the linear iterative method (3.6), if thought of as a discrete linear dynamical system, is ensured if the matrix  $S = I - AK$  has all its eigenvalues within the unit disk (Ogata, 1995).  $\square$

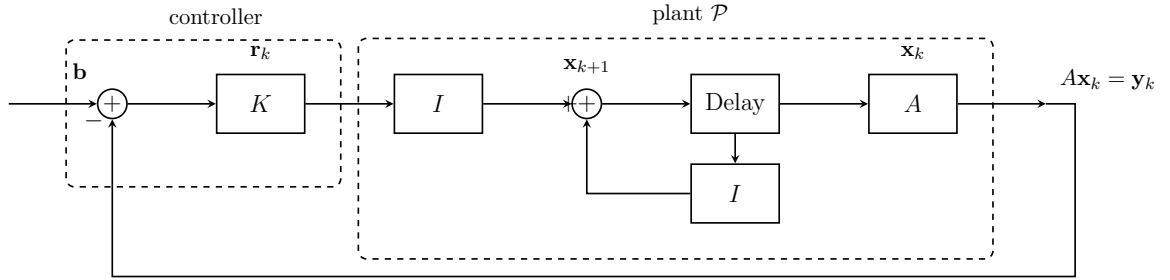


Figure 3.2: Iterative method for the linear system  $A\mathbf{x} = \mathbf{b}$  as feedback control loop. Adapted from Bhaya and Kaszkurewicz (2006), page 73.

### 3.3 Basic Iterative Methods

The first iterative methods for solving large linear systems were designed to eliminate components of the residual vector  $\mathbf{r}_k = \mathbf{b} - A\mathbf{x}_k$  (Saad, 2003). Today, these techniques are rarely used alone. However, they can be quite successful when combined with more powerful methods.

### 3.3.1 Jacobi Iteration Method

The Jacobi iteration determines the  $i$ -th component of the next approximation to annihilate the  $i$ -th component of the residual vector:

$$(\mathbf{b} - A\mathbf{x}_{k+1})^{(i)} = 0, \quad (3.7)$$

in which  $v^{(i)}$  denotes the  $i$ -th component of the vector  $\mathbf{v}$ . The component-wise form of Jacobi iteration that yields Equation 3.7 is:

$$(\mathbf{x}_{k+1})^{(i)} = \frac{1}{a_{ii}} \left( \mathbf{b}^{(i)} - \sum_{\substack{j=1 \\ j \neq i}}^n a_{ij} (\mathbf{x}_k)^{(j)} \right). \quad (3.8)$$

The matrix form of the Jacobi iteration is:

$$\mathbf{x}_{k+1} = D^{-1}(L + U)\mathbf{x}_k + D^{-1}\mathbf{b}. \quad (3.9)$$

### 3.3.2 Gauss-Seidel Iterative Method

Similarly, the Gauss-Seidel iteration corrects the  $i$ -th component of the approximate solution in the order  $i = 1, 2, \dots, n$  to also reduce  $(\mathbf{r}_{k+1})^{(i)}$  to zero. However, the component  $(\mathbf{x}_{k+1})^{(i)}$  can be updated with information from the current iteration, stored in the previous components  $(\mathbf{x}_{k+1})^{(1)}, (\mathbf{x}_{k+1})^{(2)}, \dots, (\mathbf{x}_{k+1})^{(i-1)}$ . Hence, Equation 3.7 results in:

$$(\mathbf{x}_{k+1})^{(i)} = \frac{1}{a_{ii}} \left( b^{(i)} - \sum_{j < i} a_{ij} (\mathbf{x}_{k+1})^{(j)} - \sum_{j > i} a_{ij} (\mathbf{x}_k)^{(j)} \right). \quad (3.10)$$

The matrix form of 3.10 is:

$$\mathbf{x}_{k+1} = D^{-1}(\mathbf{b} + L\mathbf{x}_{k+1} + U\mathbf{x}_k), \quad (3.11)$$

or, rearranged:

$$\mathbf{x}_{k+1} = (D - L)^{-1}U\mathbf{x}_k + (D - L)^{-1}\mathbf{b}. \quad (3.12)$$

## 3.4 Krylov Subspace Methods

One might have an initial guess for the solution. If not, the only vector associated with the problem is the right-hand side vector  $\mathbf{b}$ . Without computing any

matrix-vector products, it is natural to choose a multiple of  $\mathbf{b}$  as the first approximation to the solution (Bhaya and Kaszkurewicz, 2006):

$$\mathbf{x}_1 \in \text{span}\{\mathbf{b}\}.$$

Then, one computes the product  $A\mathbf{b}$  and takes the next approximation to be some linear combination of  $\mathbf{b}$  and  $A\mathbf{b}$ :

$$\mathbf{x}_2 \in \text{span}\{\mathbf{b}, A\mathbf{b}\}.$$

This process continues so that the approximation at step  $k$  of the solution satisfies:

$$\mathbf{x}_k \in \text{span}\{\mathbf{b}, A\mathbf{b}, A^2\mathbf{b}, \dots, A^{k-1}\mathbf{b}\}. \quad (3.13)$$

The space represented by (3.13) is called the *Krylov subspace* associated with matrix  $A$  and an initial vector  $\mathbf{b}$ . For a matrix  $A$  and any other initial vector  $\mathbf{v}$ , it is common to use the notation  $\mathcal{K}_k(A, \mathbf{v})$ .

If it turns out that the space (3.13) does not provide a good approximate solution for any moderate values of  $k$ , one might consider modifying the original problem to obtain a better Krylov subspace. A common strategy is to use a *preconditioner*  $M$  to transform the original problem into a new one that is easier to solve. The preconditioned system is given by:

$$M^{-1}A\mathbf{x} = M^{-1}\mathbf{b},$$

and the new Krylov subspace is given by:

$$\text{span}\{M^{-1}\mathbf{b}, (M^{-1}A)M^{-1}\mathbf{b}, (M^{-1}A)^2M^{-1}\mathbf{b}, \dots, (M^{-1}A)^{k-1}M^{-1}\mathbf{b}\}.$$

For the new Krylov subspace to be effective, the preconditioner  $M$  should satisfy some minimal requirements. The matrix  $M$  should be close to  $A$  in some sense, it must be nonsingular, and its inverse  $M^{-1}$  should be cheaper to apply than  $A^{-1}$ .

## 3.5 GMRES Method

While the conjugate gradient method (Hestenes et al., 1952) is usually the preferred choice for symmetric linear systems, the GMRES method (Saad and Schultz, 1986) is currently the most popular Krylov subspace method for solving nonsymmetric ones. At the start, an approximate solution vector  $\mathbf{x}_0$  and an initial residual vector

$\mathbf{r}_0 = \mathbf{b} - A\mathbf{x}_0$  are defined. In the  $k$ -th iteration, the GMRES method returns a solution vector of the form:

$$\mathbf{x}_k = \mathbf{x}_0 + \mathbf{z}_k,$$

where  $\mathbf{z}_k \in \mathcal{K}_k(A, \mathbf{r}_0)$ . To generate  $\mathbf{z}_k$ , an orthogonalization of the base (either using Gram-Schmidt—standard or modified—or Householder) is employed. A matrix  $V_k$  of size  $n \times k$ , whose column vectors are a base of the subspace, and a Hessenberg matrix

$$\tilde{H}_k = \begin{pmatrix} h_{1,1} & h_{1,2} & \cdots & h_{1,k-1} & h_{1,k} \\ h_{2,1} & h_{2,2} & \cdots & h_{2,k-1} & h_{2,k} \\ 0 & h_{3,2} & \cdots & h_{3,k-1} & h_{3,k} \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & \cdots & 0 & h_{k,k-1} & h_{k,k} \\ 0 & \cdots & 0 & 0 & h_{k+1,k} \end{pmatrix}, \quad (3.14)$$

are products of this process. The equation that resumes the Arnoldi process is: where  $\mathbf{z}_k \in \mathcal{K}_k(A, \mathbf{r}_0)$ . To generate  $\mathbf{z}_k$ , an orthogonalization of the basis (using either Gram-Schmidt—standard or modified—or Householder) is employed. A matrix  $V_k$  of size  $n \times k$ , whose column vectors form a basis for the subspace, and a Hessenberg matrix  $\tilde{H}_k$  of size  $(k+1) \times k$  are products of this process. The equation that summarizes the Arnoldi process is:

$$AV_k = V_{k+1}\tilde{H}_k. \quad (3.15)$$

The approximate solution, then, is of the form:

$$\mathbf{x}_k = \mathbf{x}_0 + V_k\mathbf{y}_k, \quad (3.16)$$

where  $\mathbf{y}_k$  minimizes the Euclidean norm of the residual  $\mathbf{r}_k = \mathbf{b} - A\mathbf{x}_k$  at each iteration  $k$ :

$$\begin{aligned} \mathbf{y}_k &= \underset{\mathbf{y}}{\operatorname{argmin}} \|\mathbf{b} - A\mathbf{x}_k\| \\ &= \underset{\mathbf{y}}{\operatorname{argmin}} \|\mathbf{b} - A(\mathbf{x}_0 + \mathbf{V}_k\mathbf{y})\| \\ &= \underset{\mathbf{y}}{\operatorname{argmin}} \|\mathbf{r}_0 - AV_k\mathbf{y}\| \\ &= \underset{\mathbf{y}}{\operatorname{argmin}} \|V_{k+1}(\beta\mathbf{e}_1 - \tilde{H}_k\mathbf{y})\| \\ &= \underset{\mathbf{y}}{\operatorname{argmin}} \|\beta\mathbf{e}_1 - \tilde{H}_k\mathbf{y}\|. \end{aligned} \quad (3.17)$$

where  $\beta = \|\mathbf{r}_0\|$ ,  $\mathbf{e}_1 = [1, 0, 0, \dots, 0]^T$  is the first unit vector.

The minimization process implies that, at each iteration, the residual norms are non-increasing:

$$\|\mathbf{r}_k\| \leq \|\mathbf{r}_{k-1}\|, \quad (3.18)$$

which is a highly desirable property for iterative methods. This feature makes GMRES more robust in terms of convergence than other Krylov subspace methods for nonsymmetric linear systems (Meurant and Tebbens, 2020), such as the bi-conjugate gradient methods BiCG (Fletcher, 1976) or BiCGSTAB (Van der Vorst, 1992), where the residual norms oscillate. Assuming exact arithmetic, GMRES converges before the  $n$ -th iteration, where  $n$  is the dimension of matrix  $A$  (Saad, 2003). However, the cost of computing the orthonormal basis for the Krylov subspace increases with the number of iterations.

### 3.6 Restarting Version of GMRES, or GMRES( $m$ )

The restarted GMRES method or GMRES( $m$ ) (Saad and Schultz, 1986) is a variant that was designed to reduce the computational cost of the original version. The idea is to limit the number of iterations to  $m$ , a user-defined parameter, such that it is much less than the size  $n$  of the coefficient matrix  $A$  ( $m \ll n$ ). After  $m$  iterations, the algorithm restarts using the current residual vector as the new initial guess. This process is repeated until convergence.

The group of  $m$  iterations between restarts is called a *restart cycle*. This process is repeated until the monotonic, non-increasing relative 2-norm of the residual becomes less than a user-defined tolerance  $\epsilon$ . GMRES( $m$ ) approximates the solution to the linear system at the  $j$ -th restart cycle by using the previous residual,  $\mathbf{r}_{j-1} = \mathbf{b} - A\mathbf{x}_{j-1}$ , for the construction of a Krylov subspace  $\mathcal{K}_m(A, \mathbf{r}_{j-1}) = \text{span}\{\mathbf{r}_{j-1}, A\mathbf{r}_{j-1}, \dots, A^{m-1}\mathbf{r}_{j-1}\}$  of dimension  $m$ . The  $j$ -th approximation is built as

$$\mathbf{x}_j = \mathbf{x}_{j-1} + \mathcal{K}_m(A, \mathbf{r}_{j-1}), \quad (3.19)$$

where the index  $m$  denotes that the restarting parameter was set to the value  $m$ . GMRES( $m$ ) obtains an approximate solution that minimizes the 2-norm of the residual  $\mathbf{r}_j$ , i.e.,

$$\min_{\mathbf{x}_j \in \mathbf{x}_{j-1} + \mathcal{K}_m(A, \mathbf{r}_{j-1})} \|\mathbf{b} - A\mathbf{x}_j\|. \quad (3.20)$$

To solve this problem, the Arnoldi process is normally used for obtaining an orthonormal basis for the Krylov subspace. At the  $j$ -th cycle, the first  $m$  steps of this procedure

can be expressed as:

$$AV_m = V_{m+1}\tilde{H}_m, \quad (3.21)$$

where  $V_m \in \mathbb{C}^{n \times m}$  and  $V_{m+1} := [V_m \ \mathbf{v}_{m+1}] \in \mathbb{C}^{n \times (m+1)}$  have orthonormal columns, and  $\tilde{H}_m \in \mathbb{C}^{(m+1) \times m}$  is the upper Hessenberg matrix formed by an upper matrix  $H_m$  of dimension  $m \times m$  and an entry  $h_{m+1,m}$  located at position  $(m+1, m)$ . If the Arnoldi process starts with  $\mathbf{v}_1 = (\frac{1}{\beta})\mathbf{r}_{j-1}$ , where  $\beta = \|\mathbf{r}_{j-1}\|$  then, by construction, the columns of  $V_m$  form an orthogonal basis of the subspace  $\mathcal{K}_m(A, \mathbf{r}_{j-1})$ .

The approximate solution  $\mathbf{x}_j$  is obtained by

$$\mathbf{x}_j = \mathbf{x}_{j-1} + V_m \mathbf{y}_j, \quad (3.22)$$

and the expression (3.20) becomes

$$\min_{\mathbf{y}} \|\mathbf{r}_{j-1} - AV_m \mathbf{y}\|. \quad (3.23)$$

Therefore, the approximate solution  $\mathbf{x}_j$  minimizes the residual norm. One drawback of GMRES( $m$ ) is that it does not preserve orthogonality between the approximation spaces generated at successive restarts (Saad, 2003; Baker et al., 2005). As a consequence, GMRES( $m$ ) generally exhibits a slower rate of convergence compared to its full version. The problem becomes evident when the residual norm between consecutive cycles does not decrease sufficiently, i.e., when there is a sharp slowdown in the convergence rate. A more severe situation occurs when the rate of convergence stagnates; in other words, when the reduction in the residual norms at consecutive cycles is very small, i.e.,  $\|\mathbf{r}_j\| \approx \|\mathbf{r}_{j-1}\|$ . Therefore, the residual vectors tend to point in nearly the same direction at the end of every restart cycle, i.e.,  $\angle(\mathbf{r}_j, \mathbf{r}_{j-1}) \approx 0$ , indicating that GMRES( $m$ ) did not introduce a significant decrease in the residual norm between consecutive cycles (Nuñez et al., 2018).

### 3.7 Strategies for GMRES( $m$ )

A drawback of the GMRES( $m$ ) method is that it can lead to slow convergence or even stagnation (Joubert, 1994). The problem of improving the convergence rate and avoiding stagnation has been addressed using several strategies, which are mentioned below.

- Augmenting the search subspace with a subspace containing information vec-

tors from previous restart cycles. (Baker et al., 2005) This strategy accelerates GMRES( $m$ ) but is not helpful when the method stagnates. A similar strategy consists of including spectral information related to matrix  $A$  in the search subspace. (Morgan, 1995) This is done by adding a few eigenvectors associated with the smallest eigenvalues of matrix  $A$ , which can help avoid stagnation. However, computing the eigenvectors can be expensive and is not always feasible.

- Since it is difficult to know *a priori* how many iterations are needed for each restart cycle, the restart parameter  $m$  can be adaptively changed between cycles to avoid problems that can arise from a poor choice of its value. The selection of an appropriate controller for  $m$  is currently an active line of research. (Nuñez et al., 2018; Cabral et al., 2020; Duvenbeck et al., 2025)
- Using a preconditioner  $M$  to transform the original problem into a new one that preserves some information from the original linear system but is easier to solve (Pearson and Pestana, 2020). Despite its advantages, choosing the preconditioner matrix is a non-trivial task.

### 3.7.1 LGMRES( $m, l$ )

The motivation for LGMRES( $m, l$ ) (Baker et al., 2005) is to address the alternating behavior observed in the GMRES( $m$ ) residual at consecutive cycles, which leads to a deterioration in the convergence. The solver incorporates approximations to the error in the current search subspace. The error approximation at the  $j$ -th restart cycle is defined using the approximate solutions from previous cycles as

$$\varphi_j = \mathbf{x}_{j-1} - \mathbf{x}_{j-2} \quad (3.24)$$

and  $\varphi_j \equiv 0$  for  $j < 1$ . This error approximation vector is used for augmenting the search subspace  $\mathcal{K}_m(A, \mathbf{r}_{j-1})$  at the next cycle. Note that  $\varphi_j \in \mathcal{K}_m(A, \mathbf{r}_{j-2})$ . Therefore, this error approximation  $\varphi_j$  in some sense represents the space  $\mathcal{K}_m(A, \mathbf{r}_{j-2})$  generated in the previous cycle but subsequently discarded in the restarting process. Hence, it is a natural choice for enriching the next approximation space  $\mathcal{K}_m(A, \mathbf{r}_{j-1})$ .

The augmented approximation space  $\mathcal{S} = \mathcal{K}_m(A, \mathbf{r}_{j-1}) \cup \text{span}\{\varphi_k\}$ ,  $k = \{(j - l), \dots, (j - 1)\}$  has dimension  $s = m + l$ . This method, instead of using  $\mathcal{K}_m(A, \mathbf{r}_{j-1})$  in Equation (3.19), employs the subspace  $\mathcal{S}$ . The matrix  $V_{s+1}$  is the  $n \times (s + 1)$  orthonormal matrix, whose first  $m + 1$  columns are the Arnoldi vectors, and the last  $l$  columns result from orthogonalizing the  $l$  most recent error approximation vectors ( $\varphi_k$ ,

$k = (j - l), \dots, (j - 1)$ ) against the previous Arnoldi vectors.  $W_s$  is an  $n \times s$  matrix whose first  $m$  columns are the orthogonalized Arnoldi vectors and the last  $l$  columns are the  $l$  most recent error approximations (typically normalized so that all columns have unit norm). Then the new relationship at the  $j$ -th cycle is

$$AW_s = V_{s+1}\tilde{H}_s, \quad (3.25)$$

which holds for LGMRES( $m, l$ ). Here,  $\tilde{H}_s$  denotes an  $(s + 1) \times s$  upper Hessenberg matrix. In practice,  $l \ll m$  according to (Baker et al., 2005)  $l \leq 3$  is a good option for LGMRES( $m, l$ ). Similar to expression (3.16), the approximate solution at the  $j$ -th cycle is

$$\mathbf{x}_j = \mathbf{x}_{j-1} + W_s \mathbf{y}_j, \quad (3.26)$$

with  $W_s = [\mathbf{v}_1 \ \mathbf{v}_2 \ \dots \ \mathbf{v}_m \ \varphi_{j-l} \ \dots \ \varphi_{j-1}]$  and  $\mathbf{y}_j$  minimize the residual norm  $\|\mathbf{r}_j\|$ .

It is important to note that LGMRES is not helpful when one of the following situations occurs (Cabral et al., 2020):

- (a) when GMRES( $m$ ) skip angles ( $\angle(\mathbf{r}_j, \mathbf{r}_{j-2})$ ) are not small;
- (b) when GMRES( $m$ ) sequential angles ( $\angle(\mathbf{r}_j, \mathbf{r}_{j-1})$ ) vary greatly from cycle to cycle;
- (c) when GMRES( $m$ ) converges in a small number of iterations; or
- (d) when GMRES( $m$ ) skip angles and sequential angles are near zero (indicating stagnation).

### 3.7.2 GMRES-E( $m, d$ )

The strategy for search enhancement consists of enriching the subsequent Krylov subspaces by introducing approximate eigenvectors associated with the problematic eigenvalues. In practice, the approximate eigenvectors are the harmonic Ritz vectors associated with the harmonic Ritz values computed at each cycle (Morgan, 1995).

At the  $j$ -th cycle, the harmonic Ritz value  $\tilde{\lambda}_k$  with the associated harmonic Ritz vector  $\vartheta_k = W_s \mathbf{g}_k$ , where  $\mathbf{g}_k \in \mathbb{C}^s$ , with respect to the subspace  $AK_s(A, \mathbf{r}_{j-1})$  satisfies the following expression

$$(A\vartheta_k - \tilde{\lambda}_k \vartheta_k) \perp AK_s(A, \mathbf{r}_0), \quad (3.27)$$

which implies,

$$\begin{aligned} (AW_s)^*(AW_s g_k - \tilde{\lambda}_k W_s g_k) &= 0, \\ W_s^* A^* AW_s g_k &= \tilde{\lambda}_k W_s^* A^* W_s g_k. \end{aligned} \quad (3.28)$$

Here the  $M^*$  denotes the conjugate transpose of a matrix  $M$ . Using the Arnoldi relation,

$$AW_s = V_{s+1}\tilde{H}_s \quad (3.29)$$

where  $W_s$  is a  $n \times s$  matrix, its first  $m$  columns are Arnoldi's vectors and the last  $d$  corresponds to the approximate eigenvectors  $\vartheta_k$  and  $V_{s+1}$  is the  $n \times (s+1)$  orthonormal matrix whose first  $m+1$  columns are the Arnoldi vectors and last  $d$  columns result from orthogonalizing the  $d$  approximation eigenvectors ( $\vartheta_k, k = 1, \dots, d$ ) against the previous columns of Arnoldi's vectors.  $\tilde{H}_s$  is an  $(s+1) \times s$  upper Hessenberg matrix with its elements constructed by the new orthogonalization process. At the  $j$ -th cycle, the reduced generalized eigenvalue problem is defined using the expression (3.28) and the new Arnoldi relation (3.29).

$$\tilde{H}_s^* \tilde{H}_s g_k = \tilde{\lambda}_k H_s^* g_k, \quad (3.30)$$

where  $\tilde{H}_s^*$  is the Hermitian of the upper Hessenberg matrix of dimension  $s \times (s+1)$  and  $H_s^*$  is the hermitian of the Hessenberg matrix whose dimension is  $s \times s$ . Usually the value of  $s$  is much less than  $n$  (Morgan, 1995). These eigenvalues are known as *harmonic Ritz values*, and are the roots of the GMRES residual polynomial (Simoncini and Szyld, 2007). The eigenvectors associated with these harmonic Ritz values are called *harmonic Ritz vectors*. Some  $\mathbf{g}_k$ , associated with the smallest  $\tilde{\lambda}_k$  are needed to deflate the smallest eigenvalues and thereby improve convergence (Morgan, 1995, 2002). Since the harmonic Ritz vectors are only used at a specific cycle, they must be recomputed at each cycle. For this reason, only the residuals are stored to be reused in the next cycle.

At the end of the  $j$ -th restart cycle, GMRES-E( $m, d$ ) seeks the approximate solution  $\mathbf{x}_j$  of the form

$$\mathbf{x}_j = \mathbf{x}_{j-1} + W_s \mathbf{y}_j, \quad (3.31)$$

such that  $\mathbf{y}_j$  is obtained by solving the following minimization problem

$$\|\mathbf{r}_j\| = \|\mathbf{b} - A\mathbf{x}_j\| = \min_{\mathbf{y}_j \in \mathbb{C}^{s+1}} \|\beta \mathbf{e}_1 - \tilde{H}_s \mathbf{y}_j\|, \quad (3.32)$$

where  $\beta = \|\mathbf{r}_{j-1}\|$ ,  $\mathbf{e}_1 = [1, 0, 0, \dots, 0]^T$  is the first unit vector. The sequence of residual norms of the GMRES-E( $m, d$ ) has the property of being non-increasing, but can not guarantee convergence (Morgan, 1995; Niu and Lu, 2012).

### 3.7.3 PD-GMRES( $m_j$ )

This variant is an adaptive extension of GMRES( $m$ ) that updates the restart parameter  $m$  (Nuñez et al., 2018). As it is difficult to know a priori how to choose  $m$  to ensure a fast rate of convergence, PD-GMRES uses a Proportional-Derivative controller to determine  $m$  based on the history of the last relative residual norms. Such a controller is a particular case of the well-known family of Proportional-Integrative-Derivative (PID) controllers in control theory (Ogata, 2003).

The main PD rule for updating  $m$  for the  $j$ -th restart cycle is:

$$m_j = m_{j-1} + \left\lfloor \alpha_P \frac{\|\mathbf{r}_j^{(j)}\|}{\|\mathbf{r}_{j-1}^{(j-1)}\|} + \alpha_D \frac{\|\mathbf{r}_j^{(j)}\| - \|\mathbf{r}_{j-2}^{(j-2)}\|}{2\|\mathbf{r}_{j-1}^{(j-1)}\|} \right\rfloor, \quad (3.33)$$

where  $\alpha_P, \alpha_D \in \mathbb{R}$  and  $\lfloor (\cdot) \rfloor$  denotes the integer part.

## 3.8 The KrySBAS Library

Notably, many of these presented solvers are not currently available to the scientific computing community. Therefore, there is a genuine motivation to develop a software package that offers better reproducibility of experiments and serves as a fast prototyping tool for the development of new GMRES( $m$ ) variants. KrySBAS (*Krylov Subspace-Based Adaptive Solver*), a free open-source software written for MATLAB®/GNU Octave, is a result of that effort. It contains a collection of adaptive solvers based on partially explored features of the Krylov subspaces. KrySBAS is available at [www.github.com/nidtec-una/krysbas-dev](http://www.github.com/nidtec-una/krysbas-dev).

KrySBAS is thoroughly tested with unit and functional tests using MOxUnit within automated workflows. Test coverage metrics are automatically generated by MOxCov and CodeCov. Documentation of the package is generated using Sphinx and available through *readthedocs* at <https://krysbas-dev.readthedocs.io>. As of this writing, the solvers included in release v0.3.1 of KrySBAS are: LGMRES( $m, l$ ), GMRES-E( $m, d$ ), and PD-GMRES( $m_j$ ).

In KrySBAS, the user is encouraged to invoke the `help` command directly from the command-line interface (CLI). An extract of the output is shown in Listing 3.1.

```

1 >> help pd_gmres
2 PD-GMRES Proportional-Derivative GMRES(m)
3
4 Description:
5 -----
6
```

```

7  pd_gmres is a modified implementation of the restarted Generalized
8  Minimal Residual Error or GMRES(m) [1], performed by using a
9  proportional-derivative control-inspired law to update adaptively the
10 restarting parameter m before each restart. This implementation follows
11 closely the one presented in [2].
12
13 Signature:
14 -----
15
16 [x, flag, relresvec, kdvec, time] = pd_gmres(A, b, ...
17      mInitial, mMinMax, mStep, tol, maxit, xInitial, alphaPD)
18
19
20 Input Parameters:
21 -----
22
23 A:          n-by-n matrix
24             Left-hand side of the linear system  $Ax = b$ .
25
26 b:          n-by-1 vector
27             Right-hand side of the linear system  $Ax = b$ .
28
29 mInitial:   int, optional
30             Initial restart parameter (similar to 'restart' in MATLAB).
31             If empty, not given, or equal to n, then mInitial is not
32             used and the full unrestarted gmres algorithm with
33             maxit = min(n, 10) is employed. Note that we require
34              $1 \leq mInitial \leq n$ .
35
36 mMinMax:    2-by-1 vector, optional
37             Minimum and maximum values of the restart parameter m.
38             Default is [1; n]. We require
39              $1 \leq mMinMax(1) < mMinMax(2) \leq n$ . Note that for large
40             matrices (e.g.,  $> 600$ ), the default value  $mMinMax(2) = n$ 
41             might require large computational resources.
42
43 mStep:      int, optional
44             Step size for increasing the restart parameter m when
45              $m < mMinMax(1)$ . Default is 1 if  $n \leq 10$  and 3 otherwise.
46
47 tol:       float, optional
48             Tolerance error threshold for the relative residual norm.
49             Default is  $1e-6$ .
50
51 maxit:     int, optional
52             Maximum number of outer iterations.
53
54 xInitial:   n-by-1 vector, optional
55             Vector of initial guess. Default is zeros(n, 1).
56
57 alphaPD:    2-by-1 vector, optional
58             Proportional and derivative coefficients from the
59             proportional-derivative controller. Default is [-3, 5],
60             following Ref. [2].
61

```

```

62  Output parameters:
63  -----
64
65  x:          n-by-1 vector
66             Approximate solution of the linear system.
67
68  flag:       boolean
69             1 if the algorithm has converged, 0 otherwise.
70
71  relresvec:  (1 up to maxit)-by-1 vector
72             Vector of relative residual norms of every outer iteration
73             (cycles). The last relative residual norm is simply given
74             by relresvec(end).
75
76  kdvec:      (1 up to maxit)-by-1 vector
77             Vector of restart parameter values. In case the unrestarted
78             algorithm is invoked, kdvec = NaN.
79
80  time:       scalar
81             Computational time in seconds.

```

Listing 3.1: Invoking the command-line help in KrySBAS.

Listing 3.2 shows an example of usage for KrySBAS. Lines 2 to 4 load the coefficient matrix  $A$  and right-hand side vector  $b$  from `sherman4.mat`, a benchmark file from the SuiteSparse matrix collection (Kolodziej et al., 2019) which is pre-loaded in the `data` folder of KrySBAS. Lines 7 to 14 of the script initializes parameters including the size of the coefficient matrix  $n$ , the initial value of  $m_j$  (`mInitial = 30`) and its adaptive bounds for all cycles (`mMinMax = [10 100]`), and controller coefficients  $[\alpha_P, \alpha_D]$  for the PD rule (`alphaPD = [1 1]`), and then calls the `pd_gmres` function with a tolerance of  $10^{-9}$  and a zero vector as initial guess. The output at line 17 includes the approximate solution vector  $x$ , a convergence `flag`, the history vector of the relative residual norms per cycle (`relresvec`), the history vector of the restart parameter  $m_j$  used per cycle (`kdvec`), and the total runtime, hence providing a diagnostic of the solver’s adaptive behavior.

```

1  % Load A and b
2  load('data/sherman4.mat', 'Problem');
3  A = Problem.A;
4  b = Problem.b;
5
6  % Setup PD-GMRES
7  n = size(b, 1);          % Problem size
8  mInitial = 30;           % Initial restart parameter m
9  mMinMax = [10 100];     % Minimum and maximum limits for m
10 mStep = 3;               % Step size if m exceeds limits
11 tol = 1e-12;            % Tolerance for convergence of GMRES

```

```

12 maxit = 1000;           % Maximum number of restarts
13 xInitial = zeros(n, 1); % Initial guess
14 alphaPD = [1 1];       % Proportional & derivative coefficients
15
16 % Call PD-GMRES
17 [x, flag, relresvec, kdvec, time] = ...
18     pd_gmres(A, b, mInitial, mMinMax, mStep, tol, maxit, ...
19             xInitial, alphaPD);

```

Listing 3.2: KrySBAS sample code for the methods.

## 3.9 Preconditioning

The convergence of Krylov subspace methods can be improved by using preconditioners (Pearson and Pestana, 2020). Roughly speaking, a preconditioner is a matrix  $M$  whose inverse  $M^{-1}$  approximates, in some sense,  $A^{-1}$ .

For nonsymmetric linear systems (Ghai et al., 2019), a preconditioner may be applied to either the left or the right of  $A$ . With a left preconditioner, instead of solving  $A\mathbf{x} = \mathbf{b}$ , one solves

$$M^{-1}A\mathbf{x} = M^{-1}\mathbf{b}. \quad (3.34)$$

For a right preconditioner, one solves the linear system

$$AM^{-1}\mathbf{w} = \mathbf{b}, \quad (3.35)$$

where  $\mathbf{w}$  is the solution of the preconditioned system, and then  $\mathbf{x} = M^{-1}\mathbf{w}$ .

Choosing the most appropriate preconditioning technique is a difficult task, dependent of the problem, and often hand-crafted (Mathew et al., 2007; Sarkis et al., 2008). In the other hand, Jacobi, Gauss-Seidel, its generalization the Successive Over-Relaxation method (SOR), and incomplete LU (ILU), are examples of simple preconditioners. They are relatively easy to implement, require virtually no setup time (at least in the serial case), and are sometimes fairly effective. Therefore, they are often good choices when a preconditioner needs to be implemented from scratch.

### 3.9.1 Iteration Matrices and Preconditioning

The Jacobi and Gauss-Seidel iterations are of the form

$$\mathbf{x}_{k+1} = F\mathbf{x}_k + \mathbf{f}, \quad (3.36)$$

where  $F$  is a matrix that depends on the iteration. In the particular cases of Jacobi and Gauss-Seidel,

$$F_J(A) = I - D^{-1}A, \quad (3.37)$$

$$F_{GS}(A) = I - (D + L)^{-1}A, \quad (3.38)$$

respectively.

The iteration (3.36) can be viewed as a technique for solving the system:

$$(I - F)\mathbf{x} = \mathbf{f}. \quad (3.39)$$

Since  $F$  has the form  $I - M^{-1}A$ , this system can be rewritten as follows:

$$\begin{aligned} [I - (I - M^{-1}A)]\mathbf{x} &= M^{-1}\mathbf{b} \\ \Rightarrow M^{-1}A\mathbf{x} &= M^{-1}\mathbf{b}. \end{aligned} \quad (3.40)$$

The last linear system, which has the same solution as the original system, is the *left-preconditioned* linear system, and  $M$  is the *preconditioning matrix* or *preconditioner*.

### 3.9.2 Left Preconditioning

In the case of Equation (3.2), it follows that  $K = M^{-1}$  is already related to the preconditioning technique. Consider again the partitioning

$$A = D + (-L) + (-U), \quad (3.41)$$

as already seen in Figure 3.1, where  $D$  is the diagonal of  $A$ ,  $L$  is the strictly lower triangular part, and  $U$  is the strictly upper triangular part.

The Jacobi preconditioner is given by  $M = D$ , i.e.,

$$M_J = D. \quad (3.42)$$

The SOR preconditioner is given by

$$M_{SOR} = \frac{1}{\omega}D + L, \quad (3.43)$$

for a relaxation parameter  $\omega$ . When  $\omega = 1$ , SOR reduces to Gauss-Seidel. An optimal

choice of  $\omega$  in SOR is problem-dependent; for that reason, the matrix preconditioner

$$M_{GS} = D + L. \quad (3.44)$$

has been chosen for experimentation because it is parameter-free.

The objective of preconditioning is to lower the condition number of the problem. In the case of left preconditioning, we hope that  $\kappa(M^{-1}A) \leq \kappa(A)$ , where  $\kappa(\cdot)$  is the condition number of a matrix (see Appendix A). A possible disadvantage of preconditioning is that it may increase the number of nonzero entries in the matrix, thus losing the advantages of sparsity. However, in many cases, the benefits of preconditioning far outweigh the drawbacks, especially for large-scale problems where convergence speed is critical.

### 3.9.3 Incomplete $LU$ Preconditioning

An incomplete  $LU$  factorization (abbreviated as  $ILU$ ) of a matrix is a sparse approximation of the  $LU$  factorization,

$$A \approx L_i U_i, \quad (3.45)$$

where  $L_i$  and  $U_i$  are far sparser than those in the  $LU$  factorization of  $A$ .

More accurate  $ILU$  preconditioners require more memory, to the extent that the running time of the algorithm may increase, even though the total number of iterations of the iterative method decreases. Consequently, there is a cost/accuracy trade-off that users must evaluate when choosing this preconditioning technique.

# Chapter 4

## Experiments, Results and Discussion

This chapter presents the numerical experiments conducted to evaluate the performance of various iterative solvers on linear systems arising from high-order mimetic discretization of the 2D Poisson equation. The main topics covered are: the general setup of the experiments; the definition of the benchmark problems with known analytical solutions; the numerical convergence analysis that verifies that the mimetic operators achieve their theoretical orders of accuracy and identifies the practical limits to discretization, guiding the selection of problems for further testing sections; the comparison of performance between the selected Krylov subspace methods without and with preconditioning; and finally a brief examination of the behavior of the adaptive restart parameter, showing how the search subspace can be adjusted with base on the problem's difficulty.

### 4.1 Experimental Setup

The numerical experiments were run on a computer with an Intel Core i5-12450 processor and 16 GB of RAM, using MATLAB® R2022a for Windows 11. The code is available at <https://github.com/gusepinola/mimetic-2dpoisson>. The Mimetic Operators Library Enhanced (MOLE), release v1.0, was employed for the discretization. This process generates large, sparse linear systems, which were solved using the Krylov Subspace-Based Adaptive Solvers (KrySBAS) library, release v0.3.1. The MOLE and KrySBAS libraries are available at <https://github.com/csru-sdsu/mole> and <https://github.com/nidtec-una/krysbas-dev>, respectively. The following iterative methods were applied: the built-in MATLAB® implementation of GMRES( $m$ ) (Saad and Schultz, 1986), along with GMRES-E( $m, d$ ) (Morgan, 1995), LGMRES( $m, l$ ) (Baker et al., 2005), and PD-GMRES( $m_j$ ) (Nuñez et al., 2018) from the KrySBAS li-

brary. The numerical solution was obtained by discretizing a two-dimensional domain  $\Omega$  (see Figure 2.3) into  $m_x \times m_y$  cells, with  $m_x = m_y \in \{32, 64, 128, 256, 512\}$ . The initial guess was  $\mathbf{u}_0 = \mathbf{0}$ , and the stopping criterion was a relative residual norm of  $\|\mathbf{r}_j\|/\|\mathbf{r}_0\| < 10^{-12}$ . An experiment was labeled as non-convergent (NC) if the solver failed to converge within 1000 restarts or if the total execution time exceeded one hour (3600 s). All reported execution times are the average of 10 runs.

Table 4.1 summarizes the parameters used for the iterative methods. The restart parameter for GMRES( $m$ ) is set to  $m = 30$ , a common choice for large linear systems (i.e., systems with dimensions much larger than 30) and the default value in software packages such as *PETSc* (Balay et al., 2025). For LGMRES, the number of error approximation vectors is set to  $l = 3$ , a value recommended as "good" by the *SciPy* package (SciPy Developers, 2025). The number of approximate eigenvectors for GMRES-E was also set to  $d = 3$ ; this choice is justified because even a few vectors can deflate the critical eigenvalues that hinder convergence, achieving significant improvements without excessive storage and orthogonalization costs (Morgan, 1995). Finally, the parameters  $\alpha_P$  and  $\alpha_D$  for PD-GMRES are chosen to be positive to allow the restart parameter  $m_j$  to increase during a cycle if necessary (Cabral et al., 2020).

Table 4.1: Parameters considered for each method.

| Methods           | Parameters                                          |
|-------------------|-----------------------------------------------------|
| GMRES( $m$ )      | $m=30$ .                                            |
| PD-GMRES( $m_j$ ) | $m_{min}=30, m_{max}=100, \alpha_P=1, \alpha_D=1$ . |
| LGMRES( $m, l$ )  | $m=27, l=3$ .                                       |
| GMRES-E( $m, d$ ) | $m=27, d=3$ .                                       |

## 4.2 Problems

Consider the two-dimensional Poisson equation with a Dirichlet boundary condition:

$$\begin{aligned} \nabla \cdot (-\nabla u(x, y)) &= f(x, y) \quad \text{in} \quad \Omega = (0, 1) \times (0, 1), \\ u &= g(x, y) \quad \text{on} \quad \partial\Omega. \end{aligned}$$

**Problem A:** The homogeneous Poisson equation ( $f(x, y) = 0$ ) with

$$g(x, y) = \begin{cases} 0 & y = 0 \\ \sin(\pi x) & y = 1 \\ 0 & x = 0 \\ 0 & x = 1, \end{cases}$$

whose exact solution is

$$u(x, y) = \frac{\sin(\pi x) \sinh(\pi y)}{\sinh(\pi)}.$$

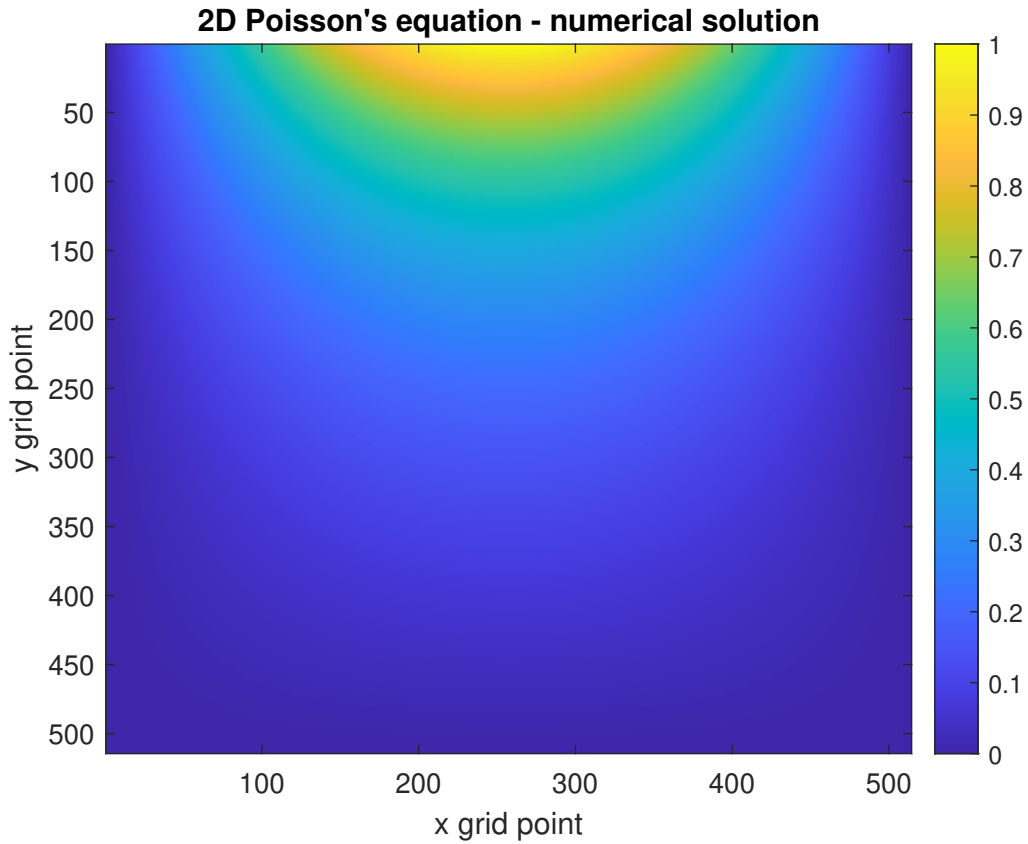


Figure 4.1: Numerical solution of the 2D Poisson equation, Problem A.

**Problem B:** The non-homogeneous Poisson equation with source term

$$f(x, y) = -2\pi^2 \sin(\pi x) \sin(\pi y),$$

whose exact solution is

$$u(x, y) = \sin(\pi x) \sin(\pi y).$$

The above solution satisfies homogeneous boundary conditions  $u = 0$  on  $\partial\Omega$

because:

$$x = 0 \text{ or } y = 0 \text{ or } x = 1 \text{ or } y = 1 \Rightarrow \sin(\pi x) \sin(\pi y) = 0.$$

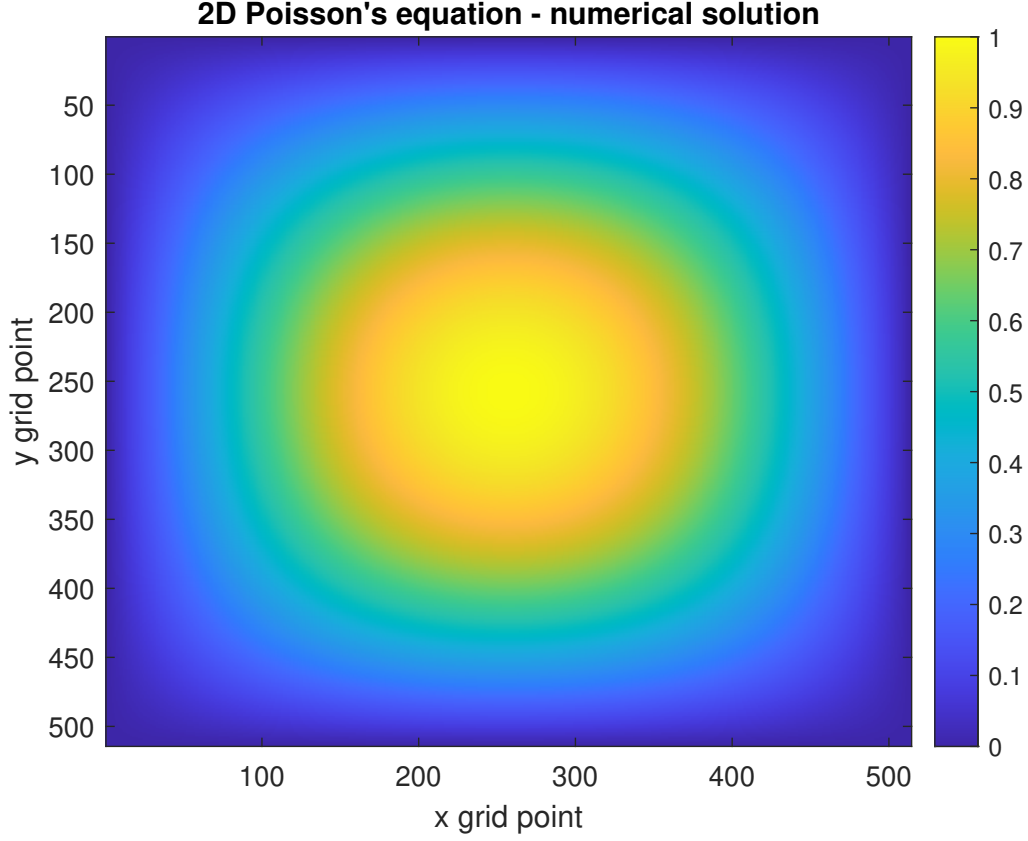


Figure 4.2: Numerical solution of the 2D Poisson equation, Problem B.

The discretization process returns the linear system of equations  $L\mathbf{u}^h = \mathbf{b}$ , with the same coefficient matrix  $L \in \mathbb{R}^{n \times n}$ , where  $n$  is the number of discretization points,  $n = (m_x + 2) \times (m_y + 2)$ , and vectors  $\mathbf{u}^h, \mathbf{b} \in \mathbb{R}^n$ . Before applying the Krylov subspace iterative methods, different preconditioning strategies were used: Jacobi, Gauss-Seidel and ILU (specifically, ILUTP: ILU with threshold dropping and pivoting, with a drop tolerance of  $10^{-6}$ ). Metrics of the resulting matrices  $L$ , for different orders of accuracy for mimetic operators  $k = 2, 4, 6, 8$  and number of 1D points  $m_x$ , are shown in Table 4.2.

Table 4.2: List of test problems in the Poisson problem with different orders of accuracy ( $k$ ), discretization sizes ( $m_x = m_y$  in the studied case) and classic preconditioners. %nnz( $L$ ): percentage of nonzero elements of matrix  $L$ , condest( $L$ ): condition number of matrix  $L$ , OOM: "Out of Memory" exception. > 1 h: the preconditioning time exceeded 1 hour.

| Parameters |       | No prec.       | Jacobi      |                | Gauss-Seidel |                | ILU         |                |
|------------|-------|----------------|-------------|----------------|--------------|----------------|-------------|----------------|
| $k$        | $m_x$ | condest( $L$ ) | %nnz( $L$ ) | condest( $L$ ) | %nnz( $L$ )  | condest( $L$ ) | %nnz( $L$ ) | condest( $L$ ) |
| 2          | 32    | 7.54E+02       | 0.39        | 6.03E+02       | 25.32        | 2.66E+02       | 82.52       | 1.00           |
| 2          | 64    | 3.02E+03       | 0.11        | 2.41E+03       | 25.26        | 1.06E+03       | 91.12       | 1.03           |
| 2          | 128   | 1.21E+04       | 0.03        | 9.66E+03       | 25.17        | 4.23E+03       | 95.74       | 1.18           |
| 2          | 256   | 4.83E+04       | 0.007       | 3.86E+04       | OOM          |                | > 1 h       |                |
| 2          | 512   | 1.93E+05       | 0.002       | 1.55E+05       | OOM          |                | > 1 h       |                |
| 4          | 32    | 1.01E+03       | 1.00        | 8.43E+02       | 30.96        | 3.65E+02       | 91.75       | 1.00           |
| 4          | 64    | 4.04E+03       | 0.28        | 3.37E+03       | 28.41        | 6.27E+24       | 96.03       | 1.02           |
| 4          | 128   | 1.62E+04       | 0.07        | 1.35E+04       | 26.82        | 6.19E+46       | 98.19       | 1.22           |
| 4          | 256   | 6.46E+04       | 0.02        | 5.40E+04       | OOM          |                | > 1 h       |                |
| 4          | 512   | 2.59E+05       | 0.005       | 2.16E+05       | OOM          |                | > 1 h       |                |
| 6          | 32    | 1.14E+03       | 1.00        | 1.08E+03       | 35.71        | 4.92E+02       | 88.51       | 1.00           |
| 6          | 64    | 4.58E+03       | 0.28        | 3.37E+03       | 31.13        | 1.96E+03       | 94.01       | 1.02           |
| 6          | 128   | 1.83E+04       | 0.07        | 1.35E+04       | 28.28        | 2.70E+18       | 96.94       | 1.21           |
| 6          | 256   | 7.33E+04       | 0.02        | 5.40E+04       | OOM          |                | > 1 h       |                |
| 6          | 512   | 2.93E+05       | 0.005       | 2.78E+05       | OOM          |                | > 1 h       |                |
| 8          | 32    | 1.87E+03       | 1.00        | 1.72E+03       | 40.21        | 7.96E+02       | 93.97       | 1.00           |
| 8          | 64    | 7.50E+04       | 0.28        | 3.37E+03       | 33.78        | 3.17E+03       | 96.93       | 1.02           |
| 8          | 128   | 3.00E+04       | 0.07        | 1.35E+04       | 29.72        | 1.27E+04       | 98.45       | 1.21           |
| 8          | 256   | 1.20E+05       | 0.02        | 5.40E+04       | OOM          |                | > 1 h       |                |
| 8          | 512   | 4.80E+05       | 0.005       | 4.42E+05       | OOM          |                | > 1 h       |                |

### 4.3 Numerical Convergence Analysis

In two dimensions, the cell size  $h$  is given by  $h = \max\{h_x, h_y\}$ , where  $h_x$  and  $h_y$  are the step sizes in the  $x$  and  $y$  directions, respectively. In the current setup,  $h_x = h_y$ , hence the notation  $h$  is enough. This section verifies that the numerical solution converges to the exact analytical solution as  $h$  approaches zero. This analysis is crucial for two main reasons: (i) it validates the implementation by confirming that the observed convergence rate matches the theoretical order of accuracy of the mimetic operators, and (ii) it identifies the practical limits of the method, particularly the point at which round-off errors begin to dominate discretization errors on very fine meshes, causing the convergence rate to degrade.

The asymptotic truncation error  $E_h$ , in terms of  $h$ , is estimated by:

$$E_h = Ch^k + O(h^{k+1}), \quad (4.1)$$

where  $k$  is the order of the error and  $C$  is a convergence-rate constant. Given two refinement levels  $h_i$  and  $h_{i+1}$ ,  $i = 1, 2, \dots$ , the order of convergence  $k$  can be estimated (LeVeque, 2007) by the Equation

$$k \approx \frac{\log(E_{h_{i+1}}/E_{h_i})}{\log(h_{i+1}/h_i)}. \quad (4.2)$$

In this experiment, we verify the convergence rate by using the infinite norm:

$$E_\infty = \|\mathbf{u}^h - \mathbf{u}^*\|_\infty = \max_{(x,y) \in \mathcal{G}} |u_{i,j}^h - u^*(x_i, y_j)|. \quad (4.3)$$

where  $\mathbf{u}^*$  is the projection of the analytical solution at the points of the centers of the cells and the outer faces of the cells at the boundary of  $\Omega$  and  $\mathbf{u}^h$  is the numerical solution returned for the points of  $\mathcal{G}$ . The infinity norm was chosen for being computationally less expensive than the Euclidean norm, especially when computing fine discretizations, but by the Equivalence of Norms theorem one can use indistinctly either norm,

Figure 4.3 and Table 4.3 present the results using Problem A as an example; the results for Problem B were similar. For  $k = 2$  and  $k = 4$ , the estimated order of convergence is consistent with the theoretical values. For  $k = 6$  and  $k = 8$ , the methods initially exhibit expected convergence rates, but as the grid is refined, the observed order degrades. This suggests that for these higher-order methods on finer meshes, the total error is dominated by round-off error rather than discretization error (LeVeque, 2007; Chapra, 2011). Consequently, one have to be careful when high-order mimetic discretizations are employed in fine grids. It is often better to use a lower-order operator (e.g.,  $k = 4$ ) or a coarser grid, which can provide similar accuracy at a lower cost. Therefore, for the subsequent experiments, we only consider  $m_x = 32, 64$  for  $k = 6$ , and only  $m_x = 32$  for  $k = 8$ .

## 4.4 Comparative Analysis of Iterative Methods

Figure 4.4 shows that the convergence rate of the standard GMRES( $m$ ) iterative method depends on the problem size. While effective for small problems, its performance deteriorates significantly as the problem size increases (i.e., as the mesh is refined). For larger problems, it requires many more iterations to achieve the desired accuracy and often fails to converge within the execution time limit. This behavior highlights the need for methods that scale more effectively with problem size.

Table 4.3: Numerical convergence analysis for Problem A. The error  $E_\infty$  is measured in the infinity norm, and the estimated order of convergence is calculated using Equation (4.2) for successive grid refinements.

| $h = 1/m_x$ | $k = 2$    |              | $k = 4$    |              |
|-------------|------------|--------------|------------|--------------|
|             | $E_\infty$ | Order (est.) | $E_\infty$ | Order (est.) |
| 1/16        | 2.81E-03   | —            | 1.63E-05   | —            |
| 1/32        | 7.54E-04   | 1.90         | 9.57E-07   | 4.09         |
| 1/64        | 1.94E-04   | 1.96         | 5.72E-08   | 4.06         |
| 1/128       | 4.94E-05   | 1.97         | 3.49E-09   | 4.03         |
| 1/256       | 1.24E-05   | 1.99         | 2.14E-10   | 4.02         |
| 1/512       | 3.12E-06   | 1.99         | 1.07E-11   | 4.32         |

| $h = 1/m_x$ | $k = 6$    |              | $k = 8$    |              |
|-------------|------------|--------------|------------|--------------|
|             | $E_\infty$ | Order (est.) | $E_\infty$ | Order (est.) |
| 1/16        | 9.42E-08   | —            | 1.51E-09   | —            |
| 1/32        | 8.93E-10   | 6.72         | 7.04E-12   | 8.48         |
| 1/64        | 9.03E-12   | 6.62         | 9.58E-14   | 6.19         |
| 1/128       | 6.82E-13   | 3.72         | 2.98E-13   | -1.63        |
| 1/256       | 1.80E-12   | -1.40        | 1.18E-12   | -1.98        |
| 1/512       | 7.33E-12   | -2.02        | 4.64E-12   | -1.97        |

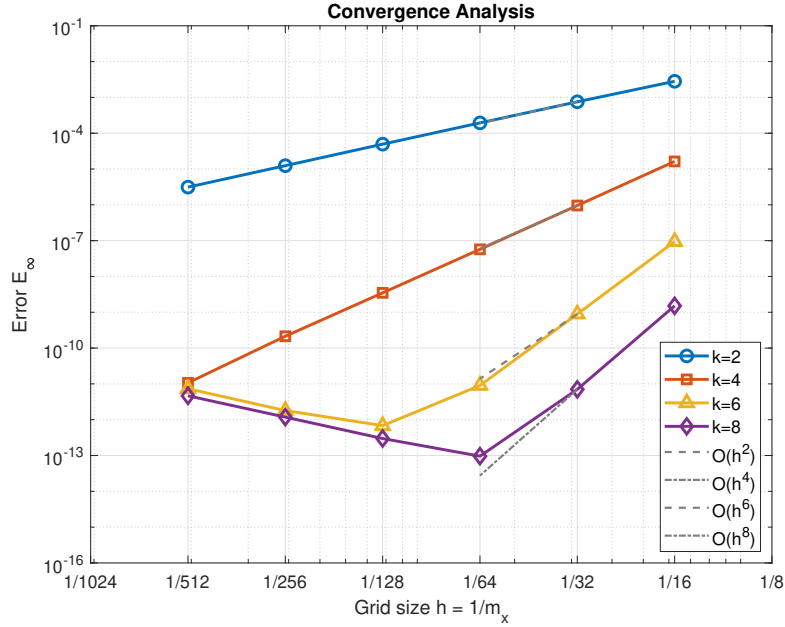


Figure 4.3: Log-log plot of the error norm  $E_\infty$  vs. the cell size  $h$ . The slopes of the dotted lines enclose the estimated orders of accuracy.

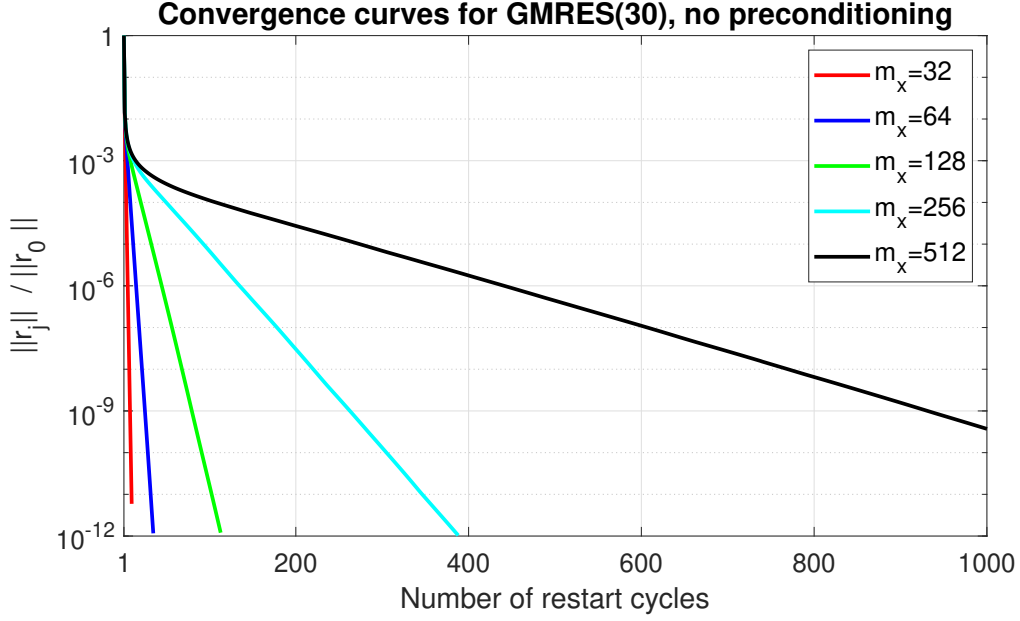


Figure 4.4: Convergence curves for the built-in MATLAB®/Octave function GMRES(30) for Problem A with  $k = 2$  and different discretization sizes  $h = 1/m_x$ . The plot shows the relative residual norm versus the number of restart cycles. For  $m_x = 512$ , GMRES(30) does not reach convergence.

For Problem A, Table 4.4 compares the iterative methods without preconditioning. For small problems (e.g.,  $m_x = 32$ ), all methods converge in milliseconds, making standard GMRES( $m$ ) sufficient. As the problem size increases, its efficiency drops, and methods with augmented Krylov subspaces or adaptive restart parameters become preferable. For the largest problem ( $m_x = 512$ ), standard GMRES( $m$ ) failed to converge. Among the other methods, LGMRES consistently provided the best balance between execution time and iteration count. While GMRES-E often required the fewest iterations, its computational overhead resulted in longer execution times. PD-GMRES offered a reasonable compromise but typically required more iterations than LGMRES and GMRES-E.

Table 4.4: Metrics for iterative methods, Problem A: average computational time in seconds and number of iterations required for  $\|\mathbf{r}_j\|/\|\mathbf{r}_0\| < 10^{-12}$ . For cases that do not reach convergence before 1000 restarts, the method is stopped and the time is denoted as NC. Lowest execution times and number of iterations are highlighted in green.

| $k$ | $m_x$ | No preconditioning |       |                   |       |          |       |          |       |
|-----|-------|--------------------|-------|-------------------|-------|----------|-------|----------|-------|
|     |       | GMRES( $m$ )       |       | PD-GMRES( $m_j$ ) |       | LGMRES   |       | GMRES-E  |       |
|     |       | Time               | Iter. | Time              | Iter. | Time     | Iter. | Time     | Iter. |
| 2   | 32    | 0.0164             | 291   | 0.0347            | 501   | 0.0294   | 330   | 0.0276   | 180   |
| 2   | 64    | 0.4354             | 1029  | 0.3935            | 1020  | 0.1843   | 480   | 0.1796   | 390   |
| 2   | 128   | 5.2278             | 3388  | 2.5934            | 1786  | 1.1593   | 930   | 1.2301   | 930   |
| 2   | 256   | 36.6954            | 11630 | 18.7563           | 4082  | 4.6839   | 1560  | 8.8936   | 2610  |
| 2   | 512   | NC                 |       | 496.526           | 9809  | 60.468   | 2970  | 187.2195 | 8520  |
| 4   | 32    | 0.0161             | 360   | 0.0371            | 572   | 0.0238   | 330   | 0.0248   | 210   |
| 4   | 64    | 0.5520             | 1321  | 0.5351            | 1330  | 0.2463   | 630   | 0.1996   | 450   |
| 4   | 128   | 7.5292             | 4562  | 3.9278            | 2236  | 1.4588   | 1050  | 1.6831   | 1140  |
| 4   | 256   | 59.5492            | 16104 | 24.923            | 4606  | 7.4842   | 2130  | 13.725   | 3330  |
| 4   | 512   | NC                 |       | 936.046           | 10387 | 121.0641 | 3750  | 405.516  | 11580 |
| 6   | 32    | 0.0220             | 471   | 0.0575            | 586   | 0.0324   | 390   | 0.0317   | 240   |
| 6   | 64    | 0.6796             | 1588  | 0.6094            | 1395  | 0.2741   | 660   | 0.2270   | 480   |
| 8   | 32    | 0.0282             | 582   | 0.0655            | 667   | 0.0342   | 450   | 0.0354   | 270   |

Problem B is a special case where the continuous problem is an eigenvalue problem for the Laplacian operator ( $\Delta u = \lambda u$ ). The discretized system is  $L\mathbf{u} = \mathbf{f}$ , where  $\mathbf{f}$  is a multiple of the exact solution,  $\mathbf{f} = \lambda\mathbf{u}$ . Since the initial guess is  $\mathbf{u}_0 = \mathbf{0}$ , the initial residual is  $\mathbf{r}_0 = \mathbf{f} - L\mathbf{u}_0 = \mathbf{f}$ . This makes  $\mathbf{r}_0$  an approximation of an eigenvector of the matrix  $L$ . In exact arithmetic, GMRES would converge in one iteration because the solution lies in the initial Krylov subspace,  $\mathcal{K}_1(L, \mathbf{r}_0) = \text{span}\{\mathbf{r}_0\} = \text{span}\{\mathbf{u}\}$ . In finite-precision arithmetic, GMRES( $m$ ) is expected to converge in a few iterations for this problem.

The results in Table 4.5 confirm this rapid convergence, showing significantly fewer iterations compared to Problem A. For higher-order discretizations ( $k = 6, 8$ ), the number of iterations for a given grid size generally decreases. This happens, most likely, because the higher-order operators produce a discrete problem that more accurately approximates the continuous eigenvalue problem, making the initial residual  $\mathbf{r}_0$  numerically closer to a true eigenvector of  $L$ . For the lower-order methods ( $k = 2, 4$ ), the iteration count still increases with problem size, driven by the worsening condition number of the matrix.

Table 4.5: Metrics for iterative methods, Problem B: average computational time in seconds and number of iterations required for  $\|\mathbf{r}_j\|/\|\mathbf{r}_0\| < 10^{-12}$ . Lowest execution times and number of iterations are highlighted in green.

| $k$ | $m_x$ | No preconditioning |       |                   |       |         |       |         |       |
|-----|-------|--------------------|-------|-------------------|-------|---------|-------|---------|-------|
|     |       | GMRES( $m$ )       |       | PD-GMRES( $m_j$ ) |       | LGMRES  |       | GMRES-E |       |
|     |       | Time               | Iter. | Time              | Iter. | Time    | Iter. | Time    | Iter. |
| 2   | 32    | 0.0087             | 49    | 0.0130            | 90    | 0.0140  | 90    | 0.0254  | 90    |
| 2   | 64    | 0.1057             | 238   | 0.1305            | 356   | 0.0794  | 210   | 0.0564  | 150   |
| 2   | 128   | 1.1850             | 701   | 0.7555            | 621   | 0.4299  | 360   | 0.3248  | 70    |
| 2   | 256   | 5.4898             | 1679  | 2.8235            | 950   | 2.2013  | 750   | 1.8345  | 570   |
| 2   | 512   | 59.8271            | 5579  | 83.4571           | 2002  | 23.4342 | 870   | 44.2523 | 1530  |
| 4   | 32    | 0.0038             | 27    | 0.0044            | 30    | 0.0045  | 30    | 0.0045  | 30    |
| 4   | 64    | 0.0681             | 121   | 0.0605            | 162   | 0.0545  | 150   | 0.0434  | 120   |
| 4   | 128   | 0.5364             | 305   | 0.3222            | 276   | 0.2710  | 210   | 0.2159  | 180   |
| 4   | 256   | 1.8380             | 493   | 1.2362            | 397   | 1.0811  | 330   | 1.0874  | 300   |
| 4   | 512   | 2.0393             | 141   | 3.6932            | 152   | 4.1699  | 180   | 3.7301  | 150   |
| 6   | 32    | 0.0030             | 27    | 0.0049            | 30    | 0.0042  | 30    | 0.0031  | 30    |
| 6   | 64    | 0.0309             | 63    | 0.0319            | 90    | 0.0451  | 120   | 0.0355  | 90    |
| 8   | 32    | 0.0025             | 29    | 0.0026            | 30    | 0.0029  | 30    | 0.0029  | 30    |

## 4.5 Effectiveness in Preconditioning

In the next series of experiments, the iterative methods were tested with Jacobi, Gauss-Seidel and ILU preconditioners. The performance of the preconditioned GMRES( $m$ ), PD-GMRES( $m_\ell$ ), LGMRES, and GMRES-E methods was then evaluated. As shown in Table 4.2, preconditioning generally reduced the matrix condition number, suggesting the problems would be easier to solve. Tables 4.6 through 4.9 present results for a subset of these experiments, including only the cases that respected the theoretical order of accuracy as determined in Section 4.3. Overall, Jacobi preconditioning was the most effective in terms of total execution time, likely because it is inexpensive to apply and the coefficient matrix is diagonally dominant.

Table 4.6: Comparative summary of numerical methods with different preconditioners. Order of accuracy:  $k = 2$ . Average computational time in seconds and number of iterations required for  $\|\mathbf{r}_j\|/\|\mathbf{r}_0\| < 10^{-12}$ . The lowest execution times per  $m_x$  are highlighted in green.

| $k = 2$ |          |       |              |       |                      |       |        |       |         |       |
|---------|----------|-------|--------------|-------|----------------------|-------|--------|-------|---------|-------|
| $m_x$   | Precond. | Prec. | GMRES( $m$ ) |       | PD-GMRES( $m_\ell$ ) |       | LGMRES |       | GMRES-E |       |
|         |          |       | Time         | Iter. | Time                 | Iter. | Time   | Iter. | Time    | Iter. |
| 32      | None     | -     | 0.02         | 291   | 0.03                 | 501   | 0.03   | 330   | 0.03    | 180   |
|         | Jacobi   | <0.01 | 0.01         | 220   | 0.03                 | 268   | 0.02   | 240   | 0.03    | 180   |
|         | G.S.     | 0.03  | 0.08         | 122   | 0.08                 | 152   | 0.10   | 180   | 0.11    | 120   |
|         | ILU      | 0.07  | 0.09         | 3     | 0.12                 | 30    | 0.09   | 30    | 0.11    | 30    |
| 64      | None     | -     | 0.43         | 1029  | 0.39                 | 1020  | 0.18   | 480   | 0.18    | 390   |
|         | Jacobi   | <0.01 | 0.32         | 767   | 0.32                 | 781   | 0.15   | 390   | 0.13    | 300   |
|         | G.S.     | 0.37  | 2.59         | 383   | 3.10                 | 492   | 2.26   | 360   | 1.99    | 240   |
|         | ILU      | 1.32  | 1.46         | 4     | 2.03                 | 30    | 1.48   | 30    | 1.95    | 30    |
| 128     | None     | -     | 5.22         | 3388  | 2.59                 | 1786  | 1.16   | 930   | 1.23    | 930   |
|         | Jacobi   | <0.01 | 3.85         | 2461  | 2.40                 | 1671  | 0.92   | 750   | 0.90    | 690   |
|         | G.S.     | 6.63  | 82.33        | 1098  | 90.83                | 1249  | 54.38  | 720   | 60.53   | 570   |
|         | ILU      | 87.96 | 88.41        | 5     | 94.72                | 30    | 88.43  | 30    | 94.50   | 30    |

Table 4.7: Comparative summary of numerical methods with different preconditioners. Order of accuracy:  $k = 4$ . Average computational time in seconds and number of iterations required for  $\|\mathbf{r}_j\|/\|\mathbf{r}_0\| < 10^{-12}$ . The lowest execution times per  $m_x$  are highlighted in green.

| $k = 4$ |          |        |        |              |        |                      |        |        |        |         |       |
|---------|----------|--------|--------|--------------|--------|----------------------|--------|--------|--------|---------|-------|
| $m_x$   | Precond. | Prec.  | Time   | GMRES( $m$ ) |        | PD-GMRES( $m_\ell$ ) |        | LGMRES |        | GMRES-E |       |
|         |          |        |        | Time         | Iter.  | Time                 | Iter.  | Time   | Iter.  | Time    | Iter. |
| 32      | None     | -      | 0.02   | 360          | 0.04   | 572                  | 0.02   | 330    | 0.03   | 210     |       |
|         | Jacobi   | <0.01  | 0.01   | 301          | 0.03   | 424                  | 0.03   | 300    | 0.03   | 180     |       |
|         | G.S.     | 0.03   | 0.08   | 162          | 0.09   | 184                  | 0.06   | 210    | 0.07   | 150     |       |
|         | ILU      | 0.23   | 0.23   | 4            | 0.26   | 30                   | 0.23   | 30     | 0.25   | 30      |       |
| 64      | None     | -      | 0.55   | 1321         | 0.53   | 1330                 | 0.24   | 630    | 0.20   | 450     |       |
|         | Jacobi   | <0.01  | 0.42   | 975          | 0.42   | 989                  | 0.21   | 480    | 0.17   | 360     |       |
|         | G.S.     | 0.37   | 2.59   | 508          | 4.13   | 595                  | 2.48   | 360    | 2.55   | 270     |       |
|         | ILU      | 9.29   | 9.56   | 5            | 10.08  | 30                   | 9.57   | 30     | 10.04  | 30      |       |
| 128     | None     | -      | 7.53   | 4562         | 3.92   | 2236                 | 1.45   | 1050   | 1.68   | 1140    |       |
|         | Jacobi   | <0.01  | 5.55   | 3203         | 3.07   | 1835                 | 1.15   | 840    | 1.24   | 840     |       |
|         | G.S.     | 6.64   | 123.63 | 1609         | 88.72  | 1157                 | 68.66  | 870    | 79.93  | 720     |       |
|         | ILU      | 389.95 | 390.51 | 6            | 396.71 | 30                   | 390.66 | 30     | 396.43 | 30      |       |

Table 4.8: Comparative summary of numerical methods with different preconditioners. Order of accuracy:  $k = 6$ . Average computational time in seconds and number of iterations required for  $\|\mathbf{r}_j\|/\|\mathbf{r}_0\| < 10^{-12}$ . The lowest execution times per  $m_x$  are highlighted in green.

| $k = 6$ |          |       |       |              |       |                      |       |        |       |         |       |
|---------|----------|-------|-------|--------------|-------|----------------------|-------|--------|-------|---------|-------|
| $m_x$   | Precond. | Prec. | Time  | GMRES( $m$ ) |       | PD-GMRES( $m_\ell$ ) |       | LGMRES |       | GMRES-E |       |
|         |          |       |       | Time         | Iter. | Time                 | Iter. | Time   | Iter. | Time    | Iter. |
| 32      | None     | -     | 0.02  | 471          | 0.06  | 586                  | 0.03  | 390    | 0.03  | 240     |       |
|         | Jacobi   | <0.01 | 0.02  | 374          | 0.03  | 572                  | 0.02  | 330    | 0.03  | 210     |       |
|         | G.S.     | 0.03  | 0.11  | 189          | 0.11  | 225                  | 0.12  | 240    | 0.10  | 150     |       |
|         | ILU      | 0.45  | 0.46  | 4            | 0.49  | 30                   | 0.46  | 30     | 0.49  | 30      |       |
| 64      | None     | -     | 0.68  | 1588         | 0.61  | 1395                 | 0.27  | 660    | 0.22  | 480     |       |
|         | Jacobi   | <0.01 | 0.52  | 1132         | 0.50  | 1144                 | 0.25  | 570    | 0.19  | 390     |       |
|         | G.S.     | 0.45  | 4.41  | 537          | 4.59  | 605                  | 2.94  | 390    | 3.07  | 300     |       |
|         | ILU      | 18.72 | 18.79 | 4            | 19.26 | 30                   | 18.76 | 30     | 19.26 | 30      |       |

Table 4.9: Comparative summary of numerical methods with different preconditioners. Order of accuracy:  $k = 8$ . Average computational time in seconds and number of iterations required for  $\|\mathbf{r}_j\|/\|\mathbf{r}_0\| < 10^{-12}$ . The lowest execution times per  $m_x$  are highlighted in green.

| $k = 8$ |          |       |      |              |       |                      |       |        |       |         |       |
|---------|----------|-------|------|--------------|-------|----------------------|-------|--------|-------|---------|-------|
| $m_x$   | Precond. | Prec. | Time | GMRES( $m$ ) |       | PD-GMRES( $m_\ell$ ) |       | LGMRES |       | GMRES-E |       |
|         |          |       |      | Time         | Iter. | Time                 | Iter. | Time   | Iter. | Time    | Iter. |
| 32      | None     | -     | 0.03 | 582          | 0.07  | 667                  | 0.03  | 450    | 0.03  | 270     |       |
|         | Jacobi   | <0.01 | 0.02 | 475          | 0.04  | 547                  | 0.03  | 360    | 0.04  | 150     |       |
|         | G.S.     | 0.03  | 0.13 | 238          | 0.17  | 337                  | 0.15  | 270    | 0.13  | 180     |       |
|         | ILU      | 0.55  | 0.61 | 4            | 0.64  | 30                   | 0.62  | 30     | 0.64  | 30      |       |

For finer discretizations, only Jacobi preconditioning was feasible; the other preconditioners exceeded either memory or time limits, as shown in Table 4.2. The combination of Jacobi preconditioning with subspace enrichment methods (LGMRES and GMRES-E) achieved a significant speedup compared to standard GMRES( $m$ ) without preconditioning, as detailed in Figure 4.5.

Table 4.10: Comparative summary of numerical methods for fine-discretization problems ( $m_x = 256, 512$ ): no preconditioning vs. Jacobi preconditioning. Average computational time in seconds and number of iterations required for  $\|\mathbf{r}_j\|/\|\mathbf{r}_0\| < 10^{-12}$ . The lowest execution times per  $m_x$  are highlighted in green.

| $k$ | $m_x$ | Precond. | GMRES( $m$ ) |       | PD-GMRES( $m_\ell$ ) |       | LGMRES |       | GMRES-E |       |
|-----|-------|----------|--------------|-------|----------------------|-------|--------|-------|---------|-------|
|     |       |          | Time         | Iter. | Time                 | Iter. | Time   | Iter. | Time    | Iter. |
| 2   | 256   | None     | 36.70        | 11630 | 18.76                | 4082  | 4.68   | 1560  | 8.89    | 2610  |
| 2   | 256   | Jacobi   | 27.83        | 8714  | 10.43                | 2631  | 3.26   | 1080  | 6.53    | 1890  |
| 2   | 512   | None     | NC           |       | 878.47               | 12506 | 89.16  | 2970  | 271.84  | 8520  |
| 2   | 512   | Jacobi   | NC           |       | 497.09               | 7706  | 63.70  | 2190  | 193.49  | 6180  |
| 4   | 256   | None     | 59.55        | 16104 | 24.92                | 4606  | 7.48   | 2130  | 13.73   | 3330  |
| 4   | 256   | Jacobi   | 43.36        | 11426 | 13.97                | 2939  | 6.09   | 1680  | 10.17   | 2400  |
| 4   | 512   | None     | NC           |       | 936.05               | 10387 | 121.06 | 3750  | 405.52  | 11580 |
| 4   | 512   | Jacobi   | NC           |       | 804.52               | 12206 | 81.22  | 2670  | 265.57  | 7860  |

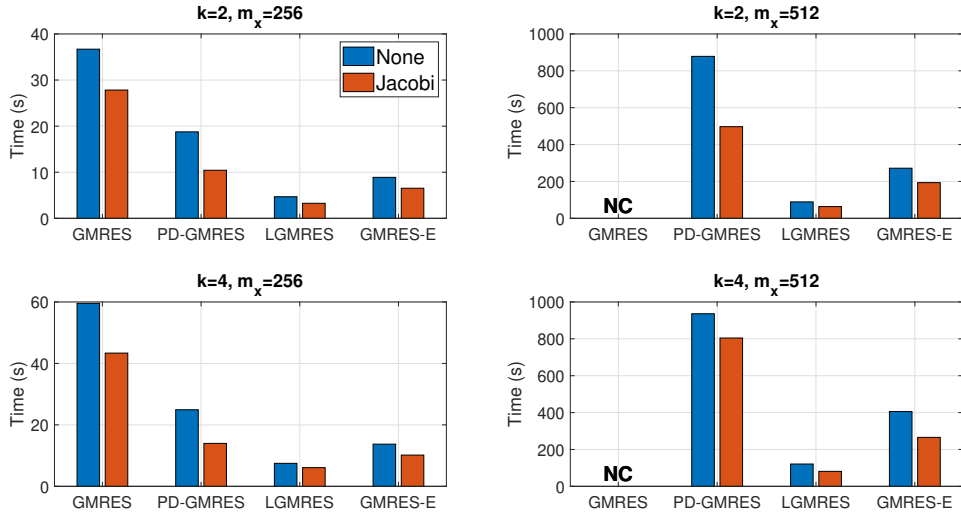


Figure 4.5: Comparison of execution times (in seconds) between no preconditioning and Jacobi preconditioning, for the selected problems listed in Table 4.10.

The Gauss-Seidel preconditioner often reduced the number of iterations compared to Jacobi (as seen in Figure 4.6), but its total computational cost was generally higher. This is due to the increased matrix fill-in (see Table 4.2), which leads to more floating-point operations per iteration.

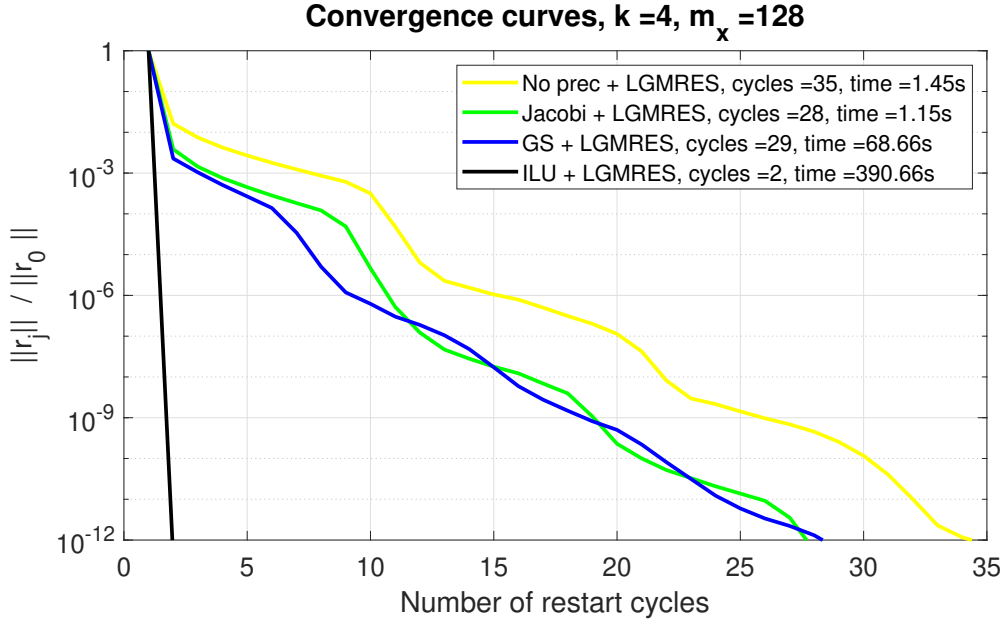


Figure 4.6: Relative residual norms for Problem A with  $k = 4$  and  $m_x = 128$ : no preconditioner, Jacobi, Gauss-Seidel and ILU, plus LGMRES.

The ILU preconditioner dramatically reduced the matrix condition number to  $\kappa \approx 1$  and, consequently, the number of iterations required for convergence. However, this came at the cost of a substantial increase in matrix fill-in, as shown in Table 4.2. The high cost of computing and applying the ILU factors dominated the total solution time, nullifying the benefit of fewer iterations. As a result, ILU was the slowest preconditioning strategy, with all iterative methods showing similarly long execution times.

## 4.6 Adapting the Restart Parameter $m$

This section provides evidence on the ability of the PD-rule, i.e. the Equation (3.33) from PD-GMRES( $m_j$ ), to adapt the restart parameter  $m$  for each problem. As shown in Figure 4.7, the rule maintains a small, constant  $m$  when convergence is stable (e.g., for  $m_x = 32, 64, 128$ ). As the problem becomes more challenging (i.e., as the mesh is refined), the rule increases  $m$  to improve the convergence rate. The controller prevents excessive growth by enforcing an upper limit,  $m_{max}$ , which constrains memory usage by allowing for more restart cycles rather than an overly large Krylov subspace.

If the restart parameter  $m$  is changed at the  $j$ -th restart cycle, that is,  $m_j$ , there were savings of time and memory when comparing with the built-in GMRES( $m$ ) method, especially previous to preconditioning (see Table 4.4). However, under the

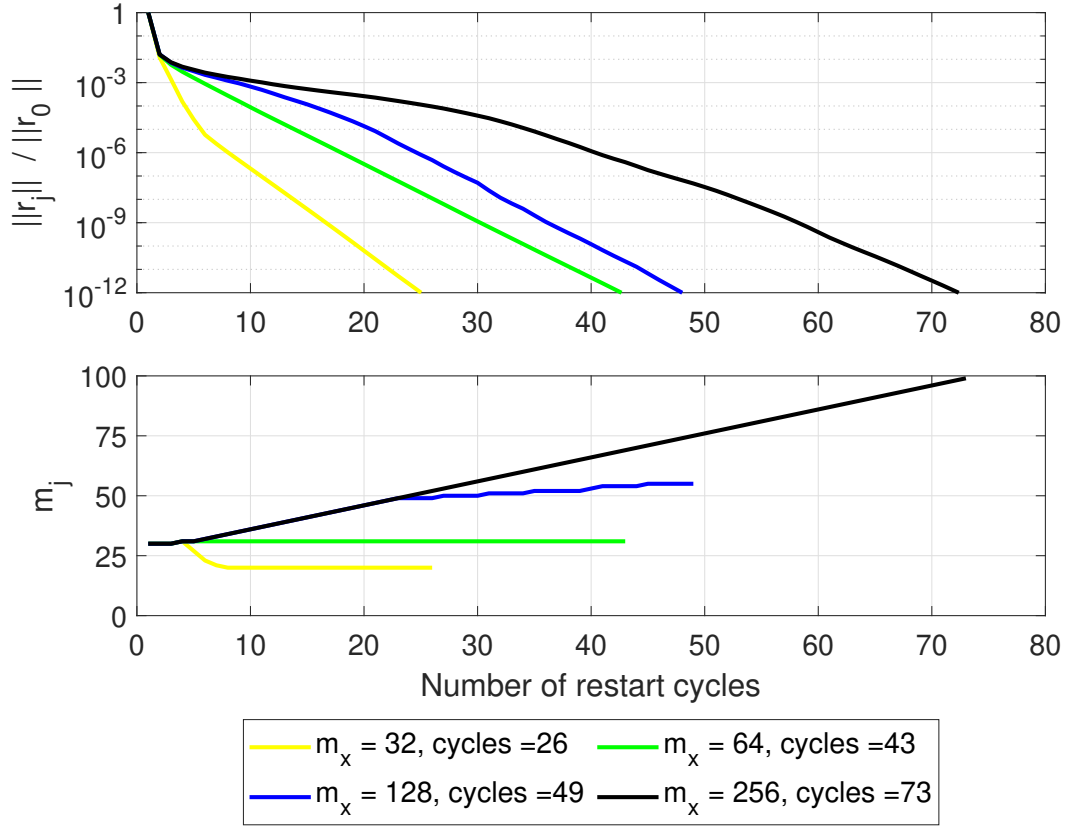


Figure 4.7: Evolution of the restart parameter  $m$  and the relative residual norm for PD-GMRES( $m_j$ ) with  $m_{min} = 1$ ,  $m_{max} = 100$ ,  $\alpha_P = 1$ , and  $\alpha_D = 1$ , problem A with  $k = 4$  and  $m_x = 32, 64, 128, 256$ .

parameters  $\alpha_P, \alpha_D$  chosen for the PD controller, there is still no substantial enhancement when comparing with either GMRES-E or LGMRES, which suggests that the enhancement of the search subspace may be enough. Whether the choice of the appropriate parameters  $\alpha_P, \alpha_D$  and/or the combination with subspace enrichment yields greater savings or not, remains an open question.

# Chapter 5

## Conclusions and Future Work

In this thesis, the high-order mimetic operator framework for the discretization of the Poisson equation and its numerical solution with a combination of different preconditioning techniques and Krylov subspace-based iterative methods have been presented.

Fundamental relationships, such as the Gauss Divergence Theorem, form the basis for representing physical conservation properties. Due to the limited applicability of analytical solutions for complex geometries, numerical methods, particularly finite difference methods, are a necessity. This work explored the use of a staggered grid, which spatially separates scalar and vector variables to better preserve the physical properties of the problem. Mimetic discretization methods were used to construct discrete analogues of the continuous divergence and gradient operators. These operators accurately reflect key integral properties and conservation laws, thereby providing a robust framework for numerical solutions that "mimic" the behavior of the original continuous systems.

Krylov subspace methods, with a particular emphasis on the Generalized Minimal Residual Method (GMRES) and its restarted variant, GMRES( $m$ ), were described. Recognizing the limitations of GMRES( $m$ ), the thesis introduced several enhanced solvers from the literature —LGMRES( $m, l$ ), GMRES-E( $m, d$ ), and PD-GMRES( $m_j$ )— which address issues of slow convergence and stagnation by augmenting the search space, enriching it with eigenvectors, or adaptively controlling the restart parameter. The implementation of these methods was achieved successfully in the KrySBAS software library for MATLAB®/Octave, now available for the scientific software community.

Convergence analysis of the experiments returns that the numerical solution converges to the analytical solution as the grid spacing approaches zero in agreement

with the choice of the correct order of accuracy of the mimetic operator, and it identifies the practical limits where round-off errors begin to dominate discretization errors. A trade-off between high order and size of discretization must be made: using both higher-order operators and finer meshes simultaneously is not correct in terms of accuracy.

Without preconditioning, the convergence rate can be slow, and the classical GMRES( $m$ ) method may fail to meet the requirements. The alternatives to GMRES( $m$ ) were found to be more robust in the majority of cases. A restart parameter  $m_j$  that can be adapted per cycle  $j$  by a Proportional-Derivative (PD) controller may be advantageous due to the relatively lower costs of PD-GMRES( $m_j$ ) compared to GMRES( $m$ ). Nevertheless, for the selected test cases, the best performance was ultimately achieved with subspace enrichment methods.

For the selected problems, the Jacobi preconditioner achieved a good trade-off between simplicity, iteration reduction, and low total time (i.e., the sum of preconditioning and linear-system solving times). This makes it a suitable choice for situations requiring low computational costs. The Gauss-Seidel preconditioner reduces the number of restarts but has a high computational cost. The ILU preconditioner achieves a drastic reduction in the number of cycles, but its high total time cost makes it infeasible in this context.

Some opportunities for future study are:

- The exploration of high-order mimetic operators in other mathematical models.
- A search for optimal parameters for the iterative methods and preconditioning techniques.
- The combination of both subspace enrichment and adaptive restart parameters in the same iterative method.
- The design and implementation of other preconditioning techniques (right preconditioning, block preconditioning) that might take advantage of the structure of the generated linear system of equations.
- The exploration of parallel implementations of the methods.

# References

- Arfken, G. B., Weber, H. J., and Harris, F. E. (2011). *Mathematical methods for physicists: a comprehensive guide*. Academic Press. 5
- Baker, A. H., Jessup, E. R., and Manteuffel, T. (2005). A technique for accelerating the convergence of restarted GMRES. *SIAM Journal on Matrix Analysis and Applications*, 26(4):962–984. 2, 35, 36, 37, 45
- Balay, S., Abhyankar, S., Adams, M. F., Benson, S., Brown, J., Brune, P., Buschelman, K., Constantinescu, E. M., Dalcin, L., Dener, A., Eijkhout, V., Faibussowitsch, J., Gropp, W. D., Hapla, V., Isaac, T., Jolivet, P., Karpeev, D., Kaushik, D., Knepley, M. G., Kong, F., Kruger, S., May, D. A., McInnes, L. C., Mills, R. T., Mitchell, L., Munson, T., Roman, J. E., Rupp, K., Sanan, P., Sarich, J., Smith, B. F., Zampini, S., Zhang, H., and Zhang, J. (2025). *KSPGMRESSetRestart — PETSc 3.23.6 documentation*. Accessed: September 22, 2025. 46
- Barrett, R., Berry, M., Chan, T. F., Demmel, J., Donato, J., Dongarra, J., Eijkhout, V., Pozo, R., Romine, C., and Van der Vorst, H. (1994). *Templates for the solution of linear systems: building blocks for iterative methods*. SIAM. 28
- Bazan, C., Abouali, M., Castillo, J., and Blomgren, P. (2011). Mimetic finite difference methods in image processing. *Computational & Applied Mathematics*, 30:701–720. 16
- Bhaya, A. and Kaszkurewicz, E. (2006). *Control perspectives on numerical algorithms and matrix problems*. SIAM. ix, 27, 28, 29, 30, 32
- Brzenski, J. and Castillo, J. E. (2023). Solving Navier–Stokes with mimetic operators. *Computers & Fluids*, 254:105817. 16
- Cabral, J. C., Schaerer, C. E., and Bhaya, A. (2020). Improving GMRES(m) using an adaptive switching controller. *Numerical Linear Algebra with Applications*, 27(5):e2305. 36, 37, 46

- Castillo, J. E. and Grone, R. (2003). A matrix analysis approach to higher-order approximations for divergence and gradients satisfying a global conservation law. *SIAM Journal on Matrix Analysis and Applications*, 25(1):128–142. 1, 15, 16, 17, 19
- Castillo, J. E. and Miranda, G. F. (2013). *Mimetic discretization methods*. CRC Press. 5, 23, 24
- Chapra, S. (2011). *Applied Numerical Methods with MATLAB for Engineers and Scientists*. McGraw Hill. x, 50, 67, 68
- Corbino, J. and Castillo, J. E. (2020). High-order mimetic finite-difference operators satisfying the extended Gauss divergence theorem. *Journal of Computational and Applied Mathematics*, 364:112326. 1, 15, 17, 23, 24
- Corbino, J., Dumett, M. A., and Castillo, J. E. (2024). MOLE: Mimetic Operators Library Enhanced. *Journal of Open Source Software*, 9(99):6288. ix, 1, 25, 26
- Demmel, J. W. (1997). *Applied Numerical Linear Algebra*. SIAM. 9
- Dumett, M. A. and Castillo, J. E. (2023). Mimetic Analogs of Vector Calculus Identities. Technical Report CSRCR2023-01, Computational Science Research Center, San Diego State University. 3, 20, 22
- Dumett, M. A. and Castillo, J. E. (2024). General framework for mimetic differences. Technical Report CSRCR2024-07, Computational Science Research Center, San Diego State University. ix, 10
- Duvenbeck, L., Riethmüller, C., and Rohde, C. (2025). Data-driven geometric parameter optimization for PD-GMRES. 36
- Fletcher, R. (1976). Conjugate gradient methods for indefinite systems. In Watson, G. A., editor, *Numerical Analysis*, pages 73–89, Berlin, Heidelberg. Springer Berlin Heidelberg. 34
- Ford, W. (2014). *Numerical Linear Algebra with Applications: Using MATLAB and Octave*. Academic Press. 70, 71
- Ghai, A., Lu, C., and Jiao, X. (2019). A comparison of preconditioned Krylov subspace methods for large-scale nonsymmetric linear systems. *Numerical Linear Algebra with Applications*, 26(1):e2215. 42
- Greenbaum, A. (1997). *Iterative methods for solving linear systems*. SIAM. 27, 28

- Hestenes, M. R., Stiefel, E., et al. (1952). Methods of conjugate gradients for solving linear systems. *Journal of research of the National Bureau of Standards*, 49(6):409–436. 9, 32
- Joubert, W. (1994). On the convergence behavior of the restarted GMRES algorithm for solving nonsymmetric linear systems. *Numerical linear algebra with applications*, 1(5):427–447. 35
- Kolodziej, S. P., Aznaveh, M., Bullock, M., David, J., Davis, T. A., Henderson, M., Hu, Y., and Sandstrom, R. (2019). The SuiteSparse Matrix Collection website interface. *Journal of Open Source Software*, 4(35):1244. 41
- LeVeque, R. J. (2007). *Finite difference methods for ordinary and partial differential equations: steady-state and time-dependent problems*. SIAM. 50
- Mathew, T. P., Sarkis, M., and Schaerer, C. E. (2007). Analysis of block matrix preconditioners for elliptic optimal control problems. *Numerical Linear Algebra with Applications*, 14(4):257–279. 42
- Meurant, G. and Tebbens, J. D. (2020). Krylov methods for nonsymmetric linear systems. *Cham: Springer*. 34
- Meyer, C. D. (2001). *Matrix analysis and applied linear algebra*. SIAM. 69, 70
- Morgan, R. B. (1995). A restarted GMRES method augmented with eigenvectors. *SIAM Journal on Matrix Analysis and Applications*, 16(4):1154–1171. 2, 36, 37, 38, 45, 46
- Morgan, R. B. (2002). GMRES with deflated restarting. *SIAM Journal on Scientific Computing*, 24(1):20–37. 38
- Niu, Q. and Lu, L. (2012). Restarted GMRES method augmented with the combination of harmonic Ritz vectors and error approximations. *International Journal of Mathematical and Computational Sciences*, 6. 38
- Núñez, R. C., Schaerer, C. E., and Bhaya, A. (2018). A proportional-derivative control strategy for restarting the GMRES(m) algorithm. *Journal of Computational and Applied Mathematics*, 337:209–224. 2, 35, 36, 39, 45
- Ogata, K. (1995). *Discrete-time control systems*. Prentice-Hall, Inc. 30
- Ogata, K. (2003). *Ingeniería de control moderna*. Pearson Educación. 39

- Pearson, J. W. and Pestana, J. (2020). Preconditioners for Krylov subspace methods: An overview. *GAMM-Mitteilungen*, 43(4):e202000015. 36, 42
- Runyan, J. B. (2011). *A novel higher order finite difference time domain method based on the Castillo-Grone mimetic curl operator with applications concerning the time-dependent Maxwell equations*. PhD thesis, San Diego State University. ix, 16, 19, 22, 23
- Saad, Y. (2003). *Iterative methods for sparse linear systems*. SIAM. ix, 9, 28, 29, 30, 34, 35
- 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. 1, 32, 34, 45
- Sarkis, M., Schaerer, C. E., and Mathew, T. (2008). Block diagonal parareal preconditioner for parabolic optimal control problems. In *Domain decomposition methods in science and engineering XVII*, pages 409–416. Springer. 42
- Schaerer, C. E. and Kaszkurewicz, E. (2001). The shooting method for the solution of ordinary differential equations: A control-theoretical perspective. *International Journal of Systems Science*, 32(8):1047–1053. 28, 71
- Schaerer, C. E., Kaszkurewicz, E., and Mangiavacchi, N. (2004). A multilevel Schwarz shooting method for the solution of the Poisson equation in two dimensional incompressible flow simulations. *Applied Mathematics and Computation*, 153(3):803–831. 7
- SciPy Developers (2025). `scipy.sparse.linalg.lgmres` — SciPy v1.16.2 manual. 46
- Shashkov, M. (1995). *Conservative finite-difference methods on general grids*. CRC Press. 11
- Shashkov, M. and Steinberg, S. (1995). Support-operator finite-difference algorithms for general elliptic problems. *Journal of Computational Physics*, 118(1):131–151. 11
- Simoncini, V. and Szyld, D. B. (2007). Recent computational developments in Krylov subspace methods for linear systems. *Numerical Linear Algebra with Applications*, 14(1):1–59. 38
- Strang, G. (2006). *Linear algebra and its applications*. Thomson, Brooks/Cole. 9

- Thomas, L. (1949). Elliptic problems in linear differential equations over a network: Watson scientific computing laboratory. *Columbia Univ., NY*, 6. 9
- Trefethen, L. N. and Bau, D. (1997). *Numerical linear algebra*. SIAM. 71
- Van der Vorst, H. A. (1992). Bi-CGSTAB: A fast and smoothly converging variant of Bi-CG for the solution of nonsymmetric linear systems. *SIAM Journal on scientific and Statistical Computing*, 13(2):631–644. 34

# Appendix A

## Notes on Numerical Analysis

### A.1 Numerical Error Estimation

When solving real-world problems with finite-precision digital computers, numerical methods are sensitive to error, i.e., a difference between the true solution and that which is reached by the computer. The *total numerical error* is the summation of the truncation errors and roundoff errors. *Roundoff errors* arise because digital computers have magnitude and precision limits on the ability to represent numbers. They may propagate (increase) as the number of mathematical operations increase. In the other hand, *truncation errors* are those that result from using an approximation equation in place of an exact mathematical procedure, e.g., the truncation of terms in a Taylor series (Chapra, 2011).

For a discretization procedure, consider to take some initial step size  $h$  and decrease its magnitude. Initially, roundoff errors are expected to be negligible and truncation errors are expected to behave like some power of  $h$ , that is,  $O(h^k)$ , then the error  $E(h)$  behaves like

$$\begin{aligned} E(h) &= E_{\text{roundoff}} + E_{\text{trunc}}(h) \\ &\approx Ch^k + O(h^{k+1}), \end{aligned}$$

where  $C$  is a constant that depends on the problem and the discretization method used. The term  $O(h^{k+1})$  represents the higher-order error terms that one can neglect as  $h$  approaches zero.

Then,

$$\log E(h) \approx \log |C| + k \log h, \quad (\text{A.1})$$

so that, in a log-log plot, the truncation error behaves linearly with slope  $k$ , which

is the expected accuracy as usually defined in finite difference methods. Truncation errors are decreased as  $h$  gets smaller. A first step to test a code and ensure that it is working correctly is to try it on a problem for which the exact solution is known. Besides checking that the error is small in some grid, we can also refine the grid and check how the error is behaving as  $h$  gets smaller.

However, because the size of the problem usually increase as  $h$  decreases, so does the amount of computations, roundoff errors are increased by several cumulative roundoff procedures in those computations, as seen in A.1. Therefore, a trade-off between these two variables must be taken in consideration for the design of numerical algorithms.

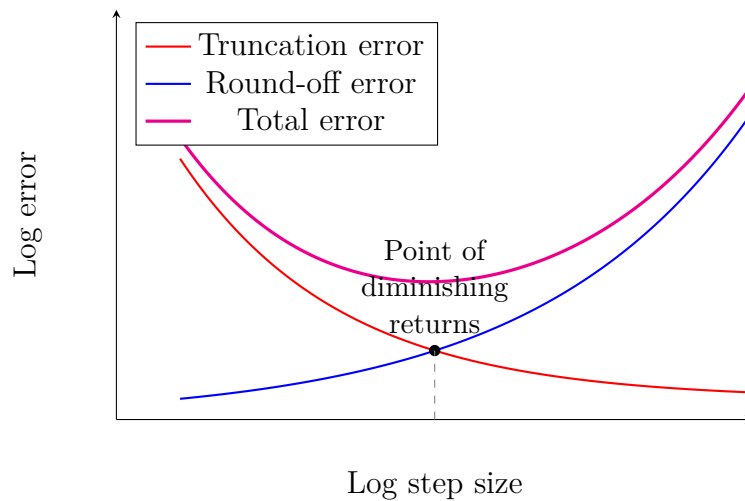


Figure A.1: A sketch of the trade-off between truncation and roundoff errors in the execution of a numerical method. As step-size gets smaller, the point of diminishing returns indicates that the roundoff errors begin to cancel the benefits of that step-size reduction. Adapted from Chapra (2011), page 79.

## A.2 Eigenvalues, Eigenvectors and Spectral Radius

A complex scalar  $\lambda$  is called an *eigenvalue* of the square matrix  $A$  if a nonzero vector  $\mathbf{x} \in \mathbb{C}^n$  exists such that  $A\mathbf{x} = \lambda\mathbf{x}$ . The vector  $\mathbf{x}$  is called an *eigenvector* of  $A$  associated with  $\lambda$ . The set of all eigenvalues of  $A$  is called the *spectrum* of  $A$  and it is denoted by  $\sigma(A)$ . The maximum modulus of the eigenvalues is called the *spectral radius* of  $A$  and it is denoted with  $\rho(A)$ :

$$\rho(A) = \max_{\lambda \in \sigma(A)} |\lambda|. \quad (\text{A.2})$$

## A.3 Norms of Vectors and Matrices

### A.3.1 Norm of a Vector

A significant portion of linear algebra deals with the necessity of generalizing the basic, geometric concepts of length or distance to nonvisual, high-dimensional vector spaces (Meyer, 2001). In order to measure the size of a vector or a matrix, we need to define a norm.

A norm for a real or complex vector space  $\mathcal{V}$  is a function  $\|\cdot\| : \mathcal{V} \rightarrow \mathbb{R}$  that satisfies the following properties:

1.  $\|\mathbf{x}\| \geq 0$  and  $\|\mathbf{x}\| = 0 \Leftrightarrow \mathbf{x} = \mathbf{0}$ .
2.  $\|\gamma\mathbf{x}\| = |\gamma|\|\mathbf{x}\|$  for all scalars  $\gamma$ .
3.  $\|\mathbf{x} + \mathbf{y}\| \leq \|\mathbf{x}\| + \|\mathbf{y}\|$  (triangle inequality).

Some important examples of norms are the *euclidean norm* (or  $L^2$  norm), defined as

$$\|\mathbf{x}\|_2 = \sqrt{\sum_{i=1}^n x_i^2}, \quad (\text{A.3})$$

where  $\mathbf{x} = (x_1, x_2, \dots, x_n) \in \mathbb{C}^n$ , and the *maximum norm* (or  $L^\infty$  norm), defined as

$$\|\mathbf{x}\|_\infty = \max_{1 \leq i \leq n} |x_i|. \quad (\text{A.4})$$

A general definition of a norm for a vector is the  $p$ -norm, defined as

$$\|\mathbf{x}\|_p = \left( \sum_{i=1}^n |x_i|^p \right)^{1/p}, \quad (\text{A.5})$$

where  $p \geq 1$ . The  $p$ -norm is a generalization of the euclidean norm and the maximum norm. For example, the  $L^2$  norm corresponds to  $p = 2$ , and the  $L^\infty$  norm corresponds to  $p = \infty$ .

### A.3.2 Norm of a Matrix

A *matrix norm* is a function  $\|\cdot\|$  from the set of all complex matrices (of all finite orders) into  $\mathbb{R}$  that satisfies the following properties (Meyer, 2001):

1.  $\|A\| \geq 0$  and  $\|A\| = 0 \Leftrightarrow A = \mathbf{0}$ .

2.  $\|\gamma A\| = |\gamma| \|A\|$  for all scalars  $\gamma$ ,
3.  $\|A + B\| \leq \|A\| + \|B\|$  for matrices of same size (triangle inequality),
4.  $\|AB\| \leq \|A\| \cdot \|B\|$  for all conformable matrices.

All properties are similar to those of the vector norm, except for the last one, which is an exclusive property of the matrix norm relative to the product of matrices.

Each vector  $p$ -norm induces a matrix norm, defined as:

$$\|A\|_p = \max_{\mathbf{x} \neq 0} \frac{\|A\mathbf{x}\|_p}{\|\mathbf{x}\|_p}. \quad (\text{A.6})$$

For different values of  $p$ , some matrix norms are:

$$\|A\|_1 = \max_{1 \leq j \leq n} \sum_{i=1}^n |a_{ij}| \quad (\text{A.7})$$

$$\|A\|_2 = (\rho(A^H A))^{1/2} \quad (\text{A.8})$$

$$\|A\|_\infty = \max_{1 \leq i \leq n} \sum_{j=1}^n |a_{ij}| \quad (\text{A.9})$$

where  $A^H$  is the conjugate transpose of  $A$  and  $\rho(M)$  is the spectral radius of a matrix  $M$ .

## A.4 Perturbation Analysis: Condition Number of a Matrix

When analyzing theoretical aspects of numerical solutions of PDEs, including their solvability by means of the singularity of the involved matrix of coefficients of the arising linear system of equations  $A\mathbf{x} = \mathbf{b}$ , an important concept is the *condition number* of a matrix. The condition number is a measure of how sensitive the solution of a linear system is to perturbations in the input data (the right-hand side vector  $\mathbf{b}$ ) or in the matrix  $A$ .

A problem is *ill-conditioned* if a small relative change in the data can cause a large relative error in its computed solution, regardless of the algorithm used to solve the problem (Meyer, 2001; Ford, 2014). One important difference to note is that between the words *stability* and *conditioning*. The property of stability refers to an algorithm (in the case of this book, the iterative, Krylov subspace-based method), whereas conditioning refers to a particular problem, regardless of the algorithm used.

In the case of linear systems of equations  $A\mathbf{x} = \mathbf{b}$ , where  $A$  is nonsingular (hence there exists a solution  $\mathbf{x}$ ), suppose that  $\delta A$ ,  $\delta \mathbf{x}$  and  $\delta \mathbf{b}$  are errors that give place to the perturbed linear system  $(A + \delta A)(\mathbf{x} + \delta \mathbf{x}) = (\mathbf{b} + \delta \mathbf{b})$ . The next Theorem specifies bounds for the errors involved.

**Theorem 2.** *If  $A$  is a nonsingular matrix,  $\mathbf{b} \neq 0$  is a vector and  $\mathbf{x}$  is solution of  $A\mathbf{x} = \mathbf{b}$ :*

1. *If  $(\mathbf{x} + \delta \mathbf{x})$  is the solution to the perturbed system  $(A + \delta A)(\mathbf{x} + \delta \mathbf{x}) = \mathbf{b}$ , then*

$$\frac{\|\delta \mathbf{x}\|}{\|\mathbf{x}\|} \leq \|A\| \|A^{-1}\| \cdot \frac{\|\delta \mathbf{b}\|}{\|\mathbf{b}\|}. \quad (\text{A.10})$$

2. *If  $(\mathbf{x} + \delta \mathbf{x})$  is the solution to the perturbed system  $A(\mathbf{x} + \delta \mathbf{x}) = \mathbf{b}$ , then*

$$\frac{\|\delta \mathbf{x}\|}{\|\mathbf{x} + \delta \mathbf{x}\|} \leq \|A\| \|A^{-1}\| \cdot \frac{\|\delta A\|}{\|A\|}. \quad (\text{A.11})$$

3. *If  $(\mathbf{x} + \delta \mathbf{x})$  is the solution to the perturbed system  $A(\mathbf{x} + \delta \mathbf{x}) = \mathbf{b} + \delta \mathbf{b}$ , then*

$$\frac{\|\delta \mathbf{x}\|}{\|\mathbf{x} + \delta \mathbf{x}\|} \leq \|A\| \|A^{-1}\| \cdot \left( \frac{\|\delta A\|}{\|A\|} + \frac{\|\delta \mathbf{b}\|}{\|\delta A\| \|\mathbf{x} + \delta \mathbf{x}\|} \right). \quad (\text{A.12})$$

*Proof.* The interested reader can go to [Ford \(2014\)](#), Theorem 10.4 for a detailed explanation.  $\square$

The previous result involves the factor  $\|A\| \|A^{-1}\|$ , which has a particular significance for the numerical definition of condition number.

For a given nonsingular matrix  $A \in \mathbb{R}^{n \times n}$ , the condition number is defined as:

$$\kappa(A) = \|A\| \cdot \|A^{-1}\|. \quad (\text{A.13})$$

We note that  $\kappa(A) \geq 1$ , and  $\kappa(\gamma A) = \kappa(A)$  for any  $\gamma \in \mathbb{R}$ .

The condition number of a nonsingular matrix  $A$  is directly related to the singular values of  $A$  ([Trefethen and Bau, 1997](#); [Schaerer and Kaszkurewicz, 2001](#)). In that sense, it can be proved that the condition number is equal to the ratio of the largest singular value  $\sigma_{\max}$  to the smallest singular value  $\sigma_{\min}$  of  $A$ :

$$\kappa(A) = \frac{\sigma_{\max}}{\sigma_{\min}}. \quad (\text{A.14})$$

and, if the matrix  $A$  is symmetric positive definite, the condition number can be expressed in terms of the eigenvalues  $\lambda_{\max}$  and  $\lambda_{\min}$  of  $A$  as:

$$\kappa(A) = \frac{\lambda_{\max}}{\lambda_{\min}}. \quad (\text{A.15})$$

The larger  $\kappa(A)$  is, the matrix is said to be more *ill-conditioned*, meaning that small changes in the input data can lead to large changes in the solution. Conversely, a small  $\kappa(A)$  indicates that the matrix is *well-conditioned*, meaning that the solution is relatively stable with respect to perturbations in the input data.

# Appendix B

## Presentation in National and International Conferences

### B.1 Presentations in International Conferences

In this appendix, the poster presentations shown at the CNMAC (Congresso de Matemática Aplicada e Computacional) event, organized by the Brazilian Society of Applied and Computational Mathematics (Sociedade Brasileira de Matemática Aplicada e Computacional, SBMAC), in the years 2023 and 2024 during the postgraduate studies. The titles of the studies are: "Mimetic Opeator Discretization 1D: Exploration of classical iterative methods and preconditioners", and "A numerical investigation on iterative methods for the two-dimensional Poisson equation discretized with high-order mimetic operators".

The CNMAC brings together researchers and practitioners in applied and computational mathematics from Brazil and Latin Amnerica. The conference features a variety of presentations, including invited talks, contributed talks, and poster sessions, covering a wide range of topics in applied mathematics, numerical analysis, and computational methods.

The CNMAC 2023 was held in the city of Bonito, MS, from September 18 to 22, 2023. The CNMAC 2024 was held in the city of Porto de Galinhas, PE, Brazil, from September 16 to 20, 2024.

## Mimetic Operator Discretization 1D:

### Exploration of classical iterative methods and preconditioners

GUSTAVO E. ESPÍNOLA<sup>1</sup>, JUAN C. CABRAL<sup>2</sup>, CHRISTIAN E. SCHAEERER<sup>3</sup>,

National University of Asuncion, Polytechnic School

Grupo de Investigación en Computación Científica y Matemática Aplicada - CCyMA

gustavo.espinola@fpuna.una.py<sup>1</sup>, jccabral@pol.una.py<sup>2</sup>, cschaerer@pol.una.py<sup>3</sup>.

#### Abstract

This work performs experiments looking for preconditioned iterative methods for sparse linear systems that overcome stagnation for the numerical resolution of the one-dimensional Poisson equation with Robin boundary conditions, discretized with a second-order mimetic method, based on the 1D Castillo-Grone mimetic operator. By refining the grid, only an appropriate preconditioner will allow the associate linear system iterative method to achieve convergence.

**Keywords:** adaptive iterative methods, Castillo-Grone mimetic operator, Krylov Sub-space methods, preconditioners, sparse linear systems.

#### Introduction

The Castillo-Grone method [1] allows for the construction of discrete differential operators, e.g. divergence, gradient, which are  $O(h^k)$  accurate (for any step size  $h$  and every even  $k$ ) at the interior points and the domain boundary. The aim of these mimetic operators is to satisfy in the discrete sense the vectorial, global conservation laws that the continuum model do in order to make the problem more accurate to physical constraints.

#### Goals

- To build non-singular matrices from second-order mimetic operators for one-dimensional boundary conditions.
- To compare the results, efficiency, and order of convergence between different implemented iterative methods for linear systems and preconditioners.

#### Methodology

Consider the one-dimensional Poisson equation on a uniform grid [2],

$$\nabla^2 u(x) = f(x) \quad \text{on } [0, 1] \quad (1)$$

subject to Robin boundary conditions

$$\alpha f(0) - \beta f'(0) = -1; \quad \alpha f(1) + \beta f'(1) = 0,$$

where

$$f(x) = \frac{\lambda^2 e^{\lambda x}}{e^{\lambda x} - 1}, \quad \alpha = -e^\lambda, \quad \beta = \frac{e^\lambda - 1}{\lambda}, \quad \lambda = -1.$$

The analytical solution to the problem is

$$u^*(x) = \frac{e^{\lambda x} - 1}{e^\lambda - 1}.$$

To solve the linear system  $Au = f$  associated with this problem, we used the restarted GMRES with  $m = 10$ , and the BiCGStab as iterative methods. Also, the following preconditioners: Jacobi and successive over-relaxation (SOR) with  $\omega = 1$ , are used to try to improve the convergence (relative residual norm  $\|b - Au_j\|/\|b\|$  less than  $10^{-6}$ ).

#### Numerical results

| h    | Jacobi  |                 |          | SOR ( $\omega = 1$ ) |                 |          |
|------|---------|-----------------|----------|----------------------|-----------------|----------|
|      | Condest | GMRES(10)       | BiCGStab | Condest              | GMRES(10)       | BiCGStab |
| 0.20 | 311.9   | 0.002184        | 0.002184 | 190.6                | 0.002184        | 0.002184 |
| 0.10 | 1073.0  | 0.000537        | 0.000545 | 662.5                | 0.000543        | 0.000541 |
| 0.05 | 3948.4  | 0.000126        | 0.00136  | 2421.5               | <b>1.008095</b> | 0.00132  |
| 0.01 | 91903.2 | <b>1.165131</b> | 0.000004 | 55928.2              | <b>1.142317</b> | 0.000010 |

Table 1: Relative error  $\|u - u^*\|/\|u^*\|$  for different iterative solvers and preconditioners.

Numbers in red, bold face indicate that the combination does not reach convergence.

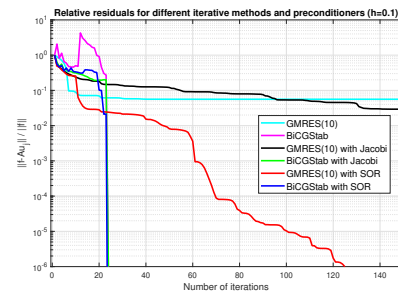


Figure 1: Relative residuals for GMRES(10) and BiCGStab using step size  $h = 0.1$ .

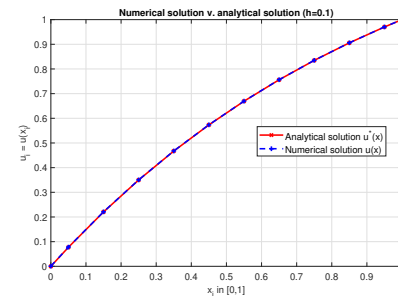


Figure 2: Numerical v. analytical solution of the 1D Poisson equation using step size  $h = 0.1$ .

#### Conclusion

For smaller step size  $h$ , only an appropriate preconditioner will allow convergence regardless of the chosen iterative method. Ongoing research work is analyzing the matrix structure for higher orders and dimensions (2D, 3D). Adaptive iterative methods and modern, block preconditioning methods may be useful to solve the stagnation problem.

#### References

- [1] Castillo, J. E., Grone, R. D. (2003). A matrix analysis approach to higher-order approximations for divergence and gradients satisfying a global conservation law. *SIAM Journal on Matrix Analysis and Applications*, 25(1), 128-142.
- [2] Hernández, F. F., Castillo, J. E., Larrazábal, G. A. (2007). Large sparse linear systems arising from mimetic discretization. *Computers Mathematics with Applications*, 53(1), 1-11.

#### Acknowledgements

This work has the financial support for Research Projects, given by the Polytechnic School - FPUNA. JCC and CES acknowledges PRONII-CONACYT-Paraguay.

Figure B.1: Poster presentation shown at the CNMAC 2023.

## A numerical investigation on iterative methods for the two-dimensional Poisson equation discretized with high-order mimetic operators

Gustavo Espínola<sup>1</sup>, Juan Cabral<sup>2</sup>, Jhabriel Varela<sup>3</sup>, and Christian Schaefer<sup>4</sup>

<sup>1,2,3,4</sup> Polytechnic School, National University of Asunción, Paraguay  
Grupo de Investigación en Computación Científica y Matemática Aplicada - CCyMA  
gustavo.espinola@fpma.edu.py<sup>1</sup>, jcabral@pol.una.py<sup>2</sup>, jhabriel@pol.una.py<sup>3</sup>, cschaefer@pol.una.py<sup>4</sup>

### Abstract

Mimetic operators are of increasing interest to the scientific computing community due to their ability to preserve many important properties of the continuous problem (e.g. conservation laws) while maintaining the same order of accuracy on the boundary as in the interior points. However, this results in locally dense and potentially ill-conditioned linear systems that are challenging to solve. This issue can partially be addressed using adequate iterative solvers, which is the focus of this work. Using the two-dimensional Poisson equation and mimetic operators of order  $k = 2$  and  $k = 4$ , we compare the computational times obtained with different Krylov-subspace-based iterative methods used for the resolution of the linear systems.

**Keywords:** mimetic operators, Krylov subspace methods, sparse linear systems.

### Introduction

Mimetic operators for the gradient ( $G = \nabla$ ), divergence ( $D = \nabla \bullet$ ) and Laplacian ( $L = \nabla^2 = \Delta$ ) are discrete analogs of the continuum counterparts that still fulfill the classical identities from vector calculus. For example, while providing uniform order of accuracy, the operators satisfy the discrete version of the Gauss extended divergence theorem:

$$h(Dv, f)_Q + h(v, Gf)_P = (Bv, f) \quad (1)$$

The discretization scheme is the staggered grid: scalar variables are stored at the centers of the cells, while vector components are placed at the edges.

When using mimetic operators, we are not discretizing the equations as it is done with standard finite-difference methods, but instead we construct matricial analogs to the differential operator:

$$\Delta \phi = f \text{ or } \nabla \bullet \nabla \phi = f \Rightarrow L \phi = f \text{ or } DG \phi = f \quad (2)$$

In this work, the studied operators are those developed by Corbino and Castillo [1].

### Methodology

The numerical experiments, a follow-up from the previous work at [3], were run on a computer with Intel Core i5-12450 with 16GB RAM, using the software MATLAB R2022a for Windows 11. The chosen benchmark problem is the 2D minimal Poisson equation, subject to Robin boundary conditions:

$$\begin{aligned} -\Delta u &= f \text{ in } \Omega \\ \alpha u + \beta \nabla u \bullet n &= \gamma \text{ on } \Gamma. \end{aligned} \quad (3)$$

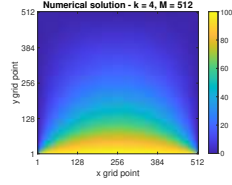


Figure 1: Numerical solution of the 2D Poisson equation.

The Mimetic Operators Library Enhanced (MOLE), available at <https://github.com/csrc-sdsu/mole>, was employed for the discretization process, which results in large sparse linear systems. To solve them, the following iterative methods were applied: GMRES( $m$ ), in which case the built-in MATLAB implementation is executed, LGMRES( $m, l$ ), GMRES-E( $m, d$ ), and PD-GMRES( $m, j$ ) [2], which is equipped with a discrete Proportional-Derivative controller for the restart parameter  $m$ :

$$m_j = m_{j-1} + \left[ \alpha_P \frac{\|r_s^{(j)}\|}{\|r_s^{(j-1)}\|} + \alpha_D \frac{\|r_s^{(j)}\| - \|r_s^{(j-2)}\|}{2\|r_s^{(j-1)}\|} \right], \quad (5)$$

where  $\alpha_P, \alpha_D \in \mathbb{R}$  and  $\lfloor \cdot \rfloor$  denotes the integer part.

All these solvers are available in the Krylov Subspace-Based Adaptive Solvers (KrySBAS) library (<https://github.com/nidtec-una/krybas-dev>).

### Numerical results

Table 1: List of test problems in the Poisson problem with different precision orders ( $k$ ) and discretization sizes ( $M$ ).

| $k \setminus M$ | size(A) | nnz(A)    | condPreJacobi(A) | condPostJacobi(A) |
|-----------------|---------|-----------|------------------|-------------------|
| 2 \ 32          | 1 156   | 5 252     | 753.8903         | 693.0954          |
| 2 \ 64          | 4 356   | 20 740    | 3.0170E+03       | 2.413E+03         |
| 2 \ 128         | 16 900  | 82 436    | 1.2008E+04       | 9.655E+03         |
| 2 \ 256         | 66 564  | 328 708   | 4.8281E+04       | 3.863E+04         |
| 2 \ 512         | 264 196 | 1 312 772 | 1.9312E+05       | 1.543E+05         |
| 4 \ 32          | 1 156   | 13 316    | 1.0895E+03       | 843.0009          |
| 4 \ 64          | 4 536   | 53 252    | 4.0391E+03       | 3.3748E+03        |
| 4 \ 128         | 16 900  | 212 996   | 1.6160E+04       | 1.3502E+04        |
| 4 \ 256         | 66 564  | 851 972   | 6.4642E+04       | 5.4011E+04        |
| 4 \ 512         | 264 196 | 3 407 876 | 2.5857E+05       | 2.1603E+05        |

Table 2: Metrics for different problems and iterative methods: average computational time in seconds and number of inner iterations required for  $\|r_j\|/\|r_0\| < 10^{-9}$ . For the cases where it does not reach convergence before 100 restarts (outer iterations) the method is stopped and the time is denoted as NC. Bold face indicate smaller computational time.

| $k, M$ | GMRES( $m$ )<br>Time<br>(Iterations) | PD-GMRES( $m, j$ )<br>Time<br>(Iterations) | LGMRES<br>Time<br>(Iterations) | GMRES-E<br>Time<br>(Iterations) |
|--------|--------------------------------------|--------------------------------------------|--------------------------------|---------------------------------|
| 2, 32  | 0.0202<br>(170)                      | 0.053<br>(183)                             | 0.0618<br>(150)                | 0.1434<br>(120)                 |
| 2, 64  | 0.1491<br>(561)                      | 0.1369<br>(391)                            | 0.1370<br>(270)                | 0.1498<br>(240)                 |
| 2, 128 | 1.9149<br>(1665)                     | 0.5185<br>(710)                            | 0.4962<br>(510)                | 0.5619<br>(540)                 |
| 2, 256 | NC                                   | 3.0038<br>(1536)                           | 2.4703<br>(969)                | 4.2089<br>(1440)                |
| 2, 512 | NC                                   | NC                                         | 46.388<br>(1740)               | NC                              |
| 4, 32  | 0.0078<br>(253)                      | 0.013<br>(250)                             | 0.011<br>(210)                 | 0.016<br>(150)                  |
| 4, 64  | 0.233<br>(700)                       | 0.138<br>(471)                             | 0.126<br>(330)                 | 0.125<br>(300)                  |
| 4, 128 | 2.698<br>(2154)                      | 0.673<br>(769)                             | 0.639<br>(570)                 | 0.806<br>(660)                  |
| 4, 256 | NC                                   | 4.503<br>(1787)                            | 3.643<br>(1140)                | 6.638<br>(1770)                 |
| 4, 512 | NC                                   | NC                                         | 61.96<br>(2010)                | NC                              |

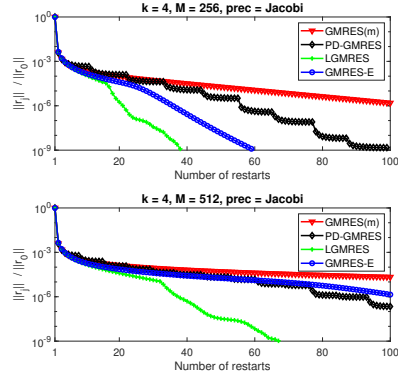


Figure 2: Relative residual norms v. number of GMRES restarts, for two examples.

### Conclusion

Numerical experiments for different discretization sizes  $M$  (of an  $(M+2) \times (M+2)$  grid), precision orders of the mimetic operators ( $k$ ) and iterative methods for the resulting linear systems are compared. According to the preliminary results, shown in Table 2, a strategy for modifying the size of the search subspace adaptively is good enough to accelerate convergence up to a certain number of discretization size: in the cases when the 1D step size is smaller, even the adjustment of the restart parameter  $m$  from GMRES is not sufficient to reach convergence under the iteration constraints. Hence, an enrichment of the search subspace is of substantial help. For the selected problems, the LGMRES( $m, k$ ) has performed better in the majority of the cases.

### Acknowledgements

The authors are funded by the Polytechnic Faculty (FPUNA) through its Research Department. GE is funded by the Paraguayan National Council of Science and Technology (CONACYT) through the Incentive for the Training of Researchers in National Postgraduate Programs. JV is funded by CONACYT through the Advanced Human Capital Insertion Program in Academia PRIA01-S.

### References

- [1] J. Corbino and J. Castillo. High-order mimetic finite-difference operators satisfying the extended Gauss divergence theorem. *Journal of Computational and Applied Mathematics*, 364:112326, 2020.
- [2] R. Cuevas, C. Schaefer, and A. Bhatta. A proportional-derivative control strategy for restarting the GMRES ( $m$ ) algorithm. *Journal of Computational and Applied Mathematics*, 337:209–224, 2018.
- [3] G. Espínola, J. Cabral, and C. Schaefer. Mimetic operator discretization 1D: Exploration of classical iterative methods and preconditioners. *Proceeding Series of the Brazilian Society of Computational and Applied Mathematics*, 10(1), 2023.

## **B.2 Presentations in National Confereneces**

Two poster presentations, "Mimetic Operators and Iterative Methods for the Resolution of Flow Equations", and "KrySBAS: Krylov Subspace-Based Adaptive Solvers", were introduced in the 4th International Conference on Specialization Courses of the Network on Sanitation and Water Resources through Innovative and Sustainable Technologies (RED AMARU) of the Ibero-American Program on Science and Technology for Development (CYTED), given place in the Polytechnic School at Universidad Nacional de Asunción, in August 20-24 2024.

## Mimetic operators and iterative methods for the resolution of flow equations

Gustavo Espínola<sup>♣</sup> Juan Cabral<sup>♣</sup> Jhabriel Varela<sup>♣</sup> Christian Schaefer<sup>♣</sup>

Scientific Computing and Applied Mathematics Group, Polytechnic School, National University of Asuncion, Paraguay<sup>♣</sup>

<sup>♣</sup>[gustavoespinola@fpuna.edu.py](mailto:gustavoespinola@fpuna.edu.py)

### Introduction

Mimetic operators [2] are discretization schemes based on matricial discrete analogs for the classical operators of vector calculus such as gradient ( $G$  or  $\nabla$ ), divergence ( $D$  or  $\nabla \bullet$ ), Laplacian ( $L$  or  $\nabla^2$  or  $\Delta$ ), etc., which satisfy, in the discrete sense, the properties of the continuous model of the problem to be solved, for example, the groundwater flow equation.

### Building mimetic operators

Mimetic operators are finite difference methods built for staggered grids. In such grids, scalar variables are stored at the centers of the cells, while vector components are located at the edges (or faces, in 3D).

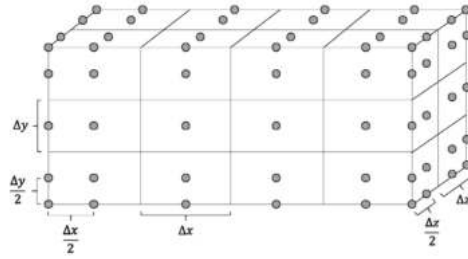


Figure 1: Three-dimensional, uniform staggered grid. Source: [2]

This design permits to satisfy local and global conservation laws from physics and provide a uniform order of accuracy over the domain up to and including the boundaries. They can be applied to solve problems in non-trivial geometries.

The first step in designing the model is the construction of one-dimensional, second-order gradient and divergence operators. They can be extended to higher-order, higher dimensional operators by using matricial operations such as the Kronecker productmatrix products such as the Kronecker product. Refer to [2] for the whole deduction process.

### Experimental setup

For testing purposes, consider the one-dimensional Poisson equation:

$$\nabla^2 u(x) = f(x) \quad \text{on } [0, 1]$$

subject to Robin boundary conditions

$$\alpha f(0) - \beta f'(0) = -1; \quad \alpha f(1) + \beta f'(1) = 0,$$

where

$$f(x) = \frac{\lambda^2 e^{\lambda x}}{e^{\lambda x} - 1}, \quad \alpha = -e^\lambda, \quad \beta = \frac{e^\lambda - 1}{\lambda}, \quad \lambda = -1.$$

The analytical solution to the problem is

$$u^*(x) = \frac{e^{\lambda x} - 1}{e^\lambda - 1}.$$

The construction of the mimetic operators can be done through the MOLE library [1]. The formulation of such a discrete model results in an ill-conditioned sparse linear system  $Au = f$ . The solution can be implemented with iterative preconditioned Krylov subspace methods. Iterative methods derived from the generalized least residue algorithm with restarts or GMRES( $m$ ), developed in the KrySBAS library [3], can be chosen for such goal.

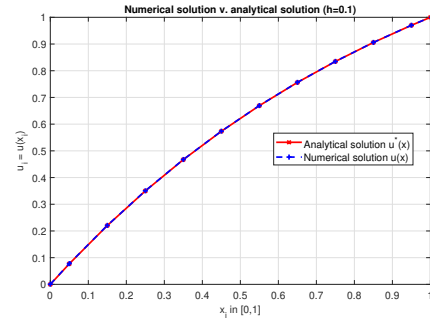


Figure 2: Numerical v. analytical solution of the 1D Poisson equation using mimetic discretization scheme of size  $h = 0.1$ .

The mimetic scheme is capable of following the analytical solution correctly. The source code for this experiment is available in the website <https://github.com/guespinola/mimetic-1dpoisson>.

### References

- [1] J. Corbino. MOLE: Mimetic Operators Library Enhanced. Online. Retrieved June 21, 2024 from <https://github.com/jcorbino/mole>.
- [2] J. Corbino and J. Castillo. High-order mimetic finite-difference operators satisfying the extended gauss divergence theorem. *Journal of Computational and Applied Mathematics*, 364:112326, 2020.
- [3] J. Varela, G. Espinola, and J. Cabral. KRYSBAS: Basic Adaptive Krylov solvers. Online. Retrieved June 21, 2024 from <https://github.com/nidtec-una/krysbas-dev>.

### Acknowledgements

GE is funded by the National Council of Science and Technology of Paraguay (CONACYT) through the Incentive for the Training of Researchers in National Postgraduate Programs - PROCIENCIA II. JV is funded by CONACYT through the Program for the Insertion of Advanced Human Capital in Academia PIA01-8.

Figure B.3: Poster presentation shown at the RED-AMARU event: Mimetic Operators and Iterative Methods for the Resolution of Flow Equations.

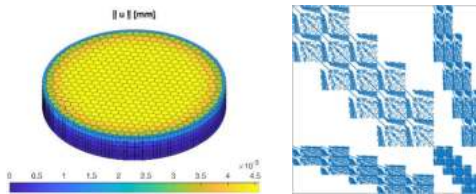


## KrySBAS: Krylov Subspace-Based Adaptive Solvers

Jhabriel Varela\* Juan C. Cabral\* Gustavo Espínola\* Christian E. Schaerer\*  
Scientific Computing and Applied Mathematics Group, Polytechnic School, National University of Asunción\*  
jhabriel@pol.una.py

### Introduction

In many applications arising in scientific computing, we must solve a linear system of the type  $Ax = b$ , where  $A$  is a sparse, often non-symmetric and potentially ill-conditioned matrix,  $x$  is the unknown solution and  $b$  is the vector of known data. See Figure 1 for a typical system.



**Figure 1:** Left: Magnitude of the displacement field after solving the equations of unsaturated flow in deformable porous media in the case of evaporation [1]. Right: Sparsity of the linearized system of equations resulting from a multipoint flux / multipoint stress finite volume discretization [1].

Solving large systems with direct solvers is not feasible due to computational limitations. One therefore employs indirect solvers such as GMRES or its restarted version GMRES( $m$ ) [1]. However, when the linear systems are ill-conditioned, GMRES( $m$ ) can exhibit low convergence at best or stagnation at worst. One potential remedy to this problem is to employ adaptive solvers.

### The need for KrySBAS

Several adaptive solvers has been proposed in the last three decades. However, many of such solvers are currently not available to the scientific community. We were therefore motivated to develop KrySBAS, an open-source toolbox written for MATLAB/OCTAVE containing a collection of adaptive solvers based on Krylov subspaces. KrySBAS is developed under GPL v.3.0 license and is available at:

[www.github.com/nidtec-una/krysbas-dev](https://www.github.com/nidtec-una/krysbas-dev)

### Installation

To install KrySBAS, simply clone the GitHub repository, and add the folder to your MATLAB or OCTAVE path.

### Available solvers

KrySBAS currently provides three adaptive solvers:

- PD-GMRES [2]
- LGMRES [3]
- GMRES-E [4]

New solvers will be included shortly.

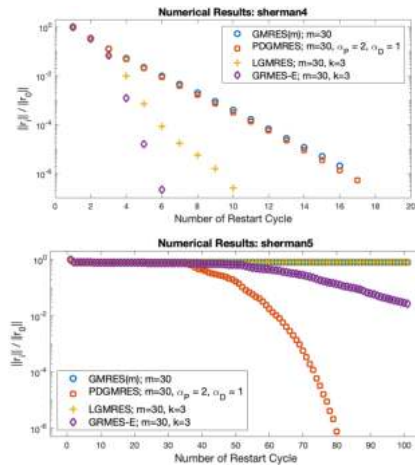
### Testing, coverage and documentation



**Figure 2:** KrySBAS is thoroughly tested with automated workflows and coverage reports. In addition, all the functionality is carefully documented and available through read-the-docs.

### Numerical examples

Convergence of available solvers for two benchmark matrices arising in fluid dynamics problems: sherman4 and sherman5.



**Figure 3:** Relative residual norms as a function of the outer iterations for GMRES( $m$ ), PDGMRES, LGMRES, and GMRES-E. Top: Analysis for the matrix sherman4. Bottom: Analysis for sherman5.

### References

- [1] Varela, J., Gasda, S. E., Keilegavlen, E., & Nordbotten, J. M. (2021). A finite-volume-based module for unsaturated poroelasticity. *Advanced Modeling with the MATLAB Reservoir Simulation Toolbox*.
- [2] Saad, Y., & 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.
- [3] Núñez, R. C., Schaerer, C. E., & Bhaya, A. (2018). A proportional-derivative control strategy for restarting the GMRES( $m$ ) algorithm. *Journal of Computational and Applied Mathematics*, 337, 209-224.
- [4] Baker, A. H., Jessup, E. R., & Manteuffel, T. (2005). A technique for accelerating the convergence of restarted GMRES. *SIAM Journal on Matrix Analysis and Applications*, 26(4), 962-984.
- [5] Morgan, R. B. (1995). A restarted GMRES method augmented with eigenvectors. *SIAM Journal on Matrix Analysis and Applications*, 16(4), 1154-1171.

### Acknowledgements

JV is funded by CONACYT through the project PRIA01-8 and the FPUNA. GE is funded by CONACYT through the Incentive for the Training of Researchers in National Postgraduate Programs.

Figure B.4: Poster presentation shown at the RED-AMARU event: Krylov Subspace-Based Adaptive Solvers.