Trends in Computational and Applied Mathematics

Trends in Computational and Applied Mathematics, **26** (2025), e01704
Sociedade Brasileira de Matemática Aplicada e Computacional
Online version ISSN 2676-0029 www.scielo.br/tcam
ORIGINAL ARTICLE
doi: 10.5540/tcam.2025.026.e01704

# A DIMENSION-INDEPENDENT FINITE DIFFERENCE METHOD FOR THE POISSON EQUATION USING AN INDEX MAPPING FUNCTION

A. V. N. Vieira[1*], A. W. Vieira[2] and N. da H. Lisboa[3]

Received on April 12, 2022 / Accepted on June 5, 2025

**ABSTRACT.** The numerical solution to the Poisson Problem is widely known and studied in various fields of science for its vast applications. However, most applications consider the case in two dimensions, with fewer studies addressing the problem in dimension three or higher. Most literature texts present a two-dimensional case implementation using an approach that makes it difficult to extend to higher dimensions. Our work aims to propose a generalization of the numerical solution to the Poisson problem that can be implemented for any dimension. The strategy used considers an index function that enumerates the mesh points so that, using this index, the implementation is easily extended to any dimension. In addition to the numerical solution extension, we developed the mathematical foundation for the consistency and stability of the solution in arbitrary dimension. The preliminary results consider the implementation in Python and experiments that demonstrate the feasibility of the proposed methodology.

**Keywords:** Poisson's equation, Dirichlet boundary condition, numerical solution.

## 1 INTRODUCTION

It is a known fact that many models exist in the literature for the Poisson Problem in one and two dimensions, both demonstrated with the necessary mathematical rigor and implemented in some programming language. However, there is a smaller amount of work on mathematical modeling for the three-dimensional case of the Poisson Problem. There is also a difficulty when trying to expand the modeling to a new dimension, requiring a new implementation, with non-trivial changes, for each desired dimension. In the present work, we note that the creation of an index function to enumerate the mesh points corresponding to unknown values, not only facilitates the implementation for the two-dimensional and three-dimensional case, but also creates a generalization that facilitates the implementation in any dimension. This approach, using the index

*Corresponding author: Antonio Wilson Vieira – E-mail: antonio.vieira@unimontes.br
[1]Instituto Federal do Norte de Minas Gerais, MG, Brazil – E-mail: anderson.vieira@ifnmg.edu.br https://orcid.org/0000-0003-3670-9735
[2]Universidade Estadual de Montes Claros, MG, Brazil – E-mail: antonio.vieira@unimontes.br https://orcid.org/0000-0001-8079-1671
[3]Universidade Estadual de Montes Claros, MG, Brazil – E-mail: narciso.lisboa@unimontes.br https://orcid.org/0009-0007-1159-8916

function, was not found in any other work during our literature review. This function plays a very important role in the implementation because, by enumerating sampled elements of the domain, it simplifies the construction of the coefficient matrix of the sparse linear system that arises in the modeling of the solution. The index function allows this to be done systematically and efficiently and, moreover, it can be rewritten very easily so that the path to access the elements is not unique and depends on the enumeration one chooses, without this choice interfering with the solution.

In this work, therefore, we generalize the numerical solution to the Poisson problem using the finite difference method in the $n$ dimension, with $n \geq 1$. Consider the second-order elliptical problem, in the dimension $n$, given by

$$-\Delta u(x) = f(x),\ x \in \Omega, \tag{1.1}$$

with the Dirichlet boundary condition

$$u(x) = g(x),\ x \in \partial\Omega,$$

where $\Omega = (r_1, s_1) \times \ldots \times (r_n, s_n)$, $r_j < s_j$, for $j = 1, \ldots, n$, $\partial\Omega$ is the boundary of $\Omega$ and $\Delta$ denotes the Laplacian operator, given by

$$\Delta u(x) = \sum_{j=1}^{n} \frac{\partial^2 u}{\partial x_j^2}(x).$$

There are different approaches to solving these problems, depending on the geometry of the domain. Some recent studies, including the one, two and three dimensional Poisson equation, present numerical solutions using increasingly accurate and efficient methods [19], [18]. There is a large amount of literature about the numerical solution of a Poisson equation, and a good review of results on this important subject can be found in [4], [12], [8], [7], [17], [21], [22], [27].

In this paper, motivated by the two-dimensional case [6], [15] and [24], the finite difference method is preferred due to the ease of implementation and computational efficiency for linear problems in rectangular regions of $\mathbb{R}^2$. In practice, problems may be nonlinear, have non-rectangular domains, and be defined in $\mathbb{R}^n$ with $n > 2$. Therefore, extending the method to handle these cases is of great importance for solving real-world problems.

In recent decades, numerous applications and extensions of the finite difference method for non-linear Poisson-type differential equations have emerged. Owing to the nonlinear behavior of the partial differentiable equation under consideration, the theoretical analysis has been proved to be considerably difficult, especially for problems with irregular geometries and non-uniform boundary conditions. To study the nonlinearity in complex solution domain, it is a long history in resorting to numerical solutions. So far, different numerical techniques, including finite difference method [1], [5], [16], exponential finite difference method [20], quasilinear boundary element method [14], hybrid fundamental solution-based finite element method [26], the method of fundamental solution [2], [3], among others, have been developed to solve nonlinear Poisson-type problems. More discussions on this topic, using other numerical techniques, can be found in the literature [28], [9].

There are several studies in the literature in which the finite difference method is applied to solve elliptic equations in the irregular domain [11], [13]. In principle, its application is restricted to a domain with a boundary that has a relatively simple geometry [10], but for a domain with a more general boundary, certain special steps must be taken, with an exploratory approach to overcome this fact, as reported in the literature [23].

Although the finite difference formulation in $\mathbb{R}^n$ follows classical generalizations, our focus is on an efficient computational abstraction via an index function that is rarely addressed explicitly in the literature. In most cases, the solution goes through the construction of a system of linear equations $AU = B$, where $A$ is a sparse matrix in which the location of non-zero elements is not trivial, especially as the dimension $n$ of the domain increases. Once the problem is formulated as a linear system, a wide range of existing numerical solvers and optimization techniques can be employed to improve CPU efficiency, particularly given that the resulting system matrix is highly sparse. This opens up opportunities for further performance improvements through the use of specialized sparse matrix libraries and parallel computing techniques.

The main contributions of this paper are the generalization of the numerical solution of the Poisson equation, with Dirichlet boundary conditions, in the dimension $n$, for all $n \geq 1$, and the introduction of an index function $I$ that facilitates the numerical modeling of the solution, and its computational implementation, using the sparse system of linear equations $AU = B$ that results from the numerical problem studied, as seen in Section 2. The index function $I$, while enumerating the unknown elements of the discrete sample of the domain, facilitates the determination of the numerical solution of the Poisson Problem in any dimension of the Euclidean space and, using it, it is possible to know, for example, in which line and in which column each non-null element of the matrix $A$ of the above system is located.

The rest of the article is organized as follows: In Section 2, we formulate the finite difference method of $2n+1$ points, with $n \geq 1$, to obtain a numerical solution of the Poisson equation, with Dirichlet boundary conditions. In Section 3, we demonstrate the convergence of the method. For this, its consistency and stability is ensured. The computational implementation of the method is discussed in Section 4. In Section 5, we present experimental results to validate the proposed method in the solution of some elliptic problems and, finally, in Section 6, we present the conclusions and future directions of the study.

## 2 THE FINITE DIFFERENCE METHOD IN DIMENSION $N$

First, consider $u$ a real function of $n$ real variables sufficiently differentiable, $x = (x_1, \ldots, x_n)$ in $\mathbb{R}^n$ and $h_j > 0$, with $j = 1, \ldots, n$. Using Taylor's formula, we get the central difference formula

$$\begin{aligned}\frac{\partial^2 u}{\partial x_j^2}(x) = {} & \frac{u(x+h_je_j) - 2u(x) + u(x-h_je_j)}{(h_j)^2} \\ & - \frac{(h_j)^2}{12}\frac{\partial^4 u}{\partial x_j^4}(x + (\rho_j - x_j)e_j),\end{aligned} \tag{2.1}$$

where $\rho_j \in (x_j - h_j, x_j + h_j)$ and $e_j = (0,...,0,1,0,...,0)$ is a vector of the canonical basis of $\mathbb{R}^n$, with the jth coordinate equal to one and the others null, for $j = 1,\dots,n$.

To approximate second-order partial derivatives $\frac{\partial^2 u}{\partial x_j^2}$, $j = 1,\dots,n$ , by finite differences, we cover the region $\Omega \cup \partial\Omega$ with a mesh. The points of this mesh are denoted by $x^{i_1,\dots,i_n} = (x_1^{i_1},\dots,x_n^{i_n})$, where $x_j^{i_j} = r_j + i_j.h_j$, $h_j = \frac{s_j - r_j}{M_j}$ with $i_j = 0,1,\dots,M_j$ and $j = 1,\dots,n$.

We denote by $\Omega_\delta$ the set of mesh points that are interior to $\Omega$ and by $\partial\Omega_\delta$ the set of mesh points that are on the border of $\Omega$. Then, using the equations $(1.1)$ and $(2.1)$, we get the finite difference method of $2n+1$ points,

$$\Delta_\delta U_{i_1,\dots,i_n} = -f(x^{i_1,\dots,i_n}) \text{ in } \Omega_\delta \tag{2.2}$$

and

$$U_{i_1,\dots,i_n} = g(x^{i_1,\dots,i_n}) \text{ on } \partial\Omega_\delta, \tag{2.3}$$

where $U_{i_1,\dots,i_n}$ is the numerical solution and $\Delta_\delta$ is the discrete Laplacian operator given by

$$\Delta_\delta U_{i_1,\dots,i_n} = \sum_{j=1}^{n} \frac{U_{i_1,\dots,i_j+1,\dots,i_n} - 2U_{i_1,\dots,i_n} + U_{i_1,\dots,i_j-1,\dots,i_n}}{(h_j)^2}.$$

By substituting into the equation $(2.2)$ each of the $(M_1-1)\cdot(M_2-1)\cdot\ldots\cdot(M_n-1)$ points from $\Omega_\delta$ and using the equation $(2.3)$, we get the linear system $AU = B$, with $(M_1-1)\cdot(M_2-1)\cdot\ldots\cdot(M_n-1)$ equations and the same number of unknowns, where $U$ is the column vector given by

$$\begin{aligned} U =& (U_{1,1,\dots,1},\dots,U_{M_1-1,1,\dots,1},\dots,U_{1,M_2-1,\dots,M_n-1},\dots,\\ & U_{M_1-1,M_2-1,\dots,M_n-1})^T, \end{aligned}$$

$$A = \begin{bmatrix} a & b_1 & & b_n & & & \\ b_1 & a & b_1 & & \ddots & & 0 \\ & \ddots & \ddots & \ddots & 0 & \ddots & \\ b_n & & \ddots & \ddots & \ddots & & b_n \\ & \ddots & 0 & \ddots & \ddots & \ddots & \\ 0 & & \ddots & & b_1 & a & b_1 \\ & & & b_n & & b_1 & a \end{bmatrix}$$

and

$$\begin{aligned} B =& (B_{1,1,\dots,1},\dots,B_{M_1-1,1,\dots,1},\dots,B_{1,M_2-1,\dots,M_n-1},\dots,\\ & B_{M_1-1,M_2-1,\dots,M_n-1})^T, \end{aligned}$$

where

$B_{1,1,\dots,1} = -f(x^{1,1,\dots,1}) - \frac{g(x^{0,1,\dots,1})}{(h_1)^2} - \frac{g(x^{1,0,\dots,1})}{(h_2)^2} - \ldots - \frac{g(x^{1,1,\dots,0})}{(h_n)^2}$,

$B_{M_1-1,1,\dots,1} = -f(x^{M_1-1,1,\dots,1}) - \frac{g(x^{M_1,1,\dots,1})}{(h_1)^2} - \frac{g(x^{M_1-1,0,\dots,1})}{(h_2)^2} - \ldots - \frac{g(x^{M_1-1,1,\dots,0})}{(h_n)^2}$,

$B_{1,M_2-1,\dots,M_n-1} = -f(x^{1,M_2-1,\dots,M_n-1}) - \frac{g(x^{0,M_2-1,\dots,M_n-1})}{(h_1)^2} - \ldots - \frac{g(x^{1,M_2-1,\dots,M_n})}{(h_n)^2}$

and

$$B_{M_1-1,M_2-1,\ldots,M_n-1} = -f(x^{M_1-1,M_2-1,\ldots,M_n-1}) - \frac{g(x^{M_1,M_2-1,\ldots,M_n-1})}{(h_1)^2} - \ldots$$

$$- \frac{g(x^{M_1-1,M_2-1,\ldots,M_n})}{(h_n)^2}.$$

The notation $X^T$ represents the transpose of the $X$ matrix. The matrix $A$ is a sparse matrix, with non-zero elements at $2n+1$ diagonals. The values $a, b_1, \ldots, b_n$ that appear on the diagonals are the coefficients of the discretization of $2n+1$ point, with $n \geq 1$, and are given by

$$a = -2\sum_{j=1}^{n} \frac{1}{(h_j)^2} \quad and \quad b_j = \frac{1}{(h_j)^2},$$

for $j = 1, \ldots, n$. Each number $b_j$ vanishes to zero in some positions of the matrix $A$. These positions, which depend on the dimension of Euclidean space and the value of each $M_j$, with $j = 1, \ldots, n$, obey a specific rule and will be seen in Section 4, where an index function enumerates the unknowns of the problem and determines the position of each non-zero element in matrix A.

In the next Section, the convergence of the method will be verified.

## 3 METHOD CONVERGENCE

To show that the method is convergent, consistency and stability will be analyzed. Thus, for the following definitions and results, given a discrete function $V : \Omega_\delta \cup \partial\Omega_\delta \to \mathbb{R}$, consider the operator $\Delta_\delta V : \Omega_\delta \cup \partial\Omega_\delta \to \mathbb{R}$ given by

$$\Delta_\delta V(x) = \sum_{j=1}^{n} \frac{V(x+h_je_j) - 2V(x) + V(x-h_je_j)}{(h_j)^2},$$

for all $x \in \Omega_\delta \cup \partial\Omega_\delta$.

**Definition 3.1.** *The local truncation error, denoted by* $\tau_{i_1,\ldots,i_n}$ *, is defined as*

$$\tau_{i_1,\ldots,i_n} = \Delta_\delta u(x^{i_1,\ldots,i_n}) + f(x^{i_1,\ldots,i_n}). \tag{3.1}$$

The next lemma shows that the local truncation error decreases as we refine the mesh and gives us the convergence rate of this error.

**Lemma 3.1.** *If the solution u of the equation* $(1.1)$ *is differentiable up to order four in* $\Omega$*, with limited fourth-order partial derivatives, then the local truncation error satisfies the inequality*

$$|\tau_{i_1,\ldots,i_n}| \leq C \sum_{j=1}^{n} (h_j)^2,$$

*where C is a positive constant independent of* $h_j$*, with* $j = 1, \ldots, n$.

**Proof.** Applying $2n$ times the Taylor's formula, we get the following developments around the point $x^{i_1,\dots,i_n}$:

$$\begin{aligned} u(x^{i_1,\dots,i_n}+h_je_j) = {} & u(x^{i_1,\dots,i_n})+h_j\frac{\partial u}{\partial x_j}(x^{i_1,\dots,i_n}) \\ & +\frac{(h_j)^2}{2!}\frac{\partial^2 u}{\partial x_j^2}(x^{i_1,\dots,i_n})+\frac{(h_j)^3}{3!}\frac{\partial^3 u}{\partial x_j^3}(x^{i_1,\dots,i_n}) \\ & +\frac{(h_j)^4}{4!}\frac{\partial^4 u}{\partial x_j^4}(x^{i_1,\dots,i_n}+(\rho_j^1-x_j^{i_j})e_j) \end{aligned}$$

and

$$\begin{aligned} u(x^{i_1,\dots,i_n}-h_je_j) = {} & u(x^{i_1,\dots,i_n})-h_j\frac{\partial u}{\partial x_j}(x^{i_1,\dots,i_n}) \\ & +\frac{(h_j)^2}{2!}\frac{\partial^2 u}{\partial x_j^2}(x^{i_1,\dots,i_n})-\frac{(h_j)^3}{3!}\frac{\partial^3 u}{\partial x_j^3}(x^{i_1,\dots,i_n}) \\ & +\frac{(h_j)^4}{4!}\frac{\partial^4 u}{\partial x_j^4}(x^{i_1,\dots,i_n}+(\rho_j^2-x_j^{i_j})e_j), \end{aligned}$$

where $\rho_j^1 \in \left(x_j^{i_j}, x_j^{i_j}+h_j\right)$ and $\rho_j^2 \in \left(x_j^{i_j}-h_j, x_j^{i_j}\right)$, with $j=1,\dots,n$.

By replacing these expansions into $(3.1)$, simplifying the similar terms and using the fact that the function $u$ satisfies the equation $(1.1)$ at the point $x^{i_1,\dots,i_n}$, we get

$$\begin{aligned} \tau_{i_1,\dots,i_n} = {} & \frac{1}{4!}\sum_{j=1}^{n}(h_j)^2\frac{\partial^4 u}{\partial x_j^4}(x^{i_1,\dots,i_n}+(\rho_j^1-x_j^{i_j})e_j) \\ & +\frac{1}{4!}\sum_{j=1}^{n}(h_j)^2\frac{\partial^4 u}{\partial x_j^4}(x^{i_1,\dots,i_n}+(\rho_j^2-x_j^{i_j})e_j). \end{aligned} \tag{3.2}$$

Since the fourth-order partial derivatives are limited, it follows from $(3.2)$ that

$$|\tau_{i_1,\dots,i_n}| \le C\sum_{j=1}^{n}(h_j)^2,$$

for some positive constant $C$, independent of $h_j$, with $j=1,\dots,n$.

Thus, we conclude that the method is of second order. This completes the proof. $\square$

**Definition 3.2.** The global error, denoted by $e_{i_1,\dots,i_n}$, is defined as

$$e_{i_1,\dots,i_n} = u(x^{i_1,\dots,i_n}) - U_{i_1,\dots,i_n}, \tag{3.3}$$

where $U_{i_1,\dots,i_n}$ denotes the numerical solution of the Poisson problem calculated at the point $x^{i_1,\dots,i_n}$.

In the following theorem, we demonstrate that the numerical method is stable. For this, we use the Discrete Maximum Principle [25].

**Theorem 3.1. (Discrete Maximum Principle)** Consider $V:\Omega_\delta\cup\partial\Omega_\delta\to\mathbb{R}$ a discrete function.

$(i)$ *If* $\Delta_\delta V(x) \geq 0$, *for all* $x \in \Omega_\delta$, *then*

$$\max_{x\in\Omega_\delta} V(x) \leq \max_{x\in\partial\Omega_\delta} V(x).$$

$(ii)$ *If* $\Delta_\delta V(x) \leq 0$, *for all* $x \in \Omega_\delta$, *then*

$$\min_{x\in\Omega_\delta} V(x) \geq \min_{x\in\partial\Omega_\delta} V(x).$$

**Proof.**

$(i)$ Suppose the maximum of $V$ does not occur on the boundary of $\Omega_\delta$. This implies that there is $P_0 \in \Omega_\delta$ so that $V(P_0) = M_0$, with $V(P) \leq M_0$, for all $P \in \Omega_\delta$, and $V(P) < M_0$, for any $P \in \partial\Omega_\delta$. Now consider the points given by

$$P_j^1 = P_0 + h_j e_j \text{ and } P_j^2 = P_0 - h_j e_j,$$

where $j = 1,\ldots,n$. Hence,

$$\Delta_\delta V(P_0) = \sum_{j=1}^{n}(h_j)^{-2}[V(P_j^1)+V(P_j^2)] - 2V(P_0)\sum_{j=1}^{n}(h_j)^{-2}.$$

As $\Delta_\delta V(P_0) \geq 0$, we conclude that

$$2V(P_0)\sum_{j=1}^{n}(h_j)^{-2} \leq \sum_{j=1}^{n}(h_j)^{-2}[V(P_j^1)+V(P_j^2)].$$

Consequently,

$$M_0 \leq \left(\sum_{j=1}^{n}(h_j)^{-2}\right)^{-1}\sum_{j=1}^{n}(h_j)^{-2}\frac{V(P_j^1)+V(P_j^2)}{2}. \tag{3.4}$$

Since $V(Q) \leq M_0$, for all $Q \in \Omega_\delta \cup \partial\Omega_\delta$, we conclude that $V(P_j^1) = M_0$ and $V(P_j^2) = M_0$, for all $j = 1,\ldots,n$. In fact, suppose $V(P_j^1) < M_0$ or $V(P_j^2) < M_0$, for some $j = 1,\ldots,n$. It follows from (3.4) that

$$M_0 < \left(\sum_{j=1}^{n}(h_j)^{-2}\right)^{-1}\sum_{j=1}^{n}(h_j)^{-2}\frac{M_0+M_0}{2} = M_0.$$

Which is absurd. Therefore, $V(P_j^1) = M_0$ and $V(P_j^2) = M_0$, for all $j = 1,\ldots,n$.

This argument is repeated for each of the interior points $P_j^1$ and $P_j^2$ instead of $P_0$. By repetition, each point of $\Omega_\delta \cup \partial\Omega_\delta$ appears as one of $P_j^1$ and $P_j^2$, for some corresponding $P_0$.

Thus, we conclude that

$$V(P) = M_0, \text{ for all } P \in \Omega_\delta \cup \partial\Omega_\delta.$$

But this contradicts the fact that $V(P) < M_0$, for all $P \in \partial\Omega_\delta$. Thus, the item $(i)$ is established.

$(ii)$ Firstly, note that

$$max\,[-V(x)] = -min\,V(x) \text{ and } \Delta_\delta\,(-V) = -\Delta_\delta V\;.$$

Suppose $\Delta_\delta V(x) \leq 0$, for all $x \in \Omega_\delta$. So,

$$\Delta_\delta(-V(x)) \geq 0,$$

for all $x \in \Omega_\delta$. By the item $(i)$,

$$\max_{x\in\Omega_\delta}[-V(x)] \leq \max_{x\in\partial\Omega_\delta}[-V(x)]$$

and consequently,

$$\min_{x\in\Omega_\delta} V(x) \geq \min_{x\in\partial\Omega_\delta} V(x).$$

Which completes the proof. □

The next theorem gives us a bound for the solution of the equation (2.2).

**Theorem 3.2. (A Priori estimate)** *Consider* $V : \Omega_\delta \cup \partial\Omega_\delta \to \mathbb{R}$ *a discrete function. Then,*

$$\max_{x\in\Omega_\delta}|V(x)| \leq \max_{x\in\partial\Omega_\delta}|V(x)| + \frac{r_1^2+s_1^2}{2}\max_{x\in\Omega_\delta}|\Delta_\delta V(x)|. \tag{3.5}$$

**Proof.** Consider $\psi : \Omega_\delta \cup \partial\Omega_\delta \to \mathbb{R}$ a discrete function defined by $\psi(x) = \frac{1}{2}x_1^2$, for all $x = (x_1,\ldots,x_n) \in \Omega_\delta \cup \partial\Omega_\delta$. Note that, for all $x \in \Omega_\delta \cup \partial\Omega_\delta$,

$$0 \leq \psi(x) \leq \frac{r_1^2+s_1^2}{2} \textit{ and } \Delta_\delta\psi(x) = 1.$$

Consider $V_+ : \Omega_\delta \cup \partial\Omega_\delta \to \mathbb{R}$ and $V_- : \Omega_\delta \cup \partial\Omega_\delta \to \mathbb{R}$ two discrete functions defined by

$$V_+(x) = V(x) + N_0\psi(x) \;\; \textit{and} \;\; V_-(x) = -V(x) + N_0\psi(x),$$

where

$$N_0 = \max_{x\in\Omega_\delta}|\Delta_\delta V(x)|.$$

So, for every $x \in \Omega_\delta$, we have that $\Delta_\delta V_+(x) = \Delta_\delta V(x) + N_0 \geq 0$ and $\Delta_\delta V_-(x) = -\Delta_\delta V(x) + N_0 \geq 0$.

By applying the item $(i)$ of the Theorem 3.1 to $V_+$, we get

$$\begin{aligned} V(x) &\leq \max_{x\in\Omega_\delta} V_+(x) \leq \max_{x\in\partial\Omega_\delta}[V(x) + N_0\psi(x)] \\ &\leq \max_{x\in\partial\Omega_\delta} V(x) + N_0\frac{r_1^2+s_1^2}{2} \\ &\leq \max_{x\in\partial\Omega_\delta}|V(x)| + \frac{r_1^2+s_1^2}{2}\max_{x\in\Omega_\delta}|\Delta_\delta V(x)|, \end{aligned} \tag{3.6}$$

for all $x \in \Omega_\delta$.

Using the Theorem 3.1, item ($i$), again, we deduce that

$$-V(x) \leq \max_{x\in\partial\Omega_\delta} |V(x)| + \frac{r_1^2+s_1^2}{2} \max_{x\in\Omega_\delta} |\Delta_\delta V(x)|, \tag{3.7}$$

for all $x \in \Omega_\delta$. As a consequence of (3.6) and (3.7), we get (3.5). □

**Note 3.1.** In the proof of Theorem 3.2, we can replace $\frac{r_1^2+s_1^2}{2}$ in (3.5) for $\frac{r_j^2+s_j^2}{2}$, with $j = 2,\ldots,n$, as long as we use one of the discrete functions $\psi_j$, defined by $\psi_j(x) = \frac{x_j^2}{2}$, for $x = (x_1,\ldots,x_n)$, in place of $\psi$.

Now, let's get an estimate for the global error. Then, consider the discrete function $e_{i_1,\ldots,i_n}$ defined in (3.3). By the Theorem 3.2 , we deduce that

$$|e_{i_1,\ldots,i_n}| \leq \max_{\partial\Omega_\delta} |e_{i_1,\ldots,i_n}| + \frac{r_1^2+s_1^2}{2} \max_{\Omega_\delta} |\Delta_\delta e_{i_1,\ldots,i_n}|.$$

Once we have, by the equation (2.3),

$$e_{i_1,\ldots,i_n} = u(x^{i_1,\ldots,i_n}) - g(x^{i_1,\ldots,i_n}) = 0$$

over the boundary of $\Omega_\delta$, we get

$$|e_{i_1,\ldots,i_n}| \leq \frac{r_1^2+s_1^2}{2} \max_{\Omega_\delta} |\Delta_\delta e_{i_1,\ldots,i_n}|. \tag{3.8}$$

From (2.2) and (3.1), we conclude that

$$\begin{aligned}\Delta_\delta e_{i_1,\ldots,i_n} &= \Delta_\delta u(x^{i_1,\ldots,i_n}) - \Delta_\delta U_{i_1,\ldots,i_n}\\ &= \Delta_\delta u(x^{i_1,\ldots,i_n}) + f(x^{i_1,\ldots,i_n})\\ &= \tau_{i_1,\ldots,i_n}.\end{aligned}$$

Using this, (3.8) and the Lemma 3.1, we have that

$$|e_{i_1,\ldots,i_n}| \leq C\sum_{j=1}^{n} (h_j)^2, \tag{3.9}$$

for some positive constant $C$, independent of $h_j$, with $j = 1,\ldots,n$.

**Note 3.2.** We conclude, from Lemma 3.1 and from (3.9), that the numerical method is convergent.

The next result establishes the uniqueness of the solution of the system $AU = B$ and is a consequence of the Discrete Maximum Principle.

**Corollary 3.1.** *The resulting system of linear equations $AU = B$ has a unique solution.*

**Proof.** It is sufficient to show that the only solution of the homogeneous linear system $AU = 0$ is the trivial solution. For this, consider the homogeneous problem

$$\begin{cases} -\Delta u = 0, \text{ in } \Omega, \\ u = 0, \text{ on } \partial\Omega. \end{cases} \tag{3.10}$$

The function $u = 0$ is the only solution to the above problem. Discretizing the problem (3.10), we obtain a homogeneous linear system for the unknowns $U_{i_1,\ldots,i_n}$. Since $U_{i_1,\ldots,i_n}$ is a solution of the difference equation (2.2), with $f = 0$, we conclude that

$$\Delta_\delta U_{i_1,\ldots,i_n} = 0 \text{ in } \Omega_\delta \text{ and } U_{i_1,\ldots,i_n} = 0 \text{ on } \partial\Omega_\delta.$$

By theorem 3.1, item $(i)$, we have

$$\max_{\Omega_\delta} U_{i_1,\ldots,i_n} \leq \max_{\partial\Omega_\delta} U_{i_1,\ldots,i_n} = 0.$$

Applying again the Theorem 3.1, item $(ii)$, we obtain

$$\min_{\Omega_\delta} U_{i_1,\ldots,i_n} \geq \min_{\partial\Omega_\delta} U_{i_1,\ldots,i_n} = 0.$$

Thus, $U_{i_1,\ldots,i_n} = 0$ in $\Omega_\delta$. Therefore, $U = 0$ is the only solution for the linear system $AU = 0$ and the proof of the corollary is complete. □

In the next section the numerical method, given by the equations (2.2) and (2.3), will be implemented computationally.

## 4 COMPUTER IMPLEMENTATION

We consider the function $u$ with domain $[r_1,s_1] \times \cdots \times [r_n,s_n] \subset \mathbb{R}^n$ and the discrete sample of $(M_1+1)\cdot(M_2+1)\cdot\ldots\cdot(M_n+1)$ domain points. Along each axis $x_j$, for $j = 1,\ldots,n$, uniform sampling of points defines segments of length

$$dx_j = \frac{s_j - r_j}{M_j}.$$

Initially, we build a multidimensional matrix $G$ with the dimension $(M_1+1)\cdot(M_2+1)\cdot\ldots\cdot(M_n+1)$, whose elements must be images from the function $u$, known on the domain boundary, that is,

$$G[i_1,i_2,\ldots,i_n] = u(r_1+i_1dx_1, r_2+i_2dx_2,\ldots,r_n+i_ndx_n).$$

Note that only the values of $u$ on the domain boundary are known, that is, we know $G[i_1,i_2,\ldots,i_n]$ only when $i_j = 0$ or $i_j = M_j$, for some $j \in \{1,2,\ldots,n\}$. Within the domain, when $0 < i_j < M_j$ for all $j$, the values $G[i_1,i_2,\ldots,i_n]$ are unknown.

The next step is the construction of a multidimensional matrix $F$ with dimension $(M_1+1)\cdot(M_2+1)\cdot\ldots\cdot(M_n+1)$, whose elements must be values of the Laplacian $\Delta u$, that is,

$$F[i_1,i_2,\ldots,i_n] = \Delta u(r_1+i_1dx_1, r_2+i_2dx_2,\ldots,r_n+i_ndx_n).$$

In this case, only the values of $\Delta u$ in the domain interior are known, that is, we know $F[i_1,i_2,\ldots,i_n]$ only when $0 < i_j < M_j$, for all $j \in \{1,2,\ldots,n\}$.

Finally, each known element of matrix $F$ is related to unknown elements of matrix $G$ by the equation,

$$\begin{aligned} F[i_1,...,i_n] = &\sum_{j=1}^{n} \left\{ b_j G[i_1,...,i_j-1,...,i_n] + b_j G[i_1,...,i_j+1,...,i_n] \right\} \\ &+ aG[i_1,i_2,...,i_n], \end{aligned} \tag{4.1}$$

where $a = -2\sum_{j=1}^{n} \frac{1}{dx_j^2}$ and $b_j = \frac{1}{dx_j^2}$, for $j = 1,...,n$.

The same equation $(4.1)$ can be rewritten considering all the elements of the matrix $G$, in order to obtain an equation with $(M_1-1)...(M_n-1)$ unknowns, given by

$$F[i_1,...,i_n] = \sum_{l_1=1}^{M_1-1} \cdots \sum_{l_n=1}^{M_n-1} \alpha(l_1,...,l_n)G[l_1,...,l_n], \tag{4.2}$$

with $i_j = 1,\ldots,M_j-1$ and $j = 1,\ldots,n$, where the coefficients $\alpha(l_1,...,l_n)$ are null, except for the following cases:

$$\begin{cases} \alpha(i_1,i_2,...,i_n) = a; \\ \alpha(i_1,...,i_j-1,...,i_n) = b_j; \\ \alpha(i_1,...,i_j+1,...,i_n) = b_j. \end{cases} \tag{4.3}$$

To construct a system of linear equations, the unknowns $G[i_1,i_2,\ldots,i_n]$ are enumerated by an index function

$$I: \{1,\ldots,M_1-1\} \times \{1,\ldots,M_2-1\} \times \ldots \times \{1,\ldots,M_n-1\} \to \mathbb{N}$$

defined by

$$\begin{aligned} I(i_1,\ldots,i_n) = &(i_1-1) + (M_1-1)(i_2-1) + (M_1-1)(M_2-1)(i_3-1) + \ldots \\ &+ (M_1-1)(M_2-1)\ldots(M_{n-1}-1)(i_n-1). \end{aligned} \tag{4.4}$$

Thus, we have $(M_1-1)\ldots(M_n-1)$ equations, with the same number of unknowns, determining a linear system $AU = B$, whose solution vector $U$ provides the searched unknowns, that is,

$$G[i_1,i_2,...,i_n] = U[I(i_1,i_2,...,i_n)]. \tag{4.5}$$

The matrix $A$ is a sparse matrix with null elements, except for:

$$\begin{cases} A[I(i_1,i_2,...,i_n),I(i_1,i_2,...,i_n)] = a; \\ A[I(i_1,i_2,...,i_n),I(i_1,...,i_j-1,...,i_n)] = b_j \quad \text{if } \; i_j > 1 \text{ and} \\ A[I(i_1,i_2,...,i_n),I(i_1,...,i_j+1,...,i_n)] = b_j \quad \text{if } \; i_j < M_j - 1, \\ \text{for } j = 1,...,n. \end{cases} \tag{4.6}$$

Note that the index $I(i_1,\dots,i_n)$ indicates that the coefficient $\alpha(i_1,i_2,...,i_n) = a$ of the equation (4.2) associated with the unknown $G[i_1,\dots,i_n]$ will appear in the line $I(i_1,\dots,i_n)$ and column $I(i_1,\dots,i_n)$ of the matrix $A$. The indices $I(i_1,\dots,i_j-1,\dots,i_n)$ and $I(i_1,\dots,i_j+1,\dots,i_n)$ indicate that the coefficients $\alpha(i_1,\dots,i_j-1,\dots,i_n) = b_j$ and $\alpha(i_1,\dots,i_j+1,\dots,i_n) = b_j$, $j = 1,\dots,n$, will appear in the line $I(i_1,\dots,i_n)$ and columns $I(i_1,\dots,i_j-1,\dots,i_n)$ and $I(i_1,\dots,i_j+1,\dots,i_n)$ of the matrix $A$, respectively. In the other positions of the line $I(i_1,\dots,i_n)$ the coefficients $\alpha(l_1,...,l_n)$ of the equation (4.2) are all null. Figure 1 illustrates the final position of each element in matrix A, for the case $n = 3$.

Now, notice that the vector $B$, which has $(M_1-1)\dots(M_n-1)$ independent terms, is not given just by values of the Laplacian $F[i_1,i_2,\dots,i_n]$, but known boundary values $G[i_1,i_2,\dots,i_n]$, for $i_j = 0$ or $i_j = M_j$ , are incorporated into the independent terms as follows:

$$\begin{aligned} B[I(i_1,i_2,\dots,i_n)] &= F[i_1,i_2,\dots,i_n] \\ &\quad - \sum_{j=1}^{n} s_j G[i_1,\dots,i_j-1,\dots,i_n] \\ &\quad - \sum_{j=1}^{n} t_j G[i_1,\dots,i_j+1,\dots,i_n], \end{aligned}$$

where the coefficients $s_j$ and $t_j$, for $j = 1,\dots,n$, are all null, except in the cases:

$$\begin{cases} s_j = b_j \quad \text{if } \; i_j = 1; \\ t_j = b_j \quad \text{if } \; i_j = M_j - 1. \end{cases}$$

Using the implementation details above, below is a code example, in Python, for solving the following Problem in $\mathbb{R}^3$:

$$\begin{cases} \Delta u = 12x^2 + 12y^2 + 12z^2, \text{ in } \Omega, \\ u = x^4 + y^4 + z^4, \text{ on } \partial\Omega, \end{cases}$$

where $\Omega = (-1,1) \times (-1,1) \times (-1,1)$.

```
import numpy as np
from scipy.sparse import lil_matrix
from scipy.sparse.linalg import spsolve

M1=M2=M3= 50
r1=r2=r3= -1
s1=s2=s3=  1
dx1=(s1-r1)/M1
dx2=(s2-r2)/M2
dx3=(s3-r3)/M3
a=-2/(dx1**2)-2/(dx2**2)-2/(dx3**2)
b1 = 1/(dx1**2)
b2 = 1/(dx2**2)
b3 = 1/(dx3**2)
S = (M1-1)*(M2-1)*(M3-1) # number of elements interior to domain
A = lil_matrix((S,S),dtype='float32') # Coefficient matrix
B = np.zeros((S, 1)) # Vector of independent values

G = np.zeros((M1+1,M2+1,M3+1))
F = np.zeros((M1+1,M2+1,M3+1))

def u(x,y,z):
    return x**4+y**4+z**4
def f(x,y,z):
    return 12*x**2+12*y**2+12*z**2
def I(i1,i2,i3):
    return (M1-1)*(M2-1)*(i3-1) + (M1-1)*(i2-1) + (i1-1)

for i1 in range(0,M1+1):
    for i2 in range(0, M2+1):
        for i3 in range(0, M3+1):
            if i1==0 or i2==0 or i3==0 or i1==M1 or i2==M2 or i3==M3:
               G[i1,i2,i3]=u(s1+i1*dx1,s2+i2*dx2,s3+i3*dx3)
            else:
               F[i1,i2,i3]=f(s1+i1*dx1,s2+i2*dx2,s3+i3*dx3)

for i1 in range(1,M1):
    for i2 in range(1, M2):
        for i3 in range(1, M3):
            A[I(i1,i2,i3),I(i1,i2,i3)]=a
            if i1>1:       A[I(i1,i2,i3),I(i1-1,i2,i3)]=b1
            if i2>1:       A[I(i1,i2,i3),I(i1,i2-1,i3)]=b2
            if i3>1:       A[I(i1,i2,i3),I(i1,i2,i3-1)]=b3
            if i1<(M1-1):  A[I(i1,i2,i3),I(i1+1,i2,i3)]=b1
            if i2<(M2-1):  A[I(i1,i2,i3),I(i1,i2+1,i3)]=b2
            if i3<(M3-1):  A[I(i1,i2,i3),I(i1,i2,i3+1)]=b3
            B[I(i1,i2,i3)]= F[i1][i2][i3]
            if i1==1:       B[I(i1,i2,i3)]=B[I(i1,i2,i3)]-b1*G[i1-1][i2][i3]
            if i2==1:       B[I(i1,i2,i3)]=B[I(i1,i2,i3)]-b2*G[i1][i2-1][i3]
            if i3==1:       B[I(i1,i2,i3)]=B[I(i1,i2,i3)]-b3*G[i1][i2][i3-1]
            if i1==(M1-1):  B[I(i1,i2,i3)]=B[I(i1,i2,i3)]-b1*G[i1+1][i2][i3]
            if i2==(M2-1): B[I(i1,i2,i3)]=B[I(i1,i2,i3)]-b2*G[i1][i2+1][i3]
            if i3==(M3-1): B[I(i1,i2,i3)]=B[I(i1,i2,i3)]-b3*G[i1][i2][i3+1]

U = spsolve(A,B) # Find solution vector

for i1 in range(1,M1):  # Construct solution for interior points
    for i2 in range(1, M2):
        for i3 in range(1, M3):
            G[i1][i2][i3]=U[I(i1,i2,i3)]
```

## 5 EXPERIMENTS

In this section, we present numerical experiments to evaluate the effectiveness of the proposed method. To illustrate the implementation and computational performance, we consider a set of test cases. In each case, the integration domain is a rectangular block discretized using a uniform mesh. The tables below display the average error, $(e)$, calculated according to the following formula:

$$e = \frac{\sum_{i_1=0}^{M_1}\sum_{i_2=0}^{M_2}\cdots\sum_{i_n=0}^{M_n} |G(i_1,\dots,i_n) - u(r_1+i_1dx_1,\dots,r_n+i_ndx_n)|}{(M_1+1).(M_2+1)\dots(M_n+1)}, \tag{5.1}$$

where $G$ is the approximate solution by the method and $u$ is the exact solution.

We also show the evolution of the error in the numerical solution, for each problem, as the resolution of the considered mesh increases. The solution for the linear system $AU = B$ was obtained using *spsolve* package for sparse matrix in Python. The numerical solution for the examples 1 and 2, considered next, is obtained using the procedure in the section 4.

**Example 1.** Consider the following problem in $\mathbb{R}^2$

$$\begin{cases} \Delta u = -sin(x) - sin(y), \text{ in } \Omega, \\ \\ u = sin(x) + sin(y), \text{ on } \partial\Omega, \end{cases}$$

where $\Omega = (-\pi,\pi) \times (-\pi,\pi)$. The exact solution to this problem is given by $u(x,y) = sin(x) + sin(y)$.

In Table 1 we present values of the average error, given by the formula $(5.1)$, for some resolutions $M \times M$ of the mesh, with $M = 4, 8, 16, 32, 64, 128, 256, 512, 1024$. In this table one can observe the convergence of the error. The curve fitting for this data indicates that the convergence is approximately of order $O(1.978)$, which is close to quadratic, as expected.

Table 1: Error in the solution of the problem Example 1, for some resolutions.

| Resolution | Error $(e)$ |
|---|---|
| 4x4 | 9.1810e-02 |
| 8x8 | 2.6437e-02 |
| 16x16 | 6.8522e-03 |
| 32x32 | 1.7279e-03 |
| 64x64 | 4.3293e-04 |
| 128x128 | 1.0829e-04 |
| 256x256 | 2.7080e-05 |
| 512x512 | 6.7730e-06 |
| 1024x1024 | 1.6960e-06 |

**Example 2.** Consider the following problem in $\mathbb{R}^3$

$$\begin{cases} \Delta u = 12x^2 + 12y^2 + 12z^2, \text{ in } \Omega, \\ \\ u = x^4 + y^4 + z^4, \text{ on } \partial\Omega, \end{cases}$$

where $\Omega = (-1,1) \times (-1,1) \times (-1,1)$.

In this case, with the domain in $\mathbb{R}^3$, the resolution $M \times M \times M$ leads to the sampling of $M^3$ points from the domain. Due to the limitation of memory allocation capacity, we only considered up to resolution $50 \times 50 \times 50$, but it is already possible to notice the convergence (Table 2). For this data, the convergence order is approximately $O(1.915)$, which is also close to quadratic, as expected. Note that in this case the matrix $A$, from the system $AU = B$, has $49^3 = 117,649$ rows and columns.

Table 2: Error in the solution of the problem Example 2, for some resolutions.

| Resolution | Error $(e)$ |
|---|---|
| 5 x 5 x 5 | 6.1741e-02 |
| 10 x 10 x 10 | 1.8259e-02 |
| 15 x 15 x 15 | 8.3816e-03 |
| 20 x 20 x 20 | 4.7689e-03 |
| 30 x 30 x 30 | 2.1370e-03 |
| 40 x 40 x 40 | 1.2055e-03 |
| 50 x 50 x 50 | 7.7261e-04 |

Figure 1 illustrates the matrix $A$ of the system $AU = B$ for the mesh with resolution $5 \times 5 \times 5$. In this case $A$ has $4^3 = 64$ rows and columns. The diagonals corresponding to the coefficients $a, b_1, b_2, b_3$ are indicated in the figure, as automatically defined in Equation 4.6.

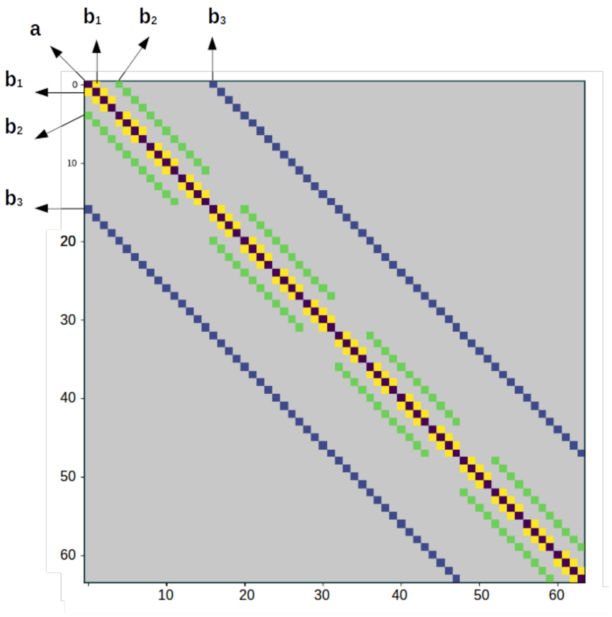

Figure 1: Illustration of the coefficient matrix of the system $AU = B$ with indication of the diagonals corresponding to the coefficients $a, b_1, b_2, b_3$.

Numerical experiments confirm the consistency and stability of the method, ie, the convergence of the numerical solution to the exact solution of the problem while the mesh is refined as demonstrated in the section 3. The code in Python presented in the section 4 uses the index function to solve the problem of Example 2, in the dimension $n = 3$, and can be modified to consider larger dimensions, as detailed in the same Section.

## 6 CONCLUSIONS

This article presents the numerical solution for the $n$-dimensional Poisson problem, for all $n \geq 1$. We propose a new approach based on an index function that plays a central role in obtaining the numerical solution, as it facilitates generalization and supports computational implementation in any dimension. The index function simplifies the construction of both the coefficient matrix and the right-hand side vector of the sparse linear system that arises in the modeling process. Moreover, this function can be easily adapted, since the access path to array elements is not unique and depends on the chosen enumeration, without affecting the final solution. This concept can be extended to obtain numerical solutions for other elliptic problems.

As future work, we intend to investigate how the use of the index function can be adapted and extended to more complex scenarios, including problems with irregular boundaries and nonlinear Poisson-type equations.

## REFERENCES

[1] B.M. Averick & J.M. Ortega. Fast solution of nonlinear Poisson-type equations. *SIAM Journal on Scientific Computing*, **14**(1) (1993), 44–48.

[2] K. Balakrishnan & P.A. Ramachandran. A particular solution Trefftz method for non-linear Poisson problems in heat and mass transfer. *Journal of Computational Physics*, **150**(1) (1999), 239–267.

[3] K. Balakrishnan & P.A. Ramachandran. Osculatory interpolation in the method of fundamental solution for nonlinear Poisson problems. *Journal of Computational Physics*, **172**(1) (2001), 1–18.

[4] R.F. Boisvert. Families of high order accurate discretizations of some elliptic problems. *SIAM Journal on Scientific and Statistical Computing*, **2**(3) (1981), 268–284.

[5] W.R. Bowen & P.M. Williams. Finite difference solution of the 2-dimensional Poisson–Boltzmann equation for spheres in confined geometries. *Colloids and Surfaces A: Physicochemical and Engineering Aspects*, **204**(1-3) (2002), 103–115.

[6] J.A. Cuminato & M.M. Junior. "Discretização de equações diferenciais parciais: técnicas de diferenças finitas". SBM, Rio de Janeiro, 1 ed. (2013).

[7] L.P. da Silva, M.A.V. Pinto & L.K. Araki. Higher-order methods for the Poisson equation obtained with geometric multigrid and completed Richardson extrapolation. *Computational and Applied Mathematics*, **43**(7) (2024), 395.

[8] L.P. da Silva, B.B. Rutyna, A.R. Santos Righi & M.A. Villela Pinto. High order of accuracy for Poisson equation obtained by grouping of repeated Richardson extrapolation with fourth order schemes. *Computer Modeling in Engineering & Sciences*, **128**(2) (2021), 699–715.

[9] G.E. Fasshauer. Newton iteration with multiquadrics for the solution of nonlinear PDEs. *Computers & Mathematics with Applications*, **43**(3-5) (2002), 423–438.

[10] J.H. Ferziger & M. Perić. "Computational methods for fluid dynamics". Springer (2002).

[11] M.M. Gupta, R.P. Manohar & J.W. Stephenson. A single cell high order scheme for the convection-diffusion equation with variable coefficients. *International Journal for Numerical Methods in Fluids*, **4**(7) (1984), 641–651.

[12] G.M.A.M.L. Guta. Solution of Two Dimensional Poisson Equation Using Finite Difference Method with Uniform and Non-uniform Mesh Size. (2019).

[13] H. Johansen & P. Colella. A Cartesian grid embedded boundary method for Poisson's equation on irregular domains. *Journal of Computational Physics*, **147**(1) (1998), 60–85.

[14] J.J. Kasab, S.R. Karur & P. Ramachandran. Quasilinear boundary element method for nonlinear Poisson type problems. *Engineering Analysis with Boundary Elements*, **15**(3) (1995), 277–282.

[15] L. Lapidus & G.F. Pinder. "Numerical solution of partial differential equations in science and engineering". John Wiley & Sons (2011).

[16] Z. Li, C. Pao & Z. Qiao. A finite difference method and analysis for 2D nonlinear Poisson–Boltzmann equations. *Journal of Scientific Computing*, **30** (2007), 61–81.

[17] B. Mebrate, P.R. Koya *et al.* Numerical solution of a two dimensional poisson equation with dirichlet boundary conditions. *American Journal of Applied Mathematics*, **3**(6) (2015), 297–304.

[18] H. Moghaderi, M. Dehghan & M. Hajarian. A fast and efficient two-grid method for solving d-dimensional poisson equations. *Numerical Algorithms*, **72** (2016), 483–537.

[19] F.M. Okoro & A.E. Owoloko. Compact finite difference schemes for Poisson equation using direct solver. *Journal of Mathematics and Technology, ISSN*, (2010), 2078–0257.

[20] P. Pandey. A higher accuracy exponential finite difference method for the numerical solution of second order elliptic partial differential equations. *J. Math. Comput. Sci.*, **3**(5) (2013), 1325–1334.

[21] J.B. Rosser. Nine-point difference solutions for Poisson's equation. *Computers & Mathematics with Applications*, **1**(3-4) (1975), 351–360.

[22] J.B. Rosser. Finite-difference solution of Poisson's equation in rectangles of arbitrary proportions. *Zeitschrift für angewandte Mathematik und Physik ZAMP*, **28** (1977), 185–196.

[23] G.H. Shortley & R. Weller. The numerical solution of Laplace's equation. *Journal of Applied Physics*, **9**(5) (1938), 334–348.

[24] G.D. Smith. "Numerical solution of partial differential equations: finite difference methods". Oxford university press (1985).

[25] J.C. Strikwerda. "Finite difference schemes and partial differential equations". SIAM (2004).

[26] H. Wang, Q.H. Qin & X.P. Liang. Solving the nonlinear Poisson-type problems with F-Trefftz hybrid finite element model. *Engineering analysis with boundary elements*, **36**(1) (2012), 39–46.

[27] Y. Wang & J. Zhang. Sixth order compact scheme combined with multigrid method and extrapolation technique for 2D Poisson equation. *Journal of Computational Physics*, **228**(1) (2009), 137–146.

[28] T. Zhu, J. Zhang & S.N. Atluri. A meshless local boundary integral equation (LBIE) method for solving nonlinear problems. *Computational mechanics*, **22**(2) (1998), 174–186.

## Acknowledgments

The authors are grateful to the support of Unimontes and IFNMG.

## Data availability

Do not apply.

**Associate editor:** Pedro Lima

**How to cite**

A.V.N. Vieira, A.W. Vieira & N. da H. Lisboa. A Dimension-Independent Finite Difference Method for the Poisson Equation Using an Index Mapping Function. *Trends in Computational and Applied Mathematics*, **26**(2025), e01704. doi: 10.5540/tcam.2025.026.e01704.