Trends in Computational and Applied Mathematics

Trends in Computational and Applied Mathematics, **26** (2025), e01833
Sociedade Brasileira de Matemática Aplicada e Computacional
Online version ISSN 2676-0029 www.scielo.br/tcam
ORIGINAL ARTICLE
doi: 10.5540/tcam.2025.026.e01833

On the relative merits of interpolation schemes for the immersed boundary method: a case study with the heat equation

W. C. LESINHOVSKI[1*], N. L. DIAS [2],
L. S. FREIRE [3] and A. C. F. S. JESUS[4]

Received on September 16, 2024 / Accepted on November 12, 2024

ABSTRACT. In this work, three implementation options for the ghost cell immersed boundary method are compared. These options are alternatives to a rather common implementation of the method that is susceptible to numerical instability in the calculation of the bilinear interpolation in some cases. The method is implemented for a second-order spatial discretization of the heat equation in a non-rectangular domain and the errors for each option are analyzed in terms of the order of accuracy and the way they are distributed in the domain. The best option, which was the only one to maintain the second order of convergence of the discretization, is to consider non-symmetric extrapolation with bilinear interpolation, instead of using inverse distance weighted interpolation with symmetric or non-symmetric extrapolation.

Keywords: Immersed boundary method, bilinear interpolation, inverse distance weighted interpolation, heat equation

1 INTRODUCTION

One of the main motivations for using numerical methods for the solution of partial differential equations is the possibility of representing complex contour geometries. However, this creates the need to specify the boundary conditions in these geometries. A commonly adopted approach involves utilizing a grid that conforms to the physical contour. However, even for simple geometries, generating a good-quality body-conformal grid can be an iterative process requiring significant input from the person generating the grid. As the geometry becomes more complicated, the task of generating an acceptable grid becomes increasingly difficult [19].

*Corresponding author: Willian Carlos Lesinhovski – E-mail: willian.carlos@ufpr.br
[1]Universidade Federal do Paraná, Programa de Pós-Graduação em Engenharia Ambiental, Av. Cel. Francisco H. dos Santos, 100, 81530-000, Curitiba, PR, Brasil – E-mail: willian.carlos@ufpr.br https://orcid.org/0000-0003-1402-5957
[2]Universidade Federal do Paraná, Departamento de Engenharia Ambiental, Av. Cel. Francisco H. dos Santos, 100, 81530-000, Curitiba, PR, Brasil – E-mail: nldias@ufpr.br https://orcid.org/0000-0002-9770-8595
[3]Universidade de São Paulo, Instituto de Ciências Matemáticas e de Computação, ICMC, Av. Trabalhador São Carlense, 400, 13566-590, São Carlos, SP, Brasil – E-mail: liviafreire@usp.br https://orcid.org/0000-0002-8992-3869
[4]Universidade de São Paulo,Instituto de Ciências Matemáticas e de Computação, ICMC, Av. Trabalhador São Carlense, 400, 13566-590, São Carlos, SP, Brasil – E-mail: carolinefelix@usp.br https://orcid.org/0000-0003-0968-4296

The Immersed Boundary Method (IBM) first proposed by Peskin in [23] is an alternative to the approach described above. The method employs a Cartesian grid that is not aligned with the physical boundary, offering versatility and efficiency for solving partial differential equations in complex geometries [29]. There are several variations of the IBM, but in general boundary conditions are imposed on the problem through a forcing term added to the governing equation [12]. Over the past two decades, the IBM has gained wide acceptance and has been applied in diverse contexts in fluid mechanics and beyond, such as electromagnetism, solid mechanics and Brownian motion, among others [29].

In the first version of the IBM developed by Peskin for blood flow simulations the forcing term for the boundary condition is calculated via Hooke's Law [23]. Since then new versions and applications have been proposed. In [10] a new way of calculating the forcing term is proposed using an integral formulation to simulate the flow of air passing through a circular cylinder with a moderate Reynolds number. In 1997, Mohd-Yusof [20] created a new class of IBM called the *discrete forcing approach* by calculating the forcing term directly through the numerical solution, without the need for user-defined parameters in the forcing term to impose the boundary condition. Within this class there is the ghost cell approach, used in [18] for the solution of the Reynolds-averaged Navier–Stokes equations (RANS). This approach implements ghost points outside the physical domain and close to the boundary to impose the boundary conditions. In [3] the *discrete forcing approach* is combined with a level set method to apply a wall model in the stress tensor in a region close to the boundary for a Large Eddy Simulation (LES) of turbulence.

Although immersed boundary methods are often used in fluid mechanics to solve the Navier-Stokes equations, they can also be used in other contexts. For example, [2] uses the IBM and a finite difference discretization to solve the heat equation associated with Stefan's problem. [8] and [16] present finite difference schemes for solving the heat equation in non-rectangular domains using level-set functions to determine the boundary of the problem. The immersed boundary method can also be used with finite volume discretization as proposed in [26] and approaches similar to Peskin's original proposal are presented in [30] and [14]. Furthermore, the IBM can also be used to solve the Poisson equation [6, 8, 26].

In the ghost cell method, it is necessary to extrapolate the values of the variables to the ghost points, which can be done in the coordinate axes directions as in [9] or in the normal direction to the boundary as in [28]. The way in which the extrapolation is carried out has implications for the accuracy of the numerical method and may reduce its order of convergence. This fact is well illustrated in [8] where linear, quadratic and cubic extrapolations provide convergence orders 2, 3 and 4, respectively for a fourth order spatial discretization for the Poisson and heat equations. In the case of extrapolation normal to the boundary, the most common approach is to reflect the ghost point across the boundary into the domain in a symmetric manner and to estimate the value of the variable at the reflected point using an interpolation scheme. In general, bilinear and trilinear interpolations are used for two and three dimensions, respectively. In [28] it is shown that this extrapolation scheme combined with bilinear interpolation maintains convergence of order 2 for a numerical scheme for the Navier-Stokes equations.

However, as mentioned in [21] and [17], in the standard ghost cell approach the bilinear interpolation may present instability in some cases due to ill-conditioning of the Vandermonde matrix involved, especially when the point to be interpolated is very close to the boundary. To overcome this problem, in [17] it is proposed to use an inverse distance weighted interpolation which, despite not presenting problems of numerical instability, has lower accuracy than bilinear interpolation. Another proposal to avoid instability with the Vandermonde matrix is presented in [21], where the point to be interpolated is far from the boundary so that the four neighboring points in the grid belong to the domain. In this case, bilinear interpolation is calculated by a simple and robust algorithm as presented in [24] without the necessity to solve a linear system associated with the Vandermonde matrix, and the numerical scheme maintains second order convergence. Clearly, the issue of the preferred interpolation scheme for the IBM and its relation to the order of convergence needs to be examined further.

In view of that, the aim of this work is to compare three versions of the ghost cell method to avoid instability when the points to be interpolated are very close to the boundary. These three versions are based on those presented in [21] and [17], namely: symmetric extrapolation with inverse distance weighted interpolation; non-symmetric extrapolation with inverse distance weighted interpolation and non-symmetric extrapolation with bilinear interpolation. For comparison purposes, the method is used to solve the heat equation with Dirichlet boundary conditions, which allows us to focus on the essence of the method, analyzing a problem for which there is an analytical solution for a non-rectangular boundary, which can be used for comparison with the numerical solution obtained. The three versions are compared by analyzing the order of convergence, the distribution of errors across the domain, as well as aspects regarding their implementation. The method with its three versions of the ghost cell method was implemented in the Chapel programming language (`https://chapel-lang.org/`) in parallel.

This paper is structured as follows: in section 2, the finite difference method and the three versions of the ghost cell method used for the numerical solution are presented; in section 3, the analytical solutions used for comparison with numerical solutions are presented and in section 4 the numerical results obtained for the three versions of the method are presented, analyzed and compared. Finally, our conclusions are presented in section 5.

2 NUMERICAL METHOD

Consider the heat equation (2.1) in arbitrary units with initial and boundary conditions (2.2) and (2.3),

$$u_t = \alpha\nabla^2 u \qquad \text{in } \Omega, \tag{2.1}$$

$$u(0,x,y) = w(x,y) \qquad \text{in } \Omega, \tag{2.2}$$

$$u(t,x,y) = g(x,y) \qquad \text{in } \partial\Omega, \tag{2.3}$$

where u is the temperature as a function of time t and the position (x,y); w is the initial condition; g are the boundary conditions; the subscript t indicates the partial derivative with respect to time,

∇^2 is the Laplacian operator and α is the thermal diffusivity coefficient. To solve this equation on a Cartesian grid with the IBM, we consider a rectangular domain D that contains $\Omega \subset \mathbb{R}^2$ as illustrated in Figure 1. The boundary condition in $\partial\Omega$ is imposed through a forcing term f that must be calculated at each iteration of the algorithm using the ghost cell method in a similar way to that done in [17], but with an approach using a signed distance function, inspired by [3]. To do so, we modify (2.1)–(2.3) to

$$u_t = \alpha\nabla^2 u + f \qquad \text{in } D, \tag{2.4}$$

$$u(t,x,y) = \overline{g}(x,y) \qquad \text{in } \partial D. \tag{2.5}$$

Note that the boundary condition $\overline{g}(x,y)$ is being imposed on the boundary of the rectangular domain D.

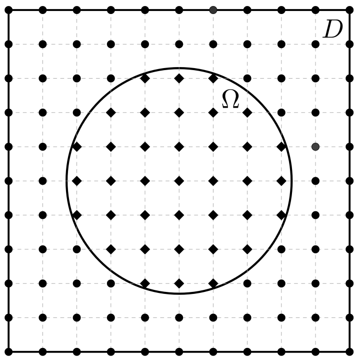


Figure 1: Illustration of grid points in Ω and D.

For the discretization of (2.4) by the classical second order finite difference scheme for the Laplacian operator and the forward Euler method for the time derivative, consider a Cartesian grid of points (x_i, y_j) in D and denote by u^n_{ij} the approximation of $u(t_n, x_i, y_j)$ with $x_i = i\Delta x$, $y_j = j\Delta y$, $t_n = n\Delta t$, and $\Delta x = x_{i+1} - x_i$, $\Delta y = y_{j+1} - y_j$ and $\Delta t = t_{n+1} - t_n$ fixed. Using the notation for numerical differentiation from [13], we discretize (2.4) in the form

$$\frac{\delta u^n_{ij}}{\delta t} = \overline{\nabla}^2 u^n_{ij} + f^n_{ij}, \tag{2.6}$$

where for each point (t_n, x_i, y_j)

$$\frac{\delta u^n_{ij}}{\delta t} = \frac{u^{n+1}_{ij} - u^n_{ij}}{\Delta t}, \tag{2.7}$$

$$\overline{\nabla}^2 u^n_{ij} = \frac{\delta^2 u^n_{ij}}{\delta x^2} + \frac{\delta^2 u^n_{ij}}{\delta y^2}, \tag{2.8}$$

$$\frac{\delta^2 u^n_{ij}}{\delta x^2} = \frac{u^n_{i+1,j} - 2u^n_{i,j} + u^n_{i-1,j}}{\Delta x^2}, \tag{2.9}$$

$$\frac{\delta^2 u^n_{ij}}{\delta y^2} = \frac{u^n_{i,j+1} - 2u^n_{i,j} + u^n_{i,j-1}}{\Delta y^2}. \tag{2.10}$$

In computational fluid mechanics, the fractional step method created by Chorin [5] is widely used for the numerical solution of the Navier-Stokes equations [1]. In the simple case of the heat equation it can be implemented by creating an intermediate step (predictor) u^*_{ij} at each time step as follows:

$$\frac{\delta u^n_{ij}}{\delta t} = \frac{u^{n+1}_{ij} - u^n_{ij}}{\Delta t} = \frac{u^{n+1}_{ij} - u^*_{ij} + u^*_{ij} - u^n_{ij}}{\Delta t} = \alpha \overline{\nabla}^2 u^n_{ij} + f^n_{ij}, \tag{2.11}$$

which we split into two parts:

$$\frac{u^*_{ij} - u^n_{ij}}{\Delta t} = \alpha \overline{\nabla}^2 u^n_{ij}, \tag{2.12}$$

$$\frac{u^{n+1}_{ij} - u^*_{ij}}{\Delta t} = f^n_{ij}. \tag{2.13}$$

As presented in [7], the stability condition for this numerical scheme is

$$\alpha \Delta t \left(\frac{1}{\Delta x^2} + \frac{1}{\Delta y^2} \right) \leqslant \frac{1}{2}. \tag{2.14}$$

The forcing f^n_{ij} must be calculated in order to modify the value of u in ghost points to impose the boundary condition. The set of ghost points is denoted by Γ and is formed by the points (x_i, y_j) of the mesh such that at least one of the points (x_{i-1}, y_j), (x_{i+1}, y_j), (x_i, y_{j-1}) or (x_i, y_{j+1}) belongs to Ω. For each point in Γ outside the physical domain of the problem, auxiliary values v^{n+1}_{ij} must be calculated through extrapolation using the intermediate step u^* and the boundary condition (2.3) to obtain f^n_{ij} as

$$f^n_{ij} = \begin{cases} \frac{v^{n+1}_{ij} - u^n_{ij}}{\Delta t} - \alpha \overline{\nabla}^2 u^n_{ij}, & \text{in}\, \Gamma, \\ 0, & \text{outside}\, \Gamma. \end{cases} \tag{2.15}$$

In the classical implementation of the ghost cell method, for each ghost point $\mathbf{x}_G \in \Gamma$ one must calculate its projection $\mathbf{x}_B$ on the boundary $\partial\Omega$ and its reflection $\mathbf{x}_P$ in the domain Ω so that $\mathbf{x}_B$ is the midpoint of the segment connecting $\mathbf{x}_G$ and $\mathbf{x}_P$. In [17, 18, 28] this is done by considering an approximation of the boundary by a polygonal line and calculating $\mathbf{x}_P$ so that the segment connecting $\mathbf{x}_G$ and $\mathbf{x}_P$ is perpendicular to the polygonal line. Another possibility used, for example, in [4] and in the present work is to consider a level-set function and its gradient to calculate the points $\mathbf{x}_B$ and $\mathbf{x}_P$ due to the ease of implementation. To do this, define $\varphi_{ij} = \varphi(x_i, y_j)$ as the signed distance function from each point of the Cartesian mesh to the boundary $\partial\Omega$, where φ is negative outside of Ω and positive in Ω. Also denote by $\overline{\nabla}\varphi$ the finite difference approximation of $\nabla\varphi$. For each ghost point $\mathbf{x}_G$ of the Cartesian mesh, we calculate its projection $\mathbf{x}_B$ on the boundary in the direction of $\overline{\nabla}\varphi(\mathbf{x}_G)$ as

$$\mathbf{x}_B = \mathbf{x}_G + |\varphi(\mathbf{x}_G)| \overline{\nabla}\varphi(\mathbf{x}_G), \tag{2.16}$$

and we calculate the reflection of $\mathbf{x}_G$ in Ω through $\mathbf{x}_B$ as

$$\mathbf{x}_P = \mathbf{x}_G + 2|\varphi(\mathbf{x}_G)| \overline{\nabla}\varphi(\mathbf{x}_G) = \mathbf{x}_B + |\varphi(\mathbf{x}_G)| \overline{\nabla}\varphi(\mathbf{x}_G). \tag{2.17}$$

In several works that use the ghost cell method, such as [18] and [17], it is common to use a simple arithmetic mean to calculate $v^{n+1}(\mathbf{x}_G)$:

$$\begin{aligned} u^{n+1}(\mathbf{x}_B) &= \frac{v^{n+1}(\mathbf{x}_G)+u^*(\mathbf{x}_P)}{2} \Rightarrow \\ v^{n+1}(\mathbf{x}_G) &= 2u^{n+1}(\mathbf{x}_B)-u^*(\mathbf{x}_P), \\ v^{n+1}(\mathbf{x}_G) &= 2g(\mathbf{x}_B)-u^*(\mathbf{x}_P). \end{aligned} \tag{2.18}$$

Note that $\mathbf{x}_P$ in general is not a grid point (x_i, y_j) and therefore the value of $u^*(\mathbf{x}_P)$ must be estimated using some interpolation method as shown below. Note also that the value of $u^{n+1}(\mathbf{x}_B)$ is given by $g(x,y)$, imposing the boundary condition. The classical point configuration is illustrated in Figure 2 (a).

In the implementation proposed by [21], instead of making a symmetric reflection of the point $\mathbf{x}_G$ to obtain $\mathbf{x}_P$, it is considered that the point $\mathbf{x}_P$ is at a distance $\beta = \sqrt{2}\Delta x$ from $\mathbf{x}_B$, that is

$$\mathbf{x}_P = \mathbf{x}_G + (\beta + |\varphi(\mathbf{x}_G)|)\overline{\nabla}\varphi(\mathbf{x}_G) = \mathbf{x}_B + \beta\overline{\nabla}\varphi(\mathbf{x}_G). \tag{2.19}$$

Different from that proposed in [21], in this work the extrapolation is done by calculating a linear function that passes through $\mathbf{x}_P$ and $\mathbf{x}_B$ to then estimate the value in $\mathbf{x}_G$ similar to the extrapolation presented in [9]. Therefore, the expression for $v^{n+1}(\mathbf{x}_G)$ is given by

$$v^{n+1}(\mathbf{x}_G) = \left(\frac{\beta + |\varphi(\mathbf{x}_G)|}{\beta}\right) g(\mathbf{x}_B) - \frac{|\varphi(\mathbf{x}_G)|}{\beta} u^*(\mathbf{x}_P). \tag{2.20}$$

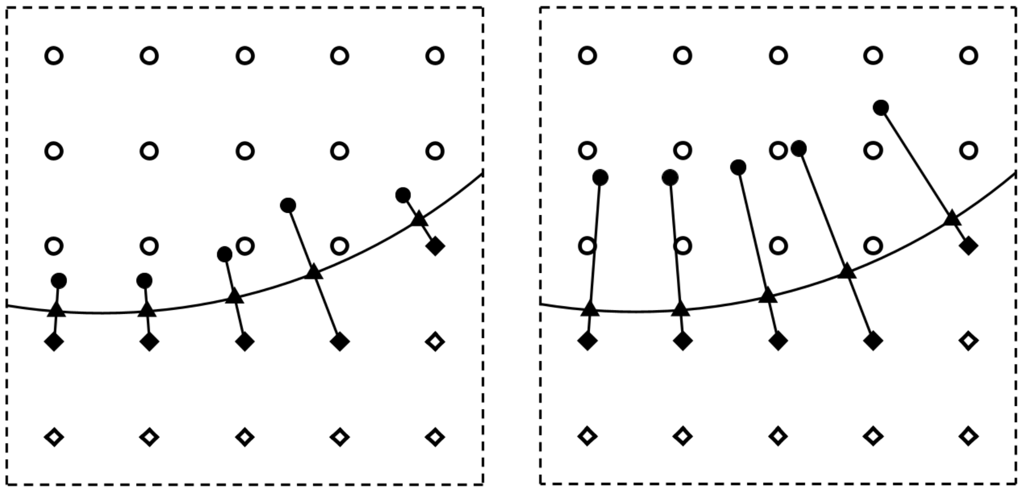
The advantage of choosing $\mathbf{x}_P$ further from the boundary is that the four closest grid points are within the domain Ω if the boundary is smooth enough. Therefore, bilinear interpolation is not subject to ill-conditioning problems. The point configuration as proposed by [21] is illustrated in Figure 2 (b). The detailed error analysis of the two versions of the method using the two interpolation schemes is presented in section 4.

In summary, the algorithm for imposing boundary conditions on the $\partial\Omega$ boundary using the IBM proceeds as follows:

1. Compute $u^*_{ij} = u^n_{ij} + \alpha\Delta t\overline{\nabla}^2 u^n_{ij}$ in $\Omega \cup \Gamma$;
2. compute v^{n+1}_{ij} by extrapolation in Γ;
3. compute f^n_{ij} by (2.15) in $\Omega \cup \Gamma$;
4. compute $u^{n+1}_{ij} = u^*_{ij} + \Delta t f^n_{ij}$ in $\Omega \cup \Gamma$.

2.1 Interpolation schemes

In the present work we consider the interpolation methods used in [17], namely: inverse distance weighted interpolation and bilinear interpolation (in the bidimensional case). We emphasize that other methods can be used, such as least squares interpolation [22].



(a) Points for symmetrical extrapolation (b) Points for non-symmetrical extrapolation

Figure 2: Illustration of points in the present immersed boundary method implementations, where ○ are grid points (x_i, y_j) in Ω; $\Diamond$ are grid points (x_i, y_j) outside Ω; ⧫ are ghost points $\mathbf{x}_G$; ▲ are boundary points $\mathbf{x}_B$; ● are reflected points $\mathbf{x}_P$.

For inverse distance weighted interpolation, denote by u_k the value of u at the m points chosen for interpolation, R_k the distance from the points to $\mathbf{x}_P$ and $R_{\max} = \max\{R_k\}$. The expression for $u^*(\mathbf{x}_P)$ is given by

$$u^*(\mathbf{x}_P) = \frac{\sum_k^m c_k u_k}{\sum_k^m c_k}, \tag{2.21}$$

where

$$c_k = \left(\frac{R_{\max} - R_k}{R_{\max} R_k} \right)^p. \tag{2.22}$$

To avoid numerical instability in the denominator of (2.22), if the distance R_1 from the closest point to $\mathbf{x}_P$ is less than $\min(\Delta x, \Delta y) \times 10^{-3}$, we define $c_1 = 1$ and $c_k = 0$ for the other coefficients.

The parameter p is chosen in order to distribute the influence of points used depending on their distance from the point to be interpolated. According to [17], $p = 1$ and $p = 1/2$ are good options because each point contributes more equally to the interpolation. In the next section, a study will be made on the best values of p and the number of points for interpolation for the problem described here.

One of the advantages of this interpolation method is that the coefficients are calculated only once. Although this is possible for other interpolation methods, as shown for least squares in [27], in the case of inverse distance weighted interpolation the calculation of the coefficients is straightforward from equation (2.22) without the need to solve an associated linear system.

For bilinear interpolation, the value of u^* is approximated by a polynomial of the form

$$u^*(x,y) = a_0 + a_1 x + a_2 y + a_3 xy, \tag{2.23}$$

where the coefficients a_0, a_1, a_2 and a_3 are calculated so that u is exact at the four points chosen for interpolation. In the most general cases, the coefficients are calculated by solving a linear system that can be ill-conditioned for small values of Δx and Δy. However, if the point to be approximated belongs to a rectangle whose vertices are the interpolating points, the value of u can be calculated by a simple algorithm such as the one described in [24].

3 ANALYTICAL SOLUTIONS

To evaluate the error of the immersed boundary method, we compare the numerical solution with the analytical solution for two different initial and boundary conditions where the domain Ω is a circle of radius b centered at the origin. The first initial condition is the paraboloid $w(x,y) = b^2 - x^2 - y^2$ and the second initial condition is $w(x,y) = H$, where H is a positive constant. In both cases, the boundary condition is $g(x,y) = 0$. Therefore, while the first condition is continuous throughout the domain, the second condition presents a jump discontinuity at the boundary.

Denoting by $J_n(r)$ the Bessel function of first kind and order $n \in \mathbb{N}$ and by z_n, $n \in \mathbb{N}$, the positive zeros of the Bessel function J_0, the analytical solution of (2.1)–(2.3) for the paraboloid initial condition is

$$u(t,x,y) = 4b^2 \sum_{n=1}^{+\infty} e^{-\alpha (z_n/b)^2 t} \frac{J_2(z_n)}{z_n^2 J_1^2(z_n)} J_0\left(\frac{z_n}{b}\sqrt{x^2+y^2}\right), \tag{3.1}$$

and the solution of (2.1)–(2.3) for the initial condition with boundary jump is

$$u(t,x,y) = 2H \sum_{n=1}^{+\infty} e^{-\alpha (z_n/b)^2 t} \frac{1}{z_n J_1(z_n)} J_0\left(\frac{z_n}{b}\sqrt{x^2+y^2}\right). \tag{3.2}$$

For more details about the Bessel functions and the derivation of the analytical solutions above see [11].

4 NUMERICAL EXPERIMENTS

As mentioned previously, this section presents and analyzes the results of the numerical tests for the three versions of the ghost cell method considered in this work: symmetric extrapolation with inverse distance weighted interpolation; non-symmetric extrapolation with inverse distance weighted interpolation and non-symmetric extrapolation with bilinear interpolation. For the numerical method we set $\Delta x = \Delta y$ and a time step $\Delta t = 0.9\Delta x^2/\alpha$ which satisfies the stability condition (2.14). In this work, we chose $\alpha = 0.09$, which produced a suitable Δt (neither too small nor too large) for graphically depicting the results of intermediate stages between the initial condition and the asymptotic state $u(\infty,x,y) = 0$. Other values in the range 0.01–0.5 were tested and did not change any of our results. The domain Ω is the unit circle centered at the origin, that is $b = 1$ for the analytical solutions (3.1) and (3.2).

However, first we look at the choice of the parameters p and m of the inverse distance weighted interpolation in the two versions of the ghost cell method that use it.

4.1 Calibration of parameters in the inverse distance weighted interpolation

To calibrate the parameters p and m of the interpolation method we compute a mean radial error in circles centered at the origin. For each $r > 0$ we define the mean error as

$$\epsilon\,(t,r) = \frac{1}{2\pi}\int_0^{2\pi} \frac{|\tilde{u}(t,r,\theta) - u(t,r,\theta)|}{u_{max}}\,d\theta, \tag{4.1}$$

where $\tilde{u}(t,r,\theta)$ is the value estimated by the inverse distance weighted interpolation for $u(t,r,\theta)$ and u_{max} is the maximum value of the analytical solution. Since the interpolation is done close to the boundary, r should be chosen close to 1. To take into account the two possibilities of extrapolation, it was decided to calculate the average interpolation error for the initial condition $w(x,y) = 1 - x^2 - y^2$ for $r_1 = 1 - \Delta x/2$ and $r_2 = 1 - \sqrt{2}\Delta x$ considering $\Delta x = \Delta y = 0.01$.

Figure 3 shows the average interpolation errors for r_1 considering $m-1$ grid points in Ω and one point at the boundary. In [17] it is mentioned that 3 or 4 points are sufficient for good accuracy. However, in this test the smallest errors are obtained for 6, 7 and 8 points in the interpolation. Figure 4 shows the average interpolation errors for r_2 considering only grid points in Ω. We can see that in this case the errors are smaller than those presented in Figure 3, which indicates that moving the reflected point $\mathbf{x}_P$ away from the boundary can reduce the error of the ghost cell method. This hypothesis will be corroborated in the next section. As in the previous case, using 3 or 4 points for interpolation does not seem to be the most appropriate choice since using more points reduces the error. For the tests performed below, we choose the value of p that minimizes the error ϵ for each value of m.

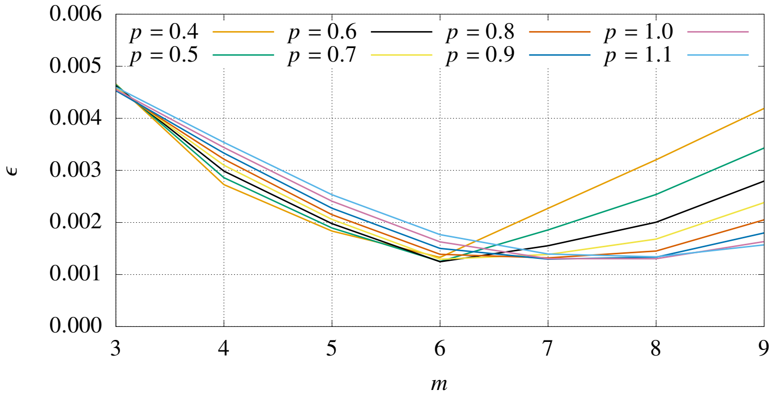


Figure 3: Mean error for inverse distance weighted interpolation in $r_1 = 0.995$ for different values of p and m.

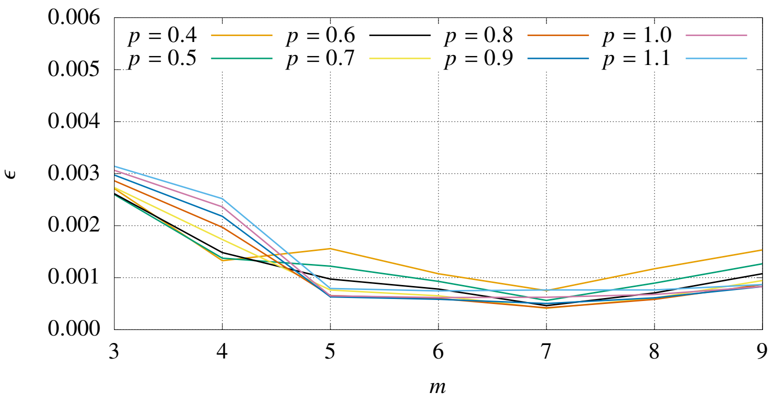


Figure 4: Mean error for inverse distance weighted interpolation in $r_2 = 0.986$ for different values of p and m.

4.2 Error analysis

For the convergence order analysis, denote by $E_1(\Delta x, t)$ the error in the L_1 norm of the numerical solution in the domain Ω at time t, and analogously denote by $E_\infty(\Delta x, t)$ the error in the L_∞ norm according to [15]. The standard procedure for estimating the order of convergence as presented in [25] is to calculate

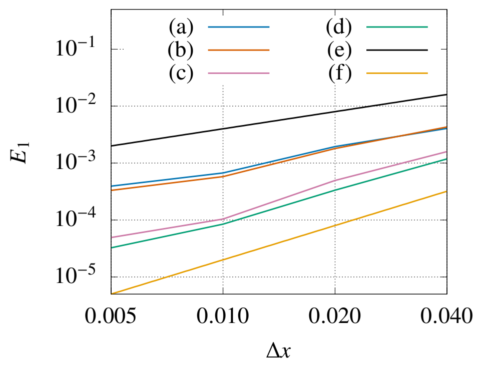
$$q(\Delta x) = \log_2 \left(\frac{E(\Delta x, 1)}{E(\Delta x/2, 1)} \right), \tag{4.2}$$

for successive refinements of the mesh $(\Delta x, \Delta x/2, \Delta x/4, \ldots)$, and then consider the order of convergence q as the average of the values obtained by (4.2). In this work we consider $\Delta x = 0.04$, $\Delta x = 0.02$, $\Delta x = 0.01$ and $\Delta x = 0.005$ in (4.2) to estimate the rate of convergence in both norms.

Case 1: Parabola

Let us consider the initial condition $w(x,y) = 1 - x^2 - y^2$ for the method. Figure 5 shows the errors E_1 (left) and E_∞ (right) at $t = 1$ for different values of Δx for the symmetric extrapolation with inverse distance weighted interpolation and Table 1 shows the estimated order of convergence; the lines with slopes of 1 and 2 are also shown for reference. We see that $m = 7$ and $m = 8$ provide the smallest errors with similar results and the largest values of q_1 near to 1.7. However $m = 8$ with $p = 1.0$ provides the smallest value of q_∞. When using 5 or 6 points the errors are larger and the order of convergence is far from 2 in both norms. Actually, q_∞ is less than 1 in the four parameter options tested for symmetric extrapolation with inverse distance weighted interpolation. Therefore, the order of convergence of the spatial discretization is not preserved.

Figure 6 shows the errors E_1 (left) and E_∞ (right) at $t = 1$ for different values of Δx for non-symmetric extrapolation with inverse distance weighted interpolation and the lines with slopes of 1 and 2 for reference. We see that the errors shown in Figure 6 are smaller than those shown in Figure 5. Therefore, as expected from the results of the previous subsection, moving the reflected



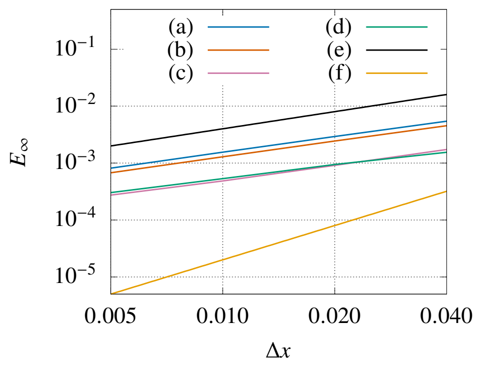


Figure 5: Errors at $t = 1$ for initial condition $w(x,y) = 1 - x^2 - y^2$ using symmetrical extrapolation with inverse distance weighted interpolation: (a) $m = 5$, $p = 0.4$; (b) $m = 6$, $p = 0.6$; (c) $m = 7$, $p = 0.9$; (d) $m = 8$, $p = 1.0$; (e) line with slope 1; (f) line with slope 2.

Table 1: Convergence order for initial condition $w(x,y) = 1 - x^2 - y^2$ using symmetric extrapolation and inverse distance weighted interpolation.

Order	$m = 5$, $p = 0.4$	$m = 6$, $p = 0.6$	$m = 7$, $p = 0.9$	$m = 8$, $p = 1.0$
q_1	1.12	1.23	1.67	1.73
q_∞	0.91	0.92	0.89	0.78

point away from the boundary increases the accuracy of the method when using the inverse distance weighted interpolation. Furthermore, Table 2 shows a gain in the order of convergence in comparison with Table 1, where the order q_1 is closer to 2 and the order q_∞ is greater than 1 but still far from 2.

Table 2: Convergence order for initial condition $w(x,y) = 1 - x^2 - y^2$ using non-symmetric extrapolation and inverse distance weighted interpolation.

Order	$m = 5$, $p = 0.9$	$m = 6$, $p = 0.9$	$m = 7$, $p = 0.7$	$m = 8$, $p = 0.8$
q_1	1.80	1.80	1.85	1.71
q_∞	1.13	1.16	1.28	1.22

Figure 7 shows the errors E_1 (left) and E_∞ (right) for the best results of symmetric and non-symmetric extrapolation with inverse distance weighted interpolation in comparison to those obtained for non-symmetric extrapolation with bilinear interpolation and the lines with slopes of 1 and 2 for reference. We see that the errors with bilinear interpolation are smaller than those obtained with inverse distance weighted interpolation, especially in the L_∞ norm. In fact, the estimated convergence order in the L_1 norm is 1.88 and in the L_∞ norm is 1.95, that is, the convergence order of the spatial discretization is maintained in both norms.

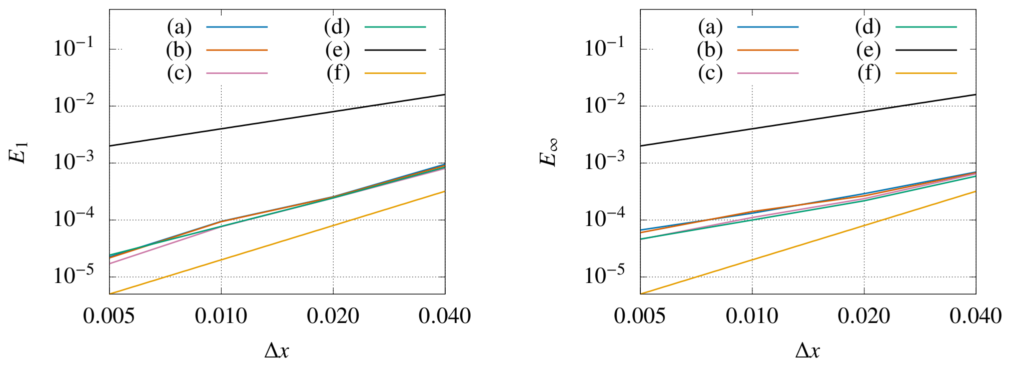


Figure 6: Errors at $t = 1$ for initial condition $w(x,y) = 1 - x^2 - y^2$ using non-symmetrical extrapolation with inverse distance weighted interpolation: (a) $m = 5$, $p = 0.9$; (b) $m = 6$, $p = 0.9$; (c) $m = 7$, $p = 0.7$; (d) $m = 8$, $p = 0.8$; (e) line with slope 1; (f) line with slope 2.

To better understand the results shown above, let us analyze the absolute error $\varepsilon(t_n, x_i, y_j) = |u_{ij}^n - u(t_n, x_i, y_j)|$ at each grid point in the domain for some instants of time. Figure 8 shows the absolute error at each grid point for $t = 0.2$ and $t = 0.4$ with $\Delta x = 0.01$, symmetric extrapolation and inverse distance weighted interpolation for $m = 8$ and $p = 1.0$. We can see that the error is not distributed uniformly over the domain, with regions (in red) near the boundary where the error is on the order of 10^{-3} and others (in blue) on the order of 10^{-7}. In fact, the blue lines near the boundary divide regions where $u_{ij}^n > u(t_n, x_i, y_j)$ from others where $u_{ij}^n < u(t_n, x_i, y_j)$. This may explain the low order of convergence in the L_∞ norm.

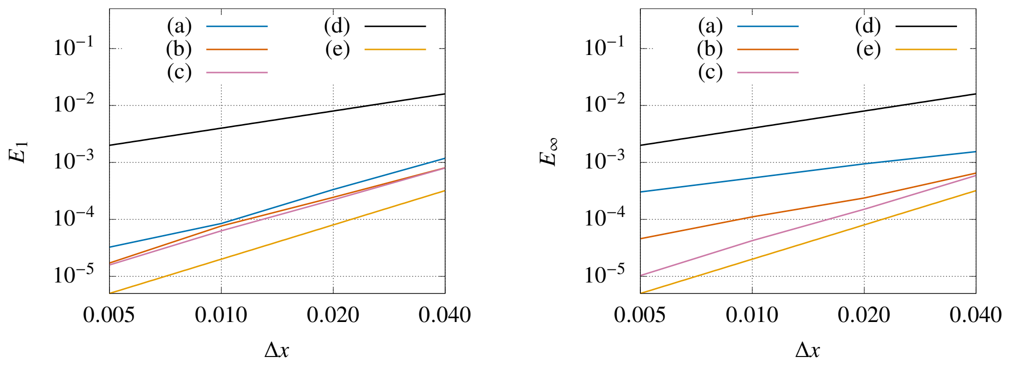
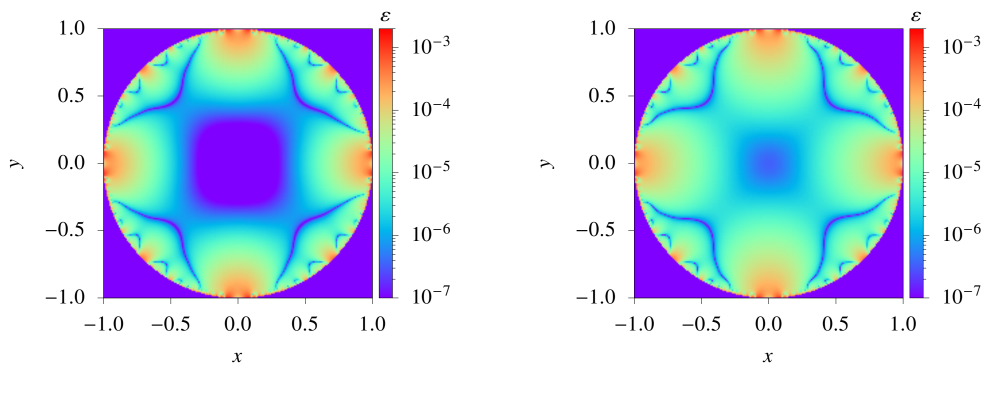
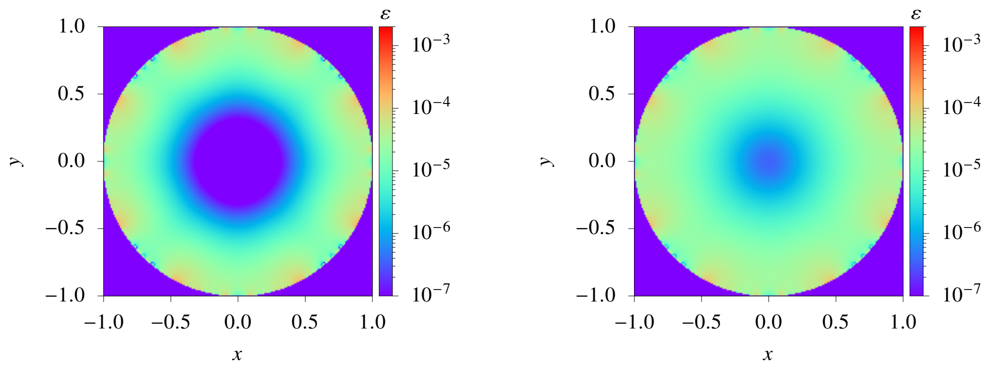


Figure 7: Errors at $t = 1$ for initial condition $w(x,y) = 1 - x^2 - y^2$ using: (a) symmetric extrapolation with $m = 8$, $p = 1.0$; (b) non-symmetric extrapolation with $m = 7$, $p = 0.7$; (c) non-symmetric extrapolation with bilinear interpolation; (d) line with slope 1; (e) line with slope 2.



(a) $t = 0.2$ (b) $t = 0.4$

Figure 8: Errors at each grid point for some times t using symmetrical extrapolation with inverse distance weighted interpolation with $m = 8$, $p = 1.0$, considering the initial condition $w(x,y) = 1 - x^2 - y^2$.



(a) $t = 0.2$ (b) $t = 0.4$

Figure 9: Errors at each grid point for some times t using non-symmetrical extrapolation with inverse distance weighted interpolation with $m = 7$, $p = 0.7$, considering the initial condition $w(x,y) = 1 - x^2 - y^2$.

On the other hand, as can be seen in Figure 9, when using non-symmetric extrapolation with inverse distance weighted interpolation the errors at each grid point are distributed more homogeneously near the boundary. In fact, in the green region close to the boundary the error varies from 10^{-5} to 10^{-4}, without the oscillations seen in the Figure 8. Furthermore, it is possible to see clearly the error propagating from the boundary to the center of the domain as the blue region decreases in size from $t = 0.2$ to $t = 0.4$, indicating an increase in error.

Similar results to those in Figure 9 occur for non-symmetric extrapolation with bilinear interpolation as can be seen in Figure 10, but with the errors at each grid point even smaller and better distributed near the boundary. Therefore, the ghost cell version with non-symmetric extrapolation and bilinear interpolation provided the best results considering the initial condition $w(x,y) = 1 - x^2 - y^2$.

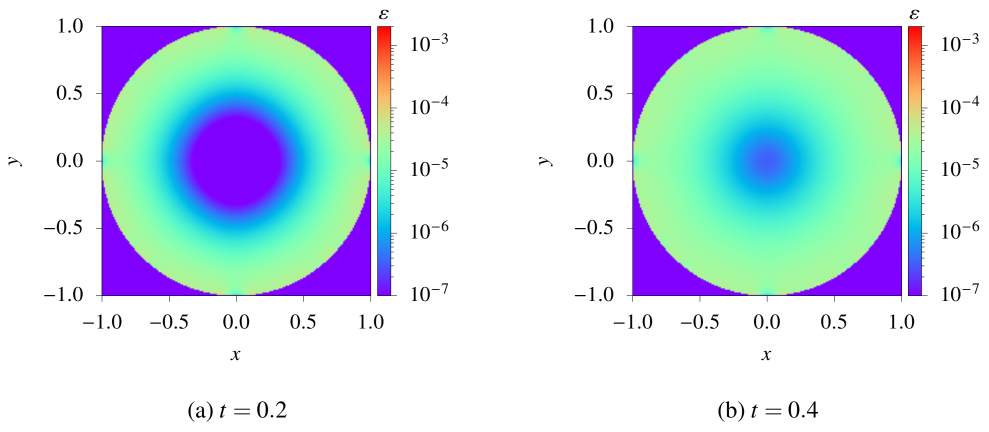


Figure 10: Errors at each grid point for some times t using non-symmetrical extrapolation with bilinear interpolation considering the initial condition $w(x,y) = 1 - x^2 - y^2$.

Case 2: Discontinuity at boundary

Let us consider the initial condition $w(x,y) = H = 0.9$ for the method. Figure 11 shows the errors E_1 (left) and E_∞ (right) at $t = 1$ for different values of Δx for the symmetric extrapolation with inverse distance weighted interpolation; the lines with slopes of 1 and 2 are also shown for reference. As in the previous case, the smallest errors occur for $m = 7$ and $m = 8$. Also, the estimated order of convergence presented in Table 3 is very similar to those in Table 1 which indicates the consistency of the method even for an initial condition with discontinuity at the boundary.

Table 3: Convergence order for initial condition $w(x,y) = H$ using symmetric extrapolation and inverse distance weighted interpolation.

Order	$m = 5,\ p = 0.4$	$m = 6,\ p = 0.6$	$m = 7,\ p = 0.9$	$m = 8,\ p = 1.0$
q_1	1.08	1.19	1.65	1.74
q_∞	0.92	0.92	0.90	0.79

Figure 12 shows the errors E_1 (left) and E_∞ (right) at $t = 1$ for different values of Δx for non-symmetric extrapolation with inverse distance weighted interpolation and the lines with slopes of 1 and 2 for reference. Also, Table 4 shows the estimated order of convergence. Again, the results

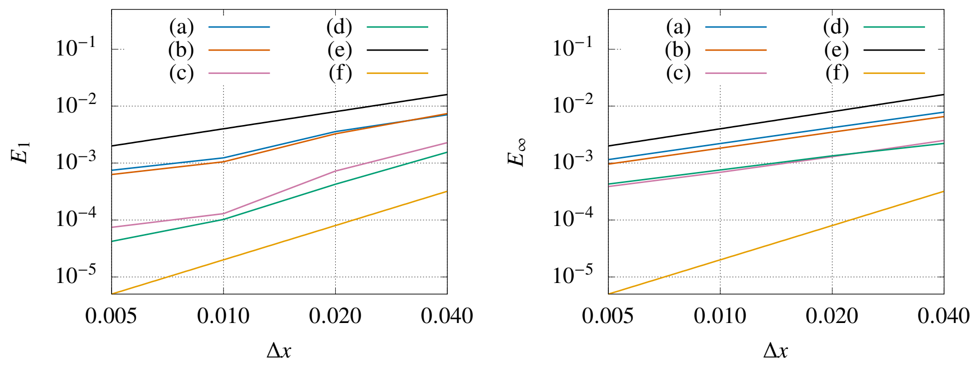


Figure 11: Errors at $t = 1$ for initial condition $w(x,y) = H$ using symmetrical extrapolation with inverse distance weighted interpolation: (a) $m = 5$, $p = 0.4$; (b) $m = 6$, $p = 0.6$; (c) $m = 7$, $p = 0.9$; (d) $m = 8$, $p = 1.0$; (e) line with slope 1; (f) line with slope 2.

obtained show that moving the reflected point away from the boundary improves the accuracy of the method, even for the initial condition with a discontinuity at the boundary.

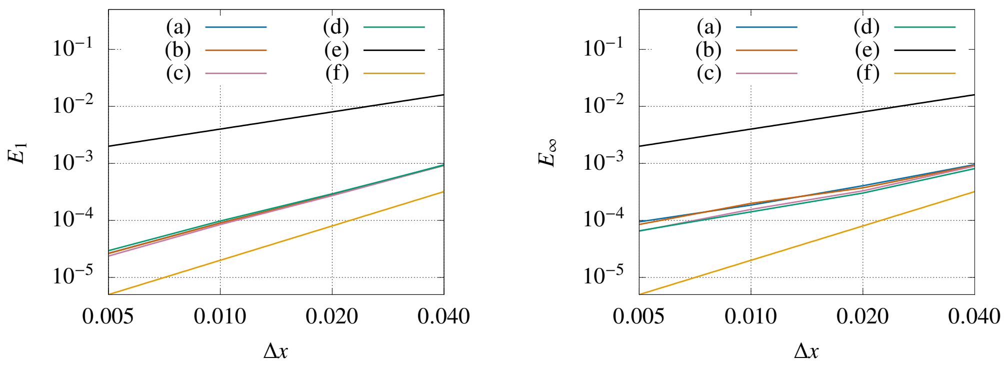


Figure 12: Errors at $t = 1$ for initial condition $w(x,y) = H$ using non-symmetrical extrapolation with inverse distance weighted interpolation: (a) $m = 5$, $p = 0.9$; (b) $m = 6$, $p = 0.9$; (c) $m = 7$, $p = 0.7$; (d) $m = 8$, $p = 0.8$; (e) line with slope 1; (f) line with slope 2.

Figure 13 shows the errors E_1 (left) and E_∞ (right) for the best results of symmetric and non-symmetric extrapolation with inverse distance weighted interpolation compared to those obtained for non-symmetric extrapolation with bilinear interpolation and the lines with slopes of 1 and 2 for reference. Again, we see that the errors with bilinear interpolation are smaller than those obtained with inverse distance weighted interpolation, especially in the L_∞ norm. The estimated convergence order in the L_1 norm is 1.83 and in the L_∞ norm is 1.9, which are only slightly smaller than the values obtained for the parabolic initial condition.

Table 4: Convergence order for initial condition $w(x,y) = H$ using non-symmetric extrapolation and inverse distance weighted interpolation.

Order	$m=5$, $p=0.9$	$m=6$, $p=0.9$	$m=7$, $p=0.7$	$m=8$, $p=0.8$
q_1	1.72	1.71	1.76	1.66
q_∞	1.11	1.15	1.26	1.21

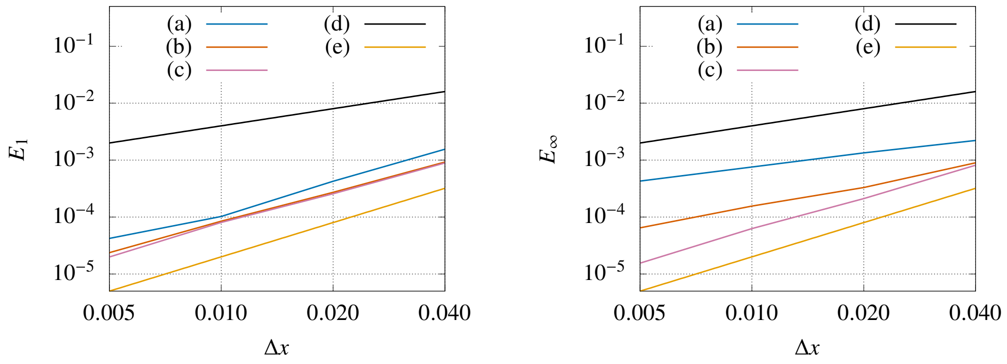
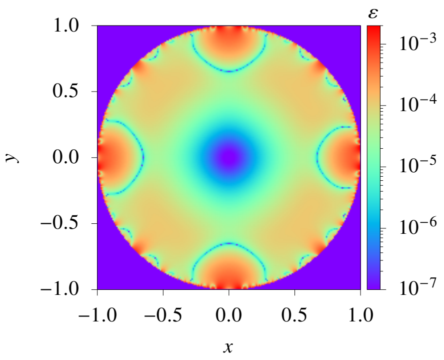


Figure 13: Errors at $t = 1$ for initial condition $w(x,y) = H$ using: (a) symmetric extrapolation with $m = 8$, $p = 1.0$; (b) non-symmetric extrapolation with $m = 7$, $p = 0.7$; (c) non-symmetric extrapolation with bilinear interpolation; (d) line with slope 1; (e) line with slope 2.

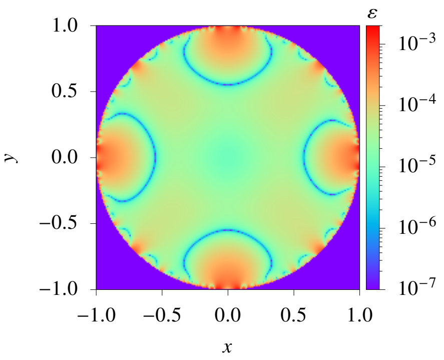
Figure 14 shows the absolute error $\varepsilon(t_n, x_i, y_j)$ at each grid point for $t = 0.2$ and $t = 0.4$ with $\Delta x = 0.01$, symmetric extrapolation and inverse distance weighted interpolation. The errors are larger than those presented in Figure 8, which is justified by the discontinuity at the boundary. Furthermore, the error is also not distributed uniformly near the boundary.

Figure 15 shows the absolute error at each grid point for $t = 0.2$ and $t = 0.4$ with $\Delta x = 0.01$, non-symmetric extrapolation and inverse distance weighted interpolation. As with symmetric extrapolation, the errors are larger than those obtained in Case 1. In the present (discontinuous) case, however, there is a blue line near the boundary where the error is small. This occurs because in the center of the domain the numerical solution satisfies $u^n_{ij} > u(t_n, x_i, y_j)$ whereas, near the boundary, $u^n_{ij} < u(t_n, x_i, y_j)$, reflecting the jump in the initial condition.

Figure 16 shows the absolute error at each grid point for $t = 0.2$ and $t = 0.4$ with $\Delta x = 0.01$, non-symmetric extrapolation and bilinear interpolation. The results are quite similar to those presented in Figure 15, but the errors are slightly smaller and more evenly distributed near the boundary. Again, the non-symmetric extrapolation with bilinear interpolation presents the best results among the options tested for the initial condition $w(x,y) = H$.

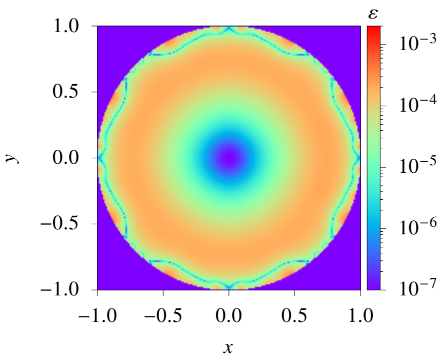


(a) $t = 0.2$

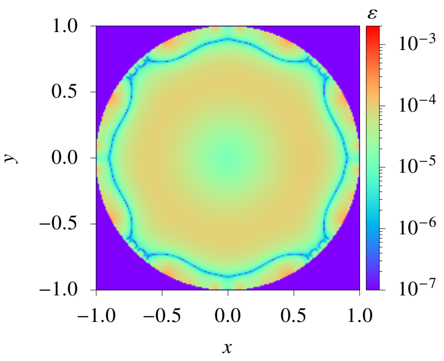


(b) $t = 0.4$

Figure 14: Errors at each grid point for symmetrical extrapolation with inverse distance weighted interpolation with $m = 8$, $p = 1.0$, considering the initial condition $w(x,y) = H$.

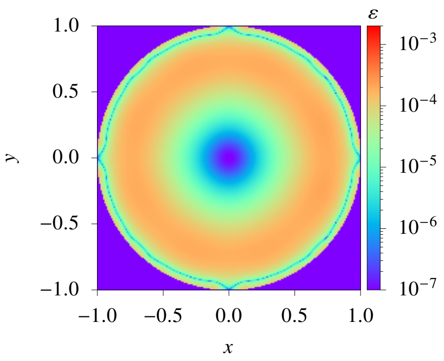


(a) $t = 0.2$

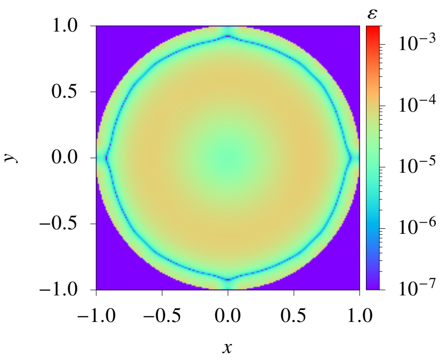


(b) $t = 0.4$

Figure 15: Errors at each grid point for non-symmetrical extrapolation with inverse distance weighted interpolation with $m = 7$, $p = 0.7$, considering the initial condition $w(x,y) = H$.



(a) $t = 0.2$



(b) $t = 0.4$

Figure 16: Errors at each grid point for non-symmetrical extrapolation with bilinear interpolation considering the initial condition $w(x,y) = H$.

5 CONCLUSIONS

Among the tested options of the ghost cell method for the heat equation, considering symmetric extrapolation with respect to the boundary and using inverse distance weighted interpolation provided the worst results with the order of convergence below 1 in the L_∞ norm. Considering a non-symmetric reflection of the ghost point with respect to the boundary so that the point to be interpolated is further away from the boundary provided an accuracy gain for the method when using inverse distance weighted interpolation; however, the order of convergence of the spatial discretization was not maintained. The best option, which was able to maintain the second order of convergence of the discretization, is to consider non-symmetric extrapolation with bilinear interpolation, since in this case the interpolation can be done with a simple algorithm that is not susceptible to instability.

Acknowledgments
The authors acknowledge the Coordination for the Improvement of Higher Education Personnel (CAPES) grant No. 88887.800036/2022-00 and 88887.671252/2022-00, the National Council for Scientific and Technological Development (CNPq) research scholarship 305903/2021-7 and the São Paulo Research Foundation (FAPESP) grant No. 2018/24284-1, for the financial support.

REFERENCES

[1] W. Chang, F. Giraldo & B. Perot. Analysis of an exact fractional step method. *Journal of Computational Physics*, **180**(1) (2002), 183–199.

[2] S. Chen, B. Merriman, S. Osher & P. Smereka. A simple level set method for solving Stefan problems. *Journal of Computational Physics*, **135**(1) (1997), 8–29.

[3] S. Chester, C. Meneveau & M.B. Parlange. Modeling turbulent flow over fractal trees with renormalized numerical simulation. *Journal of Computational Physics*, **225**(1) (2007), 427–448.

[4] C. Chi, B.J. Lee & H.G. Im. An improved ghost-cell immersed boundary method for compressible flow simulations. *International Journal for Numerical Methods in Fluids*, **83**(2) (2017), 132–148.

[5] A.J. Chorin. Numerical solution of the Navier-Stokes equations. *Mathematics of computation*, **22**(104) (1968), 745–762.

[6] A. Coco & G. Russo. Second order finite-difference ghost-point multigrid methods for elliptic problems with discontinuous coefficients on an arbitrary interface. *Journal of Computational Physics*, **361** (2018), 299–330.

[7] A. de Oliveira Fortuna. "Técnicas Computacionais para Dinâmica dos Fluídos Vol. 30". Edusp (2000).

[8] F. Gibou & R. Fedkiw. A fourth order accurate discretization for the Laplace and heat equations on arbitrary domains, with applications to the Stefan problem. *Journal of Computational Physics*, **202**(2) (2005), 577–601.

[9] F. Gibou, R.P. Fedkiw, L.T. Cheng & M. Kang. A second-order-accurate symmetric discretization of the Poisson equation on irregular domains. *Journal of Computational Physics*, **176**(1) (2002), 205–227.

[10] D. Goldstein, R. Handler & L. Sirovich. Modeling a no-slip flow boundary with an external force field. *Journal of computational physics*, **105**(2) (1993), 354–366.

[11] M.D. Greenberg. “Foundations of applied mathematics”. Courier Corporation (2013).

[12] G. Iaccarino & R. Verzicco. Immersed boundary technique for turbulent flow simulations. *Appl. Mech. Rev.*, **56**(3) (2003), 331–347.

[13] Y. Jaluria & K.E. Torrance. “Computational Heat Transfer”. Hemisphere Publishing Corporation, New York (1986).

[14] J.D. Kandilarov & L.G. Vulkov. The immersed interface method for a nonlinear chemical diffusion equation with local sites of reactions. *Numerical Algorithms*, **36** (2004), 285–307.

[15] R.J. LeVeque. “Numerical methods for conservation laws”, volume 214. Springer (1992).

[16] J.k. Liu & Z.s. Zheng. Efficient high-order immersed interface methods for heat equations with interfaces. *Applied Mathematics and Mechanics*, **35**(9) (2014), 1189–1202.

[17] K.A. Lundquist, F.K. Chow & J.K. Lundquist. An immersed boundary method enabling large-eddy simulations of flow over complex terrain in the WRF model. *Monthly Weather Review*, **140**(12) (2012), 3936–3955.

[18] S. Majumdar, G. Iaccarino, P. Durbin *et al.* RANS solvers with adaptive structured boundary non-conforming grids. *Annual Research Briefs*, **1** (2001), 179.

[19] R. Mittal & G. Iaccarino. Immersed boundary methods. *Annu. Rev. Fluid Mech.*, **37** (2005), 239–261.

[20] J. Mohd-Yusof. Combined immersed-boundary/B-spline methods for simulations of flow in complex geometries. *Annual Research Briefs. NASA Ames Research Center= Stanford University Center of Turbulence Research*, (1997), 317–327.

[21] D. Pan & T.T. Shen. Computation of incompressible flows with immersed bodies by a simple ghost cell method. *International journal for numerical methods in fluids*, **60**(12) (2009), 1378–1401.

[22] N. Peller, A.L. Duc, F. Tremblay & M. Manhart. High-order stable interpolations for immersed boundary methods. *International Journal for Numerical Methods in Fluids*, **52**(11) (2006), 1175–1193.

[23] C.S. Peskin. Flow patterns around heart valves: A numerical method. *Journal of Computational Physics*, **10**(2) (1972), 252–271. doi:https://doi.org/10.1016/0021-9991(72)90065-4. URL `https://www.sciencedirect.com/science/article/pii/0021999172900654`.

[24] W.H. Press. “Numerical recipes 3rd edition: The art of scientific computing”. Cambridge university press (2007).

[25] C.J. Roy. Review of code and solution verification procedures for computational simulation. *Journal of Computational Physics*, **205**(1) (2005), 131–156.

[26] P. Schwartz, M. Barad, P. Colella & T. Ligocki. A Cartesian grid embedded boundary method for the heat equation and Poisson's equation in three dimensions. *Journal of Computational Physics*, **211**(2) (2006), 531–550.

[27] F.S. Sousa, C.F. Lages, J.L. Ansoni, A. Castelo & A. Simao. A finite difference method with meshless interpolation for incompressible flows in non-graded tree-based grids. *Journal of Computational physics*, **396** (2019), 848–866.

[28] Y.H. Tseng & J.H. Ferziger. A ghost-cell immersed boundary method for flow in complex geometry. *Journal of computational physics*, **192**(2) (2003), 593–623.

[29] R. Verzicco. Immersed Boundary Methods: Historical Perspective and Future Outlook. *Annual Review of Fluid Mechanics*, **55** (2023).

[30] P. Young & K. Mohseni. A Second-Order Immersed Boundary Projection Method for Elliptic and Parabolic Problems. In "48th AIAA Aerospace Sciences Meeting Including the New Horizons Forum and Aerospace Exposition" (2010), p. 711.

How to cite
W.C. Lesinhovski, N.L. Dias, L.S. Freire & A.C.F.S. Jesus. On the relative merits of interpolation schemes for the immersed boundary method: a case study with the heat equation. *Trends in Computational and Applied Mathematics*, **26**(2025), e01833. doi: 10.5540/tcam.2025.026.e01833.