Penerbit Akademia Baru

Journal of Advanced Research Design

Journal homepage:
https://akademiabaru.com/submit/index.php/ard
ISSN: 2289-7984

Harmonic Path Planning using Quarter-Sweep Boosted TOR Iterative Method

Sumiati Suparmin[1], Andang Sunarto[2], Azali Saudi[1,*]

[1] Faculty of Computing and Informatics, Universiti Malaysia Sabah, 88400 Kota Kinabalu, Sabah, Malaysia
[2] Tadris Matematika, Universitas Islam Negeri Fatmawati Sukarno, Bengkulu City, Bengkulu 38211, Indonesia

ARTICLE INFO

Article history:
Received 17 January 2025
Received in revised form 10 February 2025
Accepted 2 May 2025
Available online 23 May 2025

Keywords:
Path planning; quarter-sweep; boosted iterative method; two-parameter over-relaxation (TOR); harmonic potentials

ABSTRACT

This paper presents the study to examine the effectiveness of the application of Quarter Sweep Boosted TOR with the 9-Point Laplacian operator using the families of relaxation methods in the computation of Laplace equation solutions to obtain the harmonic potentials. This work is a continuation from the past study that applied the standard application 5-Point Laplacian to solve path planning issue which a mobile robot faces because of working in indoor environment. The robot can navigate from a given initial position to a goal position by following the safest path, ensuring it avoids any obstacles and minimizes the risk of collisions. By utilizing the equation of Laplace and computing the potential values' distribution in the environments which have been simulated, the robot can determine the safest path that avoids obstacles which exists in the environment. This method ensures that the robot moves along a path where the potential for collisions is minimized. The findings confirm that QSBTOR outperforms Half Sweep Boosted TOR (HSBTOR) and Full Sweep Boosted TOR (FSBTOR). QSBTOR and HSBTOR show 73% and 50% reduction respectively, compared to FSBTOR in terms of computational complexity.

1. Introduction

Mobile robots are widely used in various industrial fields where they are exposed to hazardous conditions such as space research, nuclear industry and the mining industry. Their use is also essential for indoor applications, including offices, warehouses, pharmacies and other industrial sectors [1]. To find a safe route in a dangerous environment, mobile robots are the most suitable and safe to use [2]. One of the difficult issues with moving robots is the problem of route planning [3]. Currently, research for mobile robot path planning is increasingly becoming a hot topic among researchers [4,5].

A robot is an automated machine that can respond to the environment. To achieve such automated properties, it is necessary to use techniques from signal processing, control theory and artificial intelligence [6]. This technique is accompanied by mechanics, detectors and robot actuators. Therefore, designing a robot requires a deep understanding of its interface to the physical world. Among the key requirements for building a real automated robot is the capability in planning a route

* *Corresponding author*
E-mail address: azali@ums.edu.my

https://doi.org/10.37934/ard.132.1.114

effectively from a starting point to a specified destination point without colliding with objects or getting stuck in the path with obstacles it passes through.

Path planning is a vital component in robotics as it plays a crucial role in enabling robots to navigate from a designated starting point to a desired goal location. Especially in the ability to plan routes to allow robots to find a smooth path towards their destination. Algorithms for finding the safe path are important not only in robotics but also in network routing, video games, etc. Route planning requires a map for the purpose of allowing the robot to know its location in the environment to avoid getting stuck by any obstacles or walls while in motion.

In general, path planning strategies for navigation are divided into two categories local methods and global methods. The local methods which work in response to input sensors and global methods, which involve the creation and execution of action plans [7]. The challenge of keeping the robot in a collision-free state is solved using local planning algorithms. These strategies are referred to as local since they only assess the robot's immediate environment when determining how it should react. Only immediate sensory data is dealt with by the local technique. As a result, it operates extremely quickly, allowing it to quickly react to the environment's changes. However, this speed comes at the expense of completeness. In general, a local method follows its specific functions greedily. Therefore, it may become stuck in local minimum of the function and fail to reach its destination [8,9].

The global methods, however, solve the problem by making a full representation of the environment. Moreover, the environment model is a three-dimensional space with several obstacles of various shapes, as well as inner and outer borders. When global planners construct a plan, they consider the entire environment, which demands a substantial amount of processing power [10]. Global path planning, in general, is computationally inflexible. The cost of computing the exact solution to a path planning problem grows exponentially as the environment grows larger [11]. The real world's dynamic nature is always in motion. Thus, the availability of time for a robot to make effective planning is greatly constrained in this dynamic setting. The researchers are challenged by the complexity of the computational demands of such planning challenges. To make matters worse, data and knowledge about the environment are gathered from noisy sensors, making them inaccurate and incomplete. As a result, to deal with incomplete and perhaps erroneous representations of the world, path planning algorithms must be reliable and efficient. In this study, the exact method used in global manner, that utilize the concept Potential Field, then create the Harmonic Potential Fields by solving Laplace's Equation. They established a global method for path finding that used the harmonic potentials to construct a smooth as well as collision-free path [8]. Harmonic potentials have been obtained by solving Laplace's Eq. (1), which is defined as:

$$\nabla^2 u = \sum_{i=1}^{n} \frac{\partial^2 u}{\partial x_i^2} = 0 \tag{1}$$

where the dimension is n and the i-th Cartesian coordinate is denoted by x_i. Consequently, Harmonic potentials have been established globally throughout the whole region. Additionally, the solutions which are harmonic to Laplace's Eq. (1) are later employed to figure out the path lines from the starting point towards the goal point. Moreover, obstacles have been regarded as current sources, while the aim has been regarded as a sink. Dirichlet boundary conditions are used in this case. The path to the goal point can be discovered by executing a path search on the harmonic potentials using the gradient descent method [12].

This study applies the above paradigm to solve path-finding problems, employing the analogies of temperature as well as heat flow for the potential as well as path line, accordingly, to characterize the solutions of Laplace's equation. Numerical methods are used in solving Laplace's equation and

acquiring the harmonic potential (temperature values) for each node. The obtained temperature values are then used in the path-finding process by descending from a starting point (high temperature) to the goal point (lowest temperature).

The fundamental concept of numerical methods is to represent the problem, i.e. Laplace's Eq. (1), in the form of a linear system as in Eq. (2),

$$Ax = b \tag{2}$$

in which *A* represents a coefficient matrix, *x* is a vector which has been given, while *b* denotes the unknown vector to be determined. Although Eq. (2) can be solved using a direct method, the more efficient iterative methods have been employed to compute the solutions. This is because its application in path-finding problems often results in large linear systems along with sparse coefficient matrices [13]. Additionally, iterative methods are mathematical techniques that produce a series of improving approximations [14]. These techniques are efficient in terms of memory storage as well as computation.

Traditional iterative methods for solving linear systems have largely relied on the 5-Point Laplacian operator. While effective, this approach has certain limitations in terms of computational efficiency. Recently, the 9-Point Laplacian operator has gained attention for its ability to address these limitations, demonstrating notable success in solving various types of linear systems, as reported in prior studies [15,16]. Historically, these iterative methods employed the Full-Sweep (FS) iteration technique, which processes computational nodes on a regular fine grid. However, advancements such as the Half-Sweep (HS) method, introduced by Abdullah [17] and the Quarter-Sweep (QS) method, developed by Othman *et al.,* [18], have significantly improved execution times by reducing the number of active computational nodes using coarse grids.

Further enhancements in computation speed were achieved through the application of relaxation techniques, including Successive Over-Relaxation (SOR), Accelerated Over-Relaxation (AOR) and Two-Parameter Over-Relaxation (TOR) methods [19,20]. These approaches optimized the iterative process, offering faster convergence and reduced computational cost. Despite these advances, many existing solutions to Eq. (1) remain rooted in the 5-point iterative scheme.

Building on this foundation, recent research has highlighted the potential of the 9-Point Laplacian operator to deliver improved performance in terms of computational efficiency [21,22]. Recognizing this, the present work adopts the 9-Point Laplacian operator as a framework for developing enhanced iterative methods. These methods, referred to as the family of Boosted iterative schemes, aim to leverage the strengths of the 9-point operator to achieve superior performance in solving linear systems.

2. Methodology

To ensure the study outcome, this study is structured into four phases to comprehensively cover the following aspects:

i. Environment Setup: In the initial phase, four different environments were designed for path generation, varying in sizes: 300 by 300, 600 by 600, 900 by 900, 1200 by 1200 and 1500 by 1500. This variety of sizes allowed for a comprehensive exploration of path generation across different scales.

ii. Computing the Harmonic Potentials: This process involved the use of numerical iterative methods, including full-sweep, half-sweep and quarter-sweep iterations, applied through both regular and modified point iterative techniques. These approaches were critical for

accurately determining the Harmonic potentials. In particular, the 9-point finite difference schemes were employed to ensure computational precision and accuracy. During this phase, the proposed QSBTOR method, which is based on Quarter-Sweep iteration, was introduced and implemented.

iii. GDS Algorithm: In this phase, the computed Harmonic Potentials were fed into the Gradient Descent Search (GDS) algorithm. The GDS algorithm, starting from a designated point, strategically navigated through the environment, identifying the lowest point among its neighbouring points. Moreover, this process continued iteratively until the algorithm reached the lowest function values, which were recognized as the goal points. The GDS algorithm played a pivotal role in optimizing the path-finding process, ensuring efficiency.

iv. Path Generation: The final phase focused on generating paths using the optimized algorithms and displaying them on the simulation platform. Additionally, the simulator provided detailed information about the generated paths, including the time taken and the number of iterations required.

2.1 Formulation of Iteration Methods with 9-Point Laplacian

The approximation of the 2D Laplacian Eq. (1) based on the 9-point Laplacian is given as in Eq. (3),

$$\nabla^2 f(x,y) = \frac{1}{6h^2}\big((4u(x-h,y) + 4u\,(x+h,y) + 4u\,(x,y-h) + 4u\,(x,y+h) + u\,(x-h,y-h) + u\,(x+h,y-h) + u(x-h,y+h) + u(x+h,y+h) - 20u(x,y)\big) \quad (3)$$

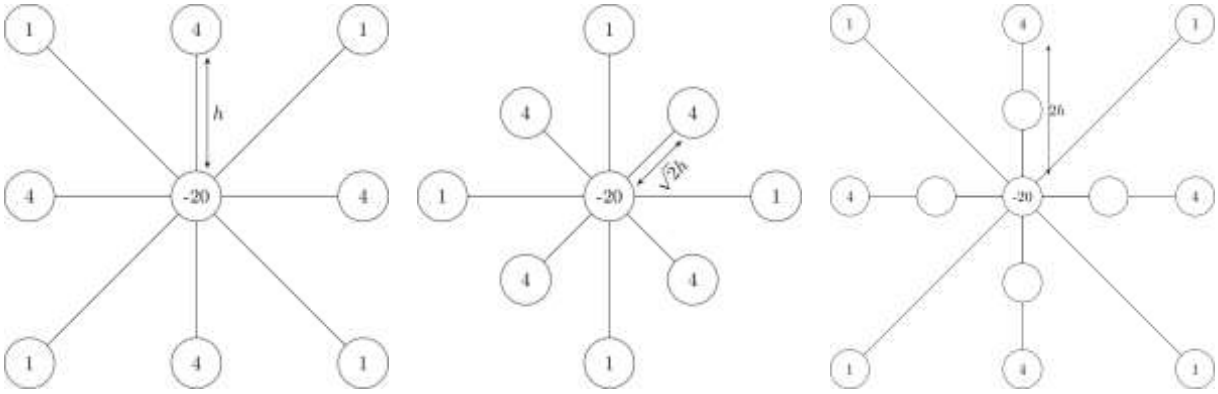
By rotating the x-y axis clockwise 45°, the rotated 9-point Laplacian approximation can be written as Eq. (4),

$$\nabla^2 f(x,y) = \frac{1}{12h^2}\big((4u(x-h,y-h) + 4u\,(x+h,y-h) + 4u\,(x-h,y+h) + 4u\,(x+h,y+h) + u\,(x-2h,y) + u\,(x+2h,y) + u(x,y-2h) + u(x,y+2h) - 20u(x,y)\big) \quad (4)$$

Moreover, by considering he points at grids size *2h*, the 9-point approximation can be expressed as Eq. (5):

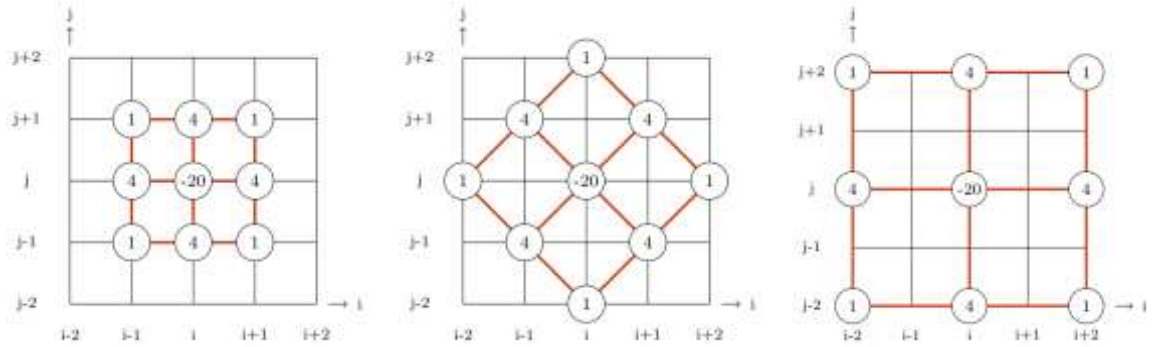
$$\nabla^2 f(x,y) = \frac{1}{24h^2}\big((4u(x-2h,y) + 4u\,(x+2h,y) + 4u\,(x,y-2h) + 4u\,(x,y+2h) + u\,(x-2h,y-2h) + u\,(x+2h,y-2h) + u(x-2h,y+2h) + u(x+2h,y+2h) - 20u(x,y)\big) \quad (5)$$

For the respective full-, half- and quarter-sweep Boosted, Figure 1 and 2 illustrate the computational molecules and portion of the computational grids.



(a) (b) (c)

Fig. 1. The 9-point Laplacian approximation computational molecules for the FS, HS and QS Boosted operations, respectively



(a) (b) (c)

Fig. 2. The 9-point Laplacian computational grids at (*i, j*) for (a) FS (b) HS (c) QS Boosted cases, respectively

By applying $u_{i,j}$ to approximate *f(x, y)* and applying the 9-point approximations Eq. (3), (4) and (5) for FS, HS and QS, the approximation equations for Eq. (1) may be expressed in a different form as Eq. (6) to (8),

$$4\left(u_{i-1,j} + u_{i+1,j} + u_{i,j-1} + u_{i,j+1}\right) + u_{i-1,j-1} + u_{i+1,j-1} + u_{i-1,j+1} + u_{i+1,j+1} - 20u_{i,j} = 0, \quad (6)$$

$$4\left(u_{i-1,j-1} + u_{i+1,j-1} + u_{i-1,j+1} + u_{i+1,j+1}\right) + u_{i-2,j} + u_{i+2,j} + u_{i,j-2} + u_{i,j+2} - 20u_{i,j} = 0, \quad (7)$$

$$4\left(u_{i-2,j} + u_{i+2,j} + u_{i,j+2} + u_{i,j+2}\right) + u_{i-2,j-2} + u_{i+2,j-2} + u_{i-2,j+2} + u_{i+2,j+2} - 20u_{i,j} = 0. \quad (8)$$

According to the finite difference Eq. (6), (7) and (8), the iterative strategies for the 9-point FS, HS and QS instances are defined as follows in Eq. (9) to (11),

$$u_{i,j}^{(k+1)} = \frac{1}{5}\left(u_{i-1,j}^{(k+1)} + u_{i+1,j}^{(k)} + u_{i,j-1}^{(k+1)} + u_{i,j+1}^{(k)}\right) + \frac{1}{20}\left(u_{i-1,j-1}^{(k+1)} + u_{i+1,j-1}^{(k+1)} + u_{i-1,j+1}^{(k)} + u_{i+1,j+1}^{(k)}\right), \quad (9)$$

$$u_{i,j}^{(k+1)} = \frac{1}{5}\left(u_{i-1,j-1}^{(k+1)} + u_{i+1,j-1}^{(k+1)} + u_{i-1,j+1}^{(k)} + u_{i+1,j+1}^{(k)}\right) + \frac{1}{20}\left(u_{i-2,j}^{(k+1)} + u_{i,+2,j}^{(k)} + u_{i,j-2}^{(k+1)} + u_{i,j+2}^{(k)}\right) \quad (10)$$

$$u_{i,j}^{(k+1)} = \frac{1}{5}\left(u_{i-2,j}^{(k+1)} + u_{i+2,j}^{(k)} + u_{i,j-2}^{(k+1)} + u_{i,j+2}^{(k)}\right) + \frac{1}{20}\left(u_{i-2,j-2}^{(k+1)} + u_{i+2,j-2}^{(k+1)} + u_{i-2,j+2}^{(k)} + u_{i+2,j+2}^{(k)}\right). \quad (11)$$

2.2 Boosted SOR Methods with the 9-Point Laplacian

By using the weighted parameter ω and in accordance with Eq. (9), (10) and (11), the corresponding 9-point SOR iterative schemes for the Full-Sweep Boosted SOR (FSBSOR) [23], Half-Sweep Boosted SOR (HSBSOR) [24] and Quarter-Sweep Boosted SOR (QSBSOR) are formulated as follows in Eq. (12) to (14),

$$u_{i,j}^{(k+1)} = \frac{\omega}{5}\left(u_{i-1,j}^{(k+1)} + u_{i+1,j}^{(k)} + u_{i,j-1}^{(k+1)} + u_{i,j+1}^{(k)}\right) + \frac{\omega}{20}\left(u_{i-1,j-1}^{(k+1)} + u_{i+1,j-1}^{(k+1)} + u_{i-1,j+1}^{(k)} + u_{i+1,j+1}^{(k)}\right) + (1-\omega)u_{i,j}^{(k)}, \quad (12)$$

$$u_{i,j}^{(k+1)} = \frac{\omega}{5}\left(u_{i-1,j-1}^{(k+1)} + u_{i+1,j-1}^{(k+1)} + u_{i-1,j+1}^{(k)} + u_{i+1,j+1}^{(k)}\right) + \frac{\omega}{20}\left(u_{i-2,j}^{(k+1)} + u_{i+2,j}^{(k)} + u_{i,j-2}^{(k+1)} + u_{i,j+2}^{(k)}\right) + (1-\omega)u_{i,j}^{(k)}, \quad (13)$$

$$u_{i,j}^{(k+1)} = \frac{\omega}{5}\left(u_{i-2,j}^{(k+1)} + u_{i+2,j}^{(k)} + u_{i,j-2}^{(k+1)} + u_{i,j+2}^{(k)}\right) + \frac{\omega}{20}\left(u_{i-2,j-2}^{(k+1)} + u_{i+2,j-2}^{(k+1)} + u_{i-2,j+2}^{(k)} + u_{i+2,j+2}^{(k)}\right) + (1-\omega)u_{i,j}^{(k)}. \quad (14)$$

2.3 Boosted AOR Methods with the 9-Point Laplacian

Over the years, the development of the AOR family's fast iterative schemes has focused on exploring the application of this method to such schemes. The AOR approach serves as a two-parameter generalization of the Successive Over-Relaxation (SOR) method. By fully leveraging these two adjustable parameters, it is possible to design iterative methods that are more flexible, widely applicable and achieve a faster rate of convergence compared to similar methods. The derivation of the Full-Sweep Accelerated Over-Relaxation (FSBAOR) scheme for this approximation is provided by Ling *et al.,* [25] and the formulation of FSBAOR is expressed as follows in Eq. (15),

$$u_{i,j}^{(k+1)} = \frac{\omega}{5}\left(u_{i-1,j}^{(k)} + u_{i+1,j}^{(k)} + u_{i,j-1}^{(k)} + u_{i,j+1}^{(k)}\right) + \frac{r}{5}\left(u_{i-1,j}^{(k+1)} - u_{i-1,j}^{(k)} + u_{i,j-1}^{(k+1)} - u_{i,j-1}^{(k)}\right) + \frac{\omega}{20}\left(u_{i-1,j-1}^{(k)} + u_{i+1,j-1}^{(k)} + u_{i-1,j+1}^{(k)} + u_{i+1,j+1}^{(k)}\right) + \frac{r}{20}\left(u_{i-1,j-1}^{(k+1)} - u_{i-1,j-1}^{(k)} + u_{i+1,j-1}^{(k+1)} - u_{i+1,j-1}^{(k)}\right) + (1-\omega)u_{i,j}^{(k)}. \quad (15)$$

By rotating the computational mesh by 45 degrees, the rotated 9-point AOR iterative scheme is derived. In this scheme, only half of the total mesh points are considered. Consequently, the Half-Sweep Boosted AOR (HSBAOR) method can be expressed as follows in Eq. (16),

$$u_{i,j}^{(k+1)} = \frac{\omega}{5}\left(u_{i-1,j-1}^{(k)} + u_{i+1,j-1}^{(k)} + u_{i-1,j+1}^{(k)} + u_{i+1,j+1}^{(k)}\right) + \frac{r}{5}\left(u_{i-1,j-1}^{(k+1)} - u_{i-1,j-1}^{(k)} + u_{i+1,j-1}^{(k+1)} - u_{i+1,j-1}^{(k)}\right) + \frac{\omega}{20}\left(u_{i-2,j}^{(k)} + u_{i+2,j}^{(k)} + u_{i,j-2}^{(k)} + u_{i,j+2}^{(k)}\right) + \frac{r}{20}\left(u_{i-2,j}^{(k+1)} - u_{i-2,j}^{(k)} + u_{i,j-2}^{(k+1)} - u_{i,j-2}^{(k)}\right) + (1-\omega)u_{i,j}^{(k)}. \quad (16)$$

When the quarter-sweep iteration is applied to the 9-point AOR iterative scheme, only a quarter of the total nodes in the mesh points are involved in the computation. Consequently, the proposed Quarter-Sweep Boosted AOR (QSBAOR) cases are formulated as follows in Eq. (17),

$$u_{i,j}^{(k+1)} = \frac{\omega}{5}\left(u_{i-2,j}^{(k)} + u_{i+2,j}^{(k)} + u_{i,j-2}^{(k)} + u_{i,j+2}^{(k)}\right) + \frac{r}{5}\left(u_{i-2,j}^{(k+1)} - u_{i-2,j}^{(k)} + u_{i,j-2}^{(k+1)} - u_{i,j-2}^{(k)}\right) + \frac{\omega}{20}\left(u_{i-2,j-2}^{(k)} + u_{i+2,j-2}^{(k)} + u_{i-2,j+2}^{(k)} + u_{i+2,j+2}^{(k)}\right) + \frac{r}{20}\left(u_{i-2,j-2}^{(k+1)} - u_{i-2,j-2}^{(k)} + u_{i+2,j-2}^{(k+1)} - u_{i+2,j-2}^{(k)}\right) + (1-\omega)u_{i,j}^{(k)}. \quad (17)$$

2.4 Boosted TOR Methods with the 9-Point Laplacian

To obtain the Full-Sweep Boosted TOR using an improved iteration with the new parameter, *s*, modifications are made to the Full-Sweep Boosted AOR iterative scheme. Specifically, the terms $\frac{r}{5}\left(u_{i,j-1}^{(k+1)} - u_{i,j-1}^{(k)}\right)$ and $\frac{r}{20}\left(u_{i,j-1}^{(k+1)} - u_{i,j-1}^{(k)}\right)$ are replaced with $\frac{s}{5}\left(u_{i,j-1}^{(k+1)} - u_{i,j-1}^{(k)}\right)$ and $\frac{s}{20}\left(u_{i,j-1}^{(k+1)} - u_{i,j-1}^{(k)}\right)$, respectively. Therefore, the Full-Sweep Boosted TOR can be expressed as follows in Eq. (18),

$$u_{i,j}^{(k+1)} = \frac{\omega}{5}\left(u_{i-1,j}^{(k)} + u_{i+1,j}^{(k)} + u_{i,j-1}^{(k)} + u_{i,j+1}^{(k)}\right) + \frac{r}{5}\left(u_{i-1,j}^{(k+1)} - u_{i-1,j}^{(k)}\right) + \frac{s}{5}\left(u_{i,j-1}^{(k+1)} - u_{i,j-1}^{(k)}\right) + \frac{\omega}{20}\left(u_{i-1,j-1}^{(k)} + u_{i+1,j-1}^{(k)} + u_{i-1,j+1}^{(k)} + u_{i+1,j+1}^{(k)}\right) + \frac{r}{20}\left(u_{i-1,j-1}^{(k+1)} - u_{i-1,j-1}^{(k)}\right) + \frac{s}{20}\left(u_{i+1,j-1}^{(k+1)} - u_{i+1,j-1}^{(k)}\right) + (1-\omega)u_{i,j}^{(k)}. \quad (18)$$

A similar procedure is applied by introducing the second acceleration parameter, *s* to the rotated 9-point iterative scheme, where only half of the total mesh points are considered. Consequently, the Half-Sweep Boosted TOR (HSBTOR) can be expressed as follows in Eq. (19),

$$u_{i,j}^{(k+1)} = \frac{\omega}{5}\left(u_{i-1,j-1}^{(k)} + u_{i+1,j-1}^{(k)} + u_{i-1,j+1}^{(k)} + u_{i+1,j+1}^{(k)}\right) + \frac{r}{5}\left(u_{i-1,j-1}^{(k+1)} - u_{i-1,j-1}^{(k)} + u_{i+1,j-1}^{(k+1)} - u_{i+1,j-1}^{(k)}\right) + \frac{\omega}{20}\left(u_{i-2,j}^{(k)} + u_{i+2,j}^{(k)} + u_{i,j-2}^{(k)} + u_{i,j+2}^{(k)}\right) + \frac{r}{20}\left(u_{i-2,j}^{(k+1)} - u_{i-2,j}^{(k)} + u_{i,j-2}^{(k+1)} - u_{i,j-2}^{(k)}\right) + (1-\omega)u_{i,j}^{(k)}. \quad (19)$$

Finally, the Quarter-Sweep Boosted TOR (QSBTOR) method is derived by applying wider spacing between nodes, involving only a quarter of the total nodes in the mesh points. As illustrated in Figure 3, only the black nodes are utilized in the computation. With the introduction of the new acceleration parameter, *s*, the QSBTOR method demonstrates a faster rate of convergence, greater flexibility and broader applicability compared to similar methods. The proposed QSBTOR formulation is given as follows in Eq. (20):

$$u_{i,j}^{(k+1)} = \frac{\omega}{5}\left(u_{i-2,j}^{(k)} + u_{i+2,j}^{(k)} + u_{i,j-2}^{(k)} + u_{i,j+2}^{(k)}\right) + \frac{r}{5}\left(u_{i-2,j}^{(k+1)} - u_{i-2,j}^{(k)}\right) + \frac{s}{5}\left(u_{i,j-2}^{(k+1)} - u_{i,j-2}^{(k)}\right) + \frac{\omega}{20}\left(u_{i-2,j-2}^{(k)} + u_{i+2,j-2}^{(k)} + u_{i-2,j+2}^{(k)} + u_{i+2,j+2}^{(k)}\right) + \frac{r}{20}\left(u_{i-2,j-2}^{(k+1)} - u_{i-2,j-2}^{(k)}\right) + \frac{s}{20}\left(u_{i+2,j-2}^{(k+1)} - u_{i+2,j-2}^{(k)}\right) + (1-\omega)u_{i,j}^{(k)}. \quad (20)$$

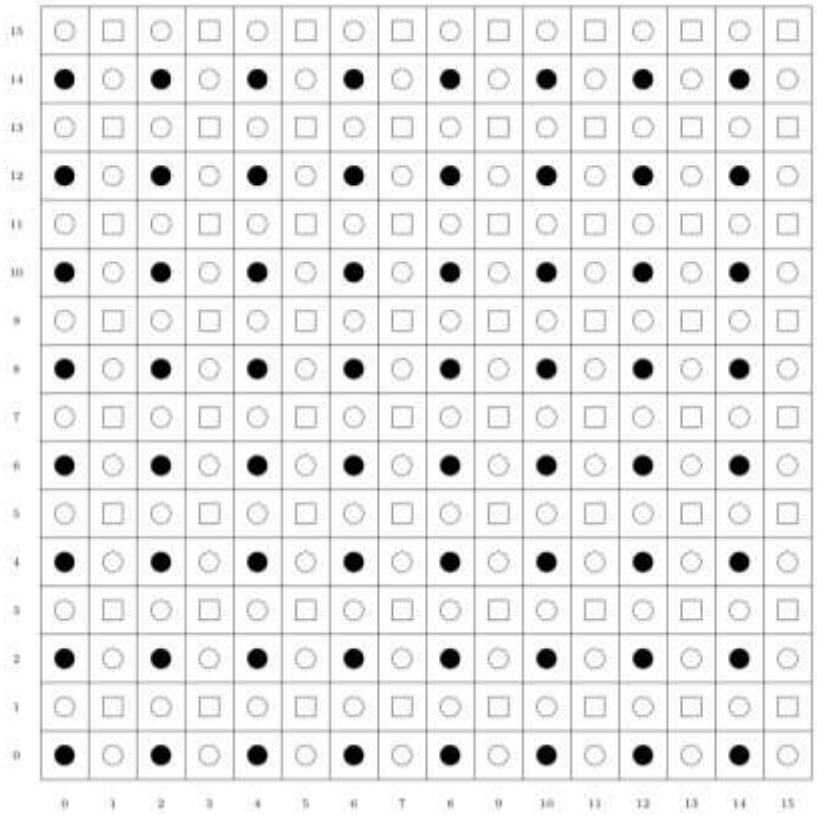


Fig. 3. The computational mesh for quarter-sweep iteration, where only black points are involved in the computation

Algorithm 1. QSBTOR

1. Set configuration space (obstacles, destination)
2. Set value of ω, r, s
3. Divide the mesh points into three types: black, white circle and white square points
4. Compute black points not including obstacles using Eq. (20)
5. $k \leftarrow 0$
6. repeat

$$u_{i,j}^{(k+1)} = \frac{\omega}{5}\left(u_{i-2,j}^{(k)} + u_{i+2,j}^{(k)} + u_{i,j-2}^{(k)} + u_{i,j+2}^{(k)}\right) + \frac{r}{5}\left(u_{i-2,j}^{(k+1)} - u_{i-2,j}^{(k)}\right) + \frac{s}{5}\left(u_{i,j-2}^{(k+1)} - u_{i,j-2}^{(k)}\right) + \frac{\omega}{20}\left(u_{i-2,j-2}^{(k)} + u_{i+2,j-2}^{(k)} + u_{i-2,j+2}^{(k)} + u_{i+2,j+2}^{(k)}\right) + \frac{r}{20}\left(u_{i-2,j-2}^{(k+1)} - u_{i-2,j-2}^{(k)}\right) + \frac{s}{20}\left(u_{i+2,j-2}^{(k+1)} - u_{i+2,j-2}^{(k)}\right) + (1-\omega)u_{i,j}^{(k)} \quad (20)$$

7. $k \leftarrow k + 1$
8. until ε is less than the convergence criterion
9. Compute all white square points (rotated) not including obstacles using Eq. (19)

$$u_{i,j}^{(k+1)} = \frac{\omega}{5}\left(u_{i-1,j-1}^{(k)} + u_{i+1,j-1}^{(k)} + u_{i-1,j+1}^{(k)} + u_{i+1,j+1}^{(k)}\right) + \frac{r}{5}\left(u_{i-1,j-1}^{(k+1)} - u_{i-1,j-1}^{(k)} + u_{i+1,j-1}^{(k+1)} - u_{i+1,j-1}^{(k)}\right) + \frac{\omega}{20}\left(u_{i-2,j}^{(k)} + u_{i+2,j}^{(k)} + u_{i,j-2}^{(k)} + u_{i,j+2}^{(k)}\right) + \frac{r}{20}\left(u_{i-2,j}^{(k+1)} - u_{i-2,j}^{(k)} + u_{i,j-2}^{(k+1)} - u_{i,j-2}^{(k)}\right) + (1-\omega)u_{i,j}^{(k)} \quad (19)$$

11. Compute all white circle points (standard) not including obstacles using Eq. (18),

$$12.\ u_{i,j}^{(k+1)} = \frac{\omega}{5}\left(u_{i-1,j}^{(k)} + u_{i+1,j}^{(k)} + u_{i,j-1}^{(k)} + u_{i,j+1}^{(k)}\right) + \frac{r}{5}\left(u_{i-1,j}^{(k+1)} - u_{i-1,j}^{(k)}\right) + \frac{s}{5}\left(u_{i,j-1}^{(k+1)} - u_{i,j-1}^{(k)}\right) + \frac{\omega}{20}\left(u_{i-1,j-1}^{(k)} + u_{i+1,j-1}^{(k)} + u_{i-1,j+1}^{(k)} + u_{i+1,j+1}^{(k)}\right) + \frac{r}{20}\left(u_{i-1,j-1}^{(k+1)} - u_{i-1,j-1}^{(k)}\right) + \frac{s}{20}\left(u_{i+1,j-1}^{(k+1)} - u_{i+1,j-1}^{(k)}\right) + (1-\omega)u_{i,j}^{(k)} \quad (18)$$

3. Results and Discussion

The simulation was conducted using the Robot 2D Simulator on a Linux system equipped with an Intel i5 processor running at 2.5 GHz and 8 GB of memory. The execution of the QSBTOR implementation, based on Eq. (18), (19) and (20), to solve the 2D Laplace's problem described in Eq. (1), is outlined in Algorithm 1. In the simulation, the start and goal points are represented by red and green points, respectively. Two different maps were used in the simulations, with five different sizes tested. The experimental results for the Full-Sweep (FS), Half-Sweep (HS) and Quarter-Sweep (QS) Boosted methods, based on the 9-Point Laplacian, are presented in this section. The iteration counts and CPU times for each algorithm are recorded in Table 1 and 2, respectively.

Table 1
Performance of the methods in terms of the number of iterations

		N				
Map	Method	300	600	900	1200	1500
Map 1	FSBSOR	2874	10450	23786	35469	62307
	FSBAOR	2317	8428	19327	28647	50524
	FSBTOR	2177	7981	18117	26916	47466
	HSBSOR	1480	5381	12253	18229	32093
	HSBAOR	1179	4343	9899	14706	25994
	HSBTOR	1107	4081	9300	13808	24417
	QSBSOR	757	2776	6375	9357	16575
	QSBAOR	187	2219	5145	8570	13379
	QSBTOR	177	2091	4840	7077	12582
Map 2	FSBSOR	2094	7956	19732	32360	45628
	FSBAOR	1721	6374	16021	26242	37267
	FSBTOR	1634	5991	15188	24602	36683
	HSBSOR	1124	4175	9460	16674	23759
	HSBAOR	905	3353	7632	13502	19335
	HSBTOR	850	3149	7171	12649	18182
	QSBSOR	594	2082	4889	8558	12185
	QSBAOR	461	1591	3920	6859	10130
	QSBTOR	426	1448	3690	6470	9555

Table 2
Performance of the methods in terms of the time of execution (in seconds)

		N				
Map	Method	300	600	900	1200	1500
Map 1	FSBSOR	1.602	23.318	123.358	326.31	932.092
	FSBAOR	1.497	22.092	120.699	307.958	893.052
	FSBTOR	1.43	21.041	109.89	290.711	805.575
	HSBSOR	0.533	8.251	44.013	116.616	328.061
	HSBAOR	0.493	7.42	39.259	103.669	291.926
	HSBTOR	0.479	6.235	33.801	103.172	263.516
	QSBSOR	0.219	3.329	17.712	46.912	137.155
	QSBAOR	0.187	2.908	16.589	45.593	120.228
	QSBTOR	0.177	2.749	14.705	38.442	110.641
Map 2	FSBSOR	1.422	22.837	150.992	345.879	756.108
	FSBAOR	1.361	20.787	153.955	322.32	713.486
	FSBTOR	1.268	19.366	82.025	300.589	747.515
	HSBSOR	0.516	8.019	37.637	119.921	266.421
	HSBAOR	0.437	6.585	33.287	105.91	235.898
	HSBTOR	0.385	5.906	27.725	88.206	195.977
	QSBSOR	0.223	3.117	14.704	46.39	103.631
	QSBAOR	0.199	2.578	12.829	40.69	93.254
	QSBTOR	0.177	2.324	12.051	38.193	87.587

Figure 4 present graphs illustrating the number of iterations and execution time, respectively, corresponding to the results summarized in Table 1 and 2. Both figures demonstrate that execution time increases with the number of iterations. A closer examination of the graphs reveals that the QSBTOR method outperforms the other proposed approaches in terms of both iteration count and execution time, as also evident from Table 1 and 2. The results clearly show that the trends in the graphs for iteration count and execution time align closely. Compared to other methods, the QSBTOR iterative scheme stands out, offering superior efficiency in terms of both performance time and the number of iterations.

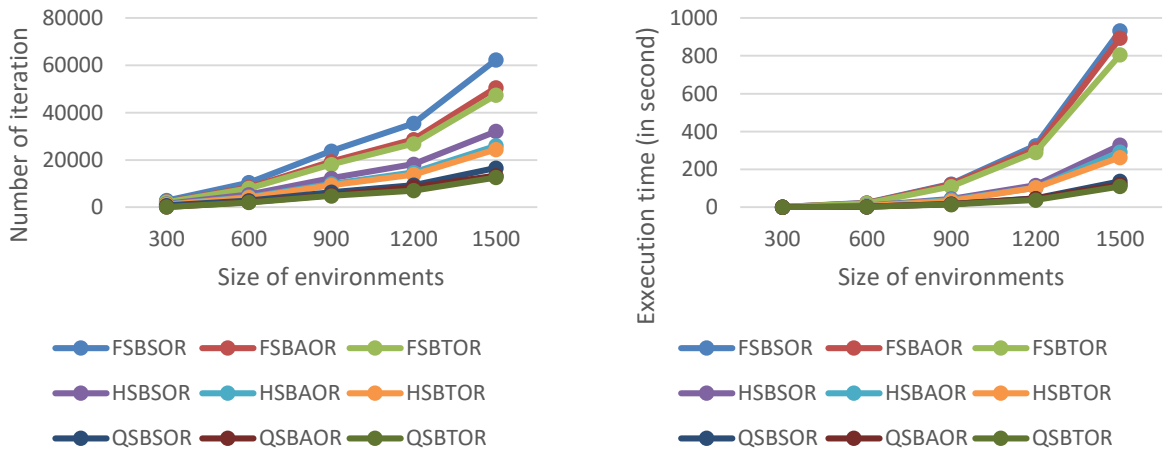


Fig. 4. Number of iterations (left) and performance time (right) by the tested methods for varying size of environments

Tables 3 to 5 display the total number of arithmetic operations acquired by all of the methods assessed, in which $M = N^2 - P$ is utilized to denote the number of cell points which are involved during

the iteration, where N^2 is the size of the environment and P represents the number of cell points occupied by obstacles.

Table 3
Number of arithmetic operations per iteration for algorithms on the 9-Point Laplacian using Boosted SOR methods

Methods	ADD/SUB	MUL/DIV
FSBSOR	$8N^2$	$3N^2$
HSBSOR	$4N^2$	$\frac{3}{2}N^2$
QSBSOR	$2N^2$	$\frac{3}{4}N^2$

Table 4
Number of arithmetic operations per iteration for algorithms on the 9-Point Laplacian using Boosted AOR methods

Methods	ADD/SUB	MUL/DIV
FSBAOR	$16N^2$	$5N^2$
HSBAOR	$8N^2$	$\frac{5}{2}N^2$
QSBAOR	$4N^2$	$\frac{5}{4}N^2$

Table 5
Number of arithmetic operations per iteration for algorithms on the 9-Point Laplacian using Boosted TOR methods

Methods	ADD/SUB	MUL/DIV
FSBTOR	$16N^2$	$7N^2$
HSBTOR	$8N^2$	$\frac{7}{2}N^2$
QSBTOR	$4N^2$	$\frac{7}{4}N^2$

The HS and QS algorithms based on the 9-Point iterative scheme employ additional arithmetic operations to calculate the remaining points after convergence by employing direct methods, as given in Table 6.

Table 6
Number of additional arithmetic operations for the remaining points for HS Boosted and QS Boosted methods

Methods	ADD/SUB		MUL/DIV		
	SOR	AOR/TOR	SOR	AOR	TOR
HS cases	$4N^2$	$8N^2$	$\frac{3}{2}N^2$	$\frac{5}{2}N^2$	$\frac{7}{2}N^2$
QS cases	$6N^2$	$12N^2$	$\frac{9}{4}N^2$	$\frac{15}{4}N^2$	$\frac{21}{4}N^2$

Table 7 presents a comparison of the Boosted AOR, Boosted SOR and Boosted TOR methods, clearly indicating that the Boosted TOR delivers the best performance in terms of iteration count and CPU time. The harmonic potentials computed using the proposed methods are utilized in the path generation process, which employs the Gradient Descent Search (GDS) algorithm. The GDS algorithm guides path generation by following the gradient of the harmonic potentials, starting from the point

with the highest potential value (the start point) and progressing to lower potential values until the lowest potential value, representing the goal point, is reached.

Table 7
Reduction percentages in terms of number of iterations and CPU time for HS Boosted and QS Boosted methods against their corresponding FS Boosted method

Methods	Iteration (%)	CPU Time (%)
HSBSOR	48.52	66.45
HSBAOR	48.64	68.97
HSBTOR	48.95	71.09
QSBSOR	72.75	86.53
QSBAOR	72.55	87.20
QSBTOR	72.92	87.23

Figure 5 illustrates the paths generated for Map 1 and 2 using the GDS algorithm, based on the harmonic potentials computed with the aforementioned iterative methods.

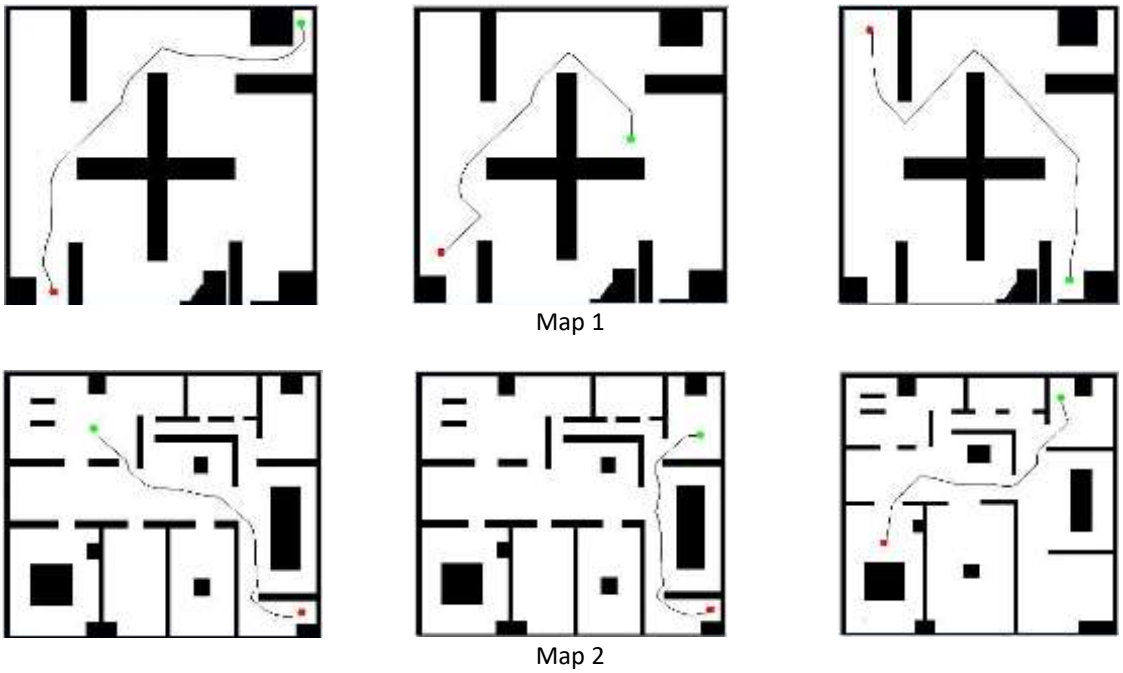


Fig. 5. Path creation for various surroundings using various starting locations (green point) and target places (red point)

The computational complexity of the tested methods is presented in Tables 3 to 5. According to these tables, the iterative techniques based on the Half-Sweep Boosted methods (HSBSOR, HSBAOR and HSBTOR) process only half of the node points in a skewed manner during the iteration process. Consequently, the computational complexity is reduced by approximately 50%. In contrast, the simulated results for the Quarter-Sweep Boosted methods (QSBSOR, QSBAOR and QSBTOR) demonstrate significantly better performance, reducing the computational complexity by approximately 75%.

4. Conclusions

The primary objective of this study is to formulate and develop a method for integrating iterative approaches with path-finding algorithms to solve the path-planning problem for mobile robots. This

work applies the concepts of Half-Sweep (HS) and Quarter-Sweep (QS) iterations, along with the computation of Laplacian harmonic potentials using the 9-point Laplacian. These techniques are employed to address the path-planning problem through the application of relaxation iterative methods. The overall performance of the path-planning algorithms is evaluated based on three key criteria: successful path generation, the number of iterations required and time efficiency.

The key contribution of this study, based on the summary of findings, is the introduction of the Quarter-Sweep Boosted TOR (QSBTOR) method within the 9-Point Laplacian operator framework. This method employed two acceleration parameters to speed up the convergence rate in computing the solutions of the Laplace's equation to obtain the harmonic potentials. The computed harmonic potentials are subsequently utilized by the Gradient Descent Search (GDS) algorithm to guide the path-finding process. This approach enables the generation of smooth paths, allowing the robot to navigate safely within a structured environment from any starting point to a specified goal while minimizing the risk of collisions.

References

[1] Hamd, Mostafa Mohammed Massoud, Ibrahim, Ahmed Abdellatif Hamed and Atia, Mostafa Rostom Ahmed. "Selecting Dynamic Path Planning Algorithm Based Upon Ranking Approach for Omni-Wheeled Mobile Robot". *Journal of Advanced Research in Applied Sciences and Engineering Technology* 41, no. 2 (2024): 125-138. https://doi.org/10.37934/araset.41.2.125138

[2] Sánchez-Ibáñez, José Ricardo, Carlos J. Pérez-del-Pulgar and Alfonso García-Cerezo. "Path planning for autonomous mobile robots: A review." *Sensors* 21, no. 23 (2021): 7898. https://doi.org/10.3390/s21237898

[3] Panchpor, Aishwarya A., Sam Shue and James M. Conrad. "A survey of methods for mobile robot localization and mapping in dynamic indoor environments." In *2018 Conference on Signal Processing And Communication Engineering Systems (SPACES)*, pp. 138-144. IEEE, 2018. https://doi.org/10.1109/SPACES.2018.8316333

[4] Xiu-xia, Yang, Cao Wei-yi, Zhang Yi, Fang Guo-wei and Yan Xuan. "Mobile robot path planning in complex environment." In *2019 IEEE International Conference on Unmanned Systems (ICUS)*, pp. 426-431. IEEE, 2019. https://doi.org/10.1109/ICUS48101.2019.8996020

[5] Shi, Kunju, Peng Wu and Mingshuai Liu. "Research on path planning method of forging handling robot based on combined strategy." In *2021 IEEE International Conference on Power Electronics, Computer Applications (ICPECA)*, pp. 292-295. IEEE, 2021. https://doi.org/10.1109/ICPECA51329.2021.9362595

[6] Rubio, Francisco, Francisco Valero and Carlos Llopis-Albert. "A review of mobile robots: Concepts, methods, theoretical framework and applications." *International Journal of Advanced Robotic Systems* 16, no. 2 (2019): 1729881419839596. https://doi.org/10.1177/1729881419839596

[7] LaValle, Steven M. "Motion Planning." *IEEE Robotics & Automation Magazine* 18, no. 1 (2011): 79-89. https://doi.org/10.1109/MRA.2011.940276

[8] Connolly, Christopher I., J. Brian Burns and Rich Weiss. "Path planning using Laplace's equation." In *Proceedings., IEEE International Conference on Robotics and Automation*, pp. 2102-2106. IEEE, 1990. https://doi.org/10.1109/ROBOT.1990.126315

[9] Faverjon, Bernard and Pierre Tournassoud. "A local based approach for path planning of manipulators with a high number of degrees of freedom." In *Proceedings. 1987 IEEE international conference on robotics and automation*, vol. 4, pp. 1152-1159. IEEE, 1987. https://doi.org/10.1109/ROBOT.1987.1087982

[10] Zelek, John S. and Martin D. Levine. "Local-global concurrent path planning and execution." *IEEE Transactions on Systems, Man and Cybernetics-Part A: Systems and Humans* 30, no. 6 (2002): 865-870. https://doi.org/10.1109/3468.895924

[11] Reif, John H. "Complexity of the mover's problem and generalizations." In *20th Annual Symposium on Foundations of Computer Science (sfcs 1979)*, pp. 421-427. IEEE Computer Society, 1979. https://doi.org/10.1109/SFCS.1979.10

[12] Karnik, Madhuri, Bhaskar Dasgupta and Vinayak Eswaran. "A comparative study of Dirichlet and Neumann conditions for path planning through harmonic functions." *Future Generation Computer Systems* 20, no. 3 (2004): 441-452. https://doi.org/10.1016/j.future.2003.07.008

[13] Sasaki, S. "A practical computational technique for mobile robot navigation." In *Proceedings of the 1998 IEEE International Conference on Control Applications (Cat. No. 98CH36104)*, vol. 2, pp. 1323-1327. IEEE, 1998. https://doi.org/10.1109/CCA.1998.721675

[14] Young, David. "Iterative methods for solving partial difference equations of elliptic type." *Transactions of the American Mathematical Society* 76, no. 1 (1954): 92-111. https://doi.org/10.1090/S0002-9947-1954-0059635-7

[15] Kew, Lee Ming and Norhashidah Hj Mohd Ali. "New explicit group iterative methods in the solution of three dimensional hyperbolic telegraph equations." *Journal of Computational Physics* 294 (2015): 382-404. https://doi.org/10.1016/j.jcp.2015.03.052

[16] Ling, W. K., A. A. Dahalan and A. Saudi. "Autonomous path planning through application of rotated two-parameter overrelaxation 9-point Laplacian iteration technique." *Indones. J. Electr. Eng. Comput. Sci* 22 (2021): 1116-1123. https://doi.org/10.11591/ijeecs.v22.i2.pp1116-1123

[17] Abdullah, Abdul Rahman. "The four point Explicit Decoupled Group (EDG) method: A fast Poisson solver." *International Journal of Computer Mathematics* 38, no. 1-2 (1991): 61-70. https://doi.org/10.1080/00207169108803958

[18] Othman, Mohamed and Abdul Rahman Abdullah. "An efficient four points modified explicit group poisson solver." *International Journal of Computer Mathematics* 76, no. 2 (2000): 203-217. https://doi.org/10.1080/00207160008805020

[19] Santos, J. L., W. S. Yousif and M. M. Martins. "The explicit group TOR method." *Neural Parallel and Scientific Computations* 20, no. 3 (2012): 459.

[20] Dahalan, A'Qilah Ahmad, Azali Saudi and Jumat Sulaiman. "Enhancing Autonomous Guided Vehicles with Red-Black TOR Iterative Method." *Mathematics* 11, no. 20 (2023): 4393. https://doi.org/10.3390/math11204393

[21] Ali, Norhashidah Mohd. "Reka Bentuk Algoritma Blok Baru Stensil 9-titik dalam Masalah Sempadan." *Matematika* (2002): 45-62.

[22] Ling, Sam Teek and Ali, Norhashidah Hj. Mohd. "New High Order Group Iterative Schemes in the Solution of Poisson Equation". *World Academy of Science, Engineering and Technology, International Journal of Mathematical, Computational, Physical, Electrical and Computer Engineering* 7, no. 11 (2013): 1694-1699. https://doi.org/10.5281/zenodo.1089343

[23] Saudi, Azali and Jumat Sulaiman. "Path planning simulation using harmonic potential fields through four point-EDGSOR method via 9-point laplacian." *Jurnal Teknologi (Sciences & Engineering)* 78, no. 8-2 (2016). https://doi.org/10.11113/jt.v78.9537

[24] Saudi, Azali and Jumat Sulaiman. "Path planning for indoor mobile robot using Half-Sweep SOR via nine-point Laplacian (HSSOR9L)." *IOSR Journal of Mathematics* 3, no. 2 (2012): 01-07. https://doi.org/10.9790/5728-0320107

[25] Ling, W. K., A. A. Dahalan and A. Saudi. "Mobile Robot Path Navigation in Static Indoor Environment via AOR 9-Point Laplacian Iteration Numerical Technique." *International Journal of Difference Equations (IJDE)* 15, no. 2 (2020): 231-242. https://doi.org/10.37622/IJDE/15.2.2020.231-242