# Laplacian Behaviour-Based Control (LBBC) for the Path Planning of Mobile Robot Via Four Point-EGSOR

Azali Saudi
School of Science and Technology
University Malaysia Sabah
Kota Kinabalu, Malaysia
azali60@gmail.com

Jumat Sulaiman
School of Science and Technology
University Malaysia Sabah
Kota Kinabalu, Malaysia
jumat@ums.edu.my

*Abstract*—**This paper proposed a behaviour-based paradigm approach known as Laplacian Behaviour-Based Control (LBBC) for solving path planning problem for a mobile robot operating in a structured indoor environment. LBBC relies on the use of Laplace's Equation to model the potential function in the environment model. For solving the Laplace's Equation, a numerical technique using a weighted block technique based on a block of four points known as Four Point-EGSOR (4EGSOR) iterative method is used to provide guidance in generating path for the robot. The simulation results show that LBBC provides robust motion for the robot, whilst 4EGSOR ensure faster computation than the previous methods.**

***Keywords: Mobile robot path planning, Behaviour-Based paradigm, Laplace's equation, Explicit Group, Four Point-EGSOR iterative method, harmonic functions***

## I. INTRODUCTION

One of the most difficult problems in robotics applications is developing robust autonomous motion planning. In order to build a truly autonomous mobile robot, it must have the capability to efficiently and reliably plan a route from start to the goal point without colliding with obstacles in between. Path planning algorithms attempt to deal with the problem of establishing a medium of communication between initial and final configurations, so that the robot can traverse the field safely. Various algorithms exist trying to solve this problem but all have shortcomings. The difficulty is due to the complexity of path planning problem, where it increases exponentially with the dimension of the configuration space.

In order to ensure completeness, every point in the configuration space has to be considered in the computation. Many global path planning methods presuppose a complete representation of the configuration space. Their main drawback is that at best they are computationally expensive and often intractable. Potential field and bug approaches are local methods that do not make this assumption but are not complete methods. Thus, produce the occurrence of local minima or loops that will often cause this class of path planners to fail. Our approach combines a local exploration method with a global numerical computation. We introduce a local control technique known as Laplacian Behaviour-Based Control (LBBC) for robust space exploration of the mobile robot. LBBC relies on the temperature values in the environment to guide its motion, which were calculated numerically using fast block iteration via Four Point-EGSOR (4EGSOR) method.

## II. LITERATURE REVIEW

The use of potential functions for robot path planning, as introduced by Khatib [1], views every obstacle to be exerting a repelling force on an end effector, while the goal exerts an attractive force. Koditschek [2], using geometrical arguments, showed that, at least in certain types of domains, there exists potential functions which can guide the effector from almost any point to a given point. These potential fields approach to path planning, however, suffer from the spontaneous creation of local minima.

Connolly et al. [3] and Akishita et al. [4] independently developed a global method using solutions to Laplace's equations for path planning to generate a smooth, collision-free path. The potential field is computed in a global manner, i.e. over the entire region, and the harmonic solutions to Laplace's equation are used to find the path lines for a robot to move from the start point to the goal point. Consequently, Connolly and Gruppen [5] reported that harmonic functions have a number of properties useful in robotic applications.

In the past, various methods have been proposed for solving linear system in order to obtain the harmonic functions. The standard methods are Jacobi, Gauss-Seidel and SOR [6]. Meanwhile, several others have reported the use of harmonic functions in robotics. Silva [7] used harmonic functions for robot exploration. Kazemi & Mehrandezh [8] employed harmonic function-based probabilistic maps for their sensor-based robot path planning. Rosell & Iniguez [9] combine harmonic functions with probabilistic cell decomposition for solving path planning

problem. Meanwhile, Daily and Bevly [10] used analytical solution for arbitrarily shaped obstacles. Garrido et al. [11] had applied finite elements to obtain harmonic functions for robotic motion. More recently, harmonic functions were used for real-time obstacles avoidance in Szulczyński et al. [12].

## III. BEHAVIOUR-BASED PARADIGM

Traditional approach robot programming assumes the availability of a complete and accurate model of the robot and its environment, relying on planners to generate actions. Unfortunately, this approach has several disadvantages. One main drawback is that they require huge amounts of computational resources. This drawback is much obvious for an autonomous mobile robot that must carry its own computational resources. Secondly, this approach must be based on highly accurate model, thus it requires a number of high-precision sensors which are also often expensive. These sensors, however, are subject to noisy data. Finally, this sense-plan-act paradigm is by nature sequential, thus it would fail if the world happens to change in between of phases. Furthermore, there is always delay between sensing and act, due to longer time required in planning.

As an alternative to the traditional approach, a new paradigm called subsumption architecture, also known as behaviour-based control, is devised [13]. In this architecture, sensors are dealt with only implicitly in that they initiate behaviours. Each behaviour is simply layers of control systems that all run in parallel. Higher level behaviours have the power to temporarily suppress lower level behaviours. Therefore, a set of priority scheme is used to resolve the dominant behaviour for a given scenario. A more rigorous explanation of behaviour-based approach for controlling robot is presented in [14].

In this work, inspired by the behaviour-based paradigm approach to robotics control [15], the searching algorithm employs Laplacian Behaviour-Based Control (LBBC) for robust space exploration of the configuration space. The LBBC comprises four core behaviours i.e. *keep-forward*, *follow-wall*, *avoid-obstacle*, and *find-slope*. All these core behaviours make use of the potential values represented by temperature distribution in the configuration space which are computed numerically to provide guidance during search exploration.

### A. Keep-Forward Behaviour

The *keep-forward* behaviour is a core behaviour that keeps the searching moving forward in the same direction as long as the temperature at current location is higher than the next location. When the searching encounters ascending slope, flat region, obstacles or walls, the *keep-forward* behaviour stops, and other behaviours would take over. The main aim of this behaviour is to guide the searching by following the descending slope until the goal location is found.

### B. Follow-Wall Behaviour

The *follow-wall* behaviour provides the search with the capability to follow the wall for a specified number of steps. With this behaviour, it will command the searching to keep turning gradually until its direction is parallel with the wall. It provides the searching with the capability of traversing the narrow path and sharp corner. In this implementation, the *follow-wall* behaviour is executed for every a specified number of steps. After that the searching switches to *find-slope* behaviour.

### C. Avoid-Obstacle Behaviour

When the searching hits an obstacle or wall, it will trigger the searching to backup and turn 90 degrees to the left or right alternately. By turning alternately to the left and right, it provides the searching with the capability to escape from a difficult position such as sharp corner.

### D. Find-Slope Behaviour

When the *find-slope* behaviour takes over, it will command the searching to move randomly hoping to encounter a descending slope that consequently triggers *keep-forward* behaviour. With this behaviour, the searching is capable of moving away from a flat region to continue its descending move towards goal location.

## IV. HARMONIC FUNCTIONS

A harmonic function on a domain $\Omega \subset R$ is a function which satisfies Laplace's equation,

$$\nabla^2 \phi = \sum_{i=1}^{n} \frac{\partial^2 \phi}{\partial x_i^2} = 0 \qquad (1)$$

where $x_i$ is the $i$-th Cartesian coordinate and n is the dimension. In the case of robot path construction, the boundary of $\Omega$ (denoted by $\delta\Omega$) consists of the outer boundary of the workspace and the boundaries of all the obstacles as well as the start point and the goal point, in a configuration space representation. The spontaneous creation of a false local minimum inside the region is avoided if Laplace's equation is imposed as a constraint on the functions used, as the harmonic functions satisfy the min-max principle. Hence the only types of critical points which can occur are saddle points. For a path-planning algorithm, an escape from such critical points can be found by performing a search in the neighbourhood of that point. Laplace's equation can be solved numerically. Standard methods are Jacobi and Gauss-Seidel, but faster computation can be obtained using Successive-Over-Relaxation (SOR) iterative method.

In the framework used in this study, the robot is represented by a point in the environment model or also known as configuration space. The path planning problem is then posed as an obstacle avoidance problem for the point robot from the start point to the goal point in the configuration space, which can have either square or rectangular outer boundaries, having projections or convolutions inside to act as barriers. Apart from projections of the boundaries, some obstacles inside the boundary are also considered. The configuration space is designed in grid or discrete form and the coordinates and function values associated

with each node are computed iteratively by applying numerical technique to satisfy equation in Eq. (1). The highest temperature is assigned to the start point whereas the goal point is assigned the lowest. In some cases with Dirichlet conditions, the start point is not assigned any temperature. In this study, Dirichlet boundary conditions are employed, thus the results are processed by assigning different temperature values to the boundaries and obstacles, and lowest temperature for the goal point. No temperature values are assigned to the start points. In this work, solution to the Laplace's equation were subjected to Dirichlet boundary conditions $\Phi \mid \delta\Omega = c$, where c is constant.

## V. The Formulation of Four Point-EGSOR Iterative Method

In the literature, Jacobi and Gauss-Seidel [6] had been used for solving any linear system. More recently, Daily and Bevly [10] use analytical solution for arbitrarily shaped obstacles. Others employed block iterative methods mainly on various points of Explicit Group (EG) methods including Evans & Yousif [16], Ibrahim [17] and Sulaiman et al. [18]. They pointed out that the block iterative method is more superior compared to the traditional point iterative methods. In robotics, our previous works on utilizing block iteration for solving robot path planning via Laplace's equation produce encouraging results [19 - 22], although they were carried out without LBBC. In [23], we introduce the use of LBBC for robust robot exploration.

Let us consider the two-dimensional Laplace equation in Eq. (1) defined as

$$\frac{\partial^2 U}{\partial^2 x} + \frac{\partial^2 U}{\partial^2 y} = 0 \tag{2}$$

By using the second-order central difference scheme, we can simplify the five point second-order standard finite difference approximation equations for problem (2) as generally stated in the following equation

$$U_{i-1,j} + U_{i+1,j} + U_{i,j-1} + U_{i,j+1} - 4U_{i,j} = 0 \tag{3}$$

The equation in Eq. (3) shown above is the standard Gauss-Seidel iterative method for solving linear system. To enhance convergence speed, an approach called Successive Over-Relaxation (SOR) method is added to Eq. (3), as can be shown below (Young [24 - 26]):

$$U_{i,j}^{k+1} = \frac{\omega}{4}(U_{i-1,j}^{k+1} + U_{i+1,j}^{k} + U_{i,j-1}^{k+1} + U_{i,j+1}^{k}) + (1-\omega)U_{i,j}^{k} \tag{4}$$

where the optimal value of $\omega$ is defined in the range, $1 \le \omega < 2$. In practice, several runs of computer program implementation of Eq. (4) are carried out with different value of ω. The value of ω is considered optimal when the program converges with the less number of iterations. By taking ω = 1, the SOR iterative method will represent Gauss-Seidel method.

To examine the effectiveness of the block iterative method, let us consider a block of four node points as shown in Fig. 1 and defined as

$$\begin{bmatrix} 4 & -1 & -1 & 0 \\ -1 & 4 & 0 & -1 \\ -1 & 0 & 4 & -1 \\ 0 & -1 & -1 & 4 \end{bmatrix} \begin{bmatrix} U_{i,j} \\ U_{i+1,j} \\ U_{i,j+1} \\ U_{i+1,j+1} \end{bmatrix} = \begin{bmatrix} S_1 \\ S_2 \\ S_3 \\ S_4 \end{bmatrix} \tag{5}$$

where

$$S_1 = U_{i-1,j} + U_{i,j-1},$$
$$S_2 = U_{i+2,j} + U_{i+1,j-1},$$
$$S_3 = U_{i-1,j+1} + U_{i,j+2},$$
$$S_4 = U_{i+2,j+1} + U_{i+1,j+2}.$$

Determining the inverse matrix of the coefficient matrix in Eq. (5), the general scheme of Eq. (5) can be rewritten as

$$\begin{bmatrix} U_{i,j} \\ U_{i+1,j} \\ U_{i,j+1} \\ U_{i+1,j+1} \end{bmatrix} = \frac{1}{24} \begin{bmatrix} 7 & 2 & 2 & 1 \\ 2 & 7 & 1 & 2 \\ 2 & 1 & 7 & 2 \\ 1 & 2 & 2 & 7 \end{bmatrix} \begin{bmatrix} S_1 \\ S_2 \\ S_3 \\ S_4 \end{bmatrix} \tag{6}$$

The implementation of Eq. (6) can be processed iteratively to compute the value of 4 node points simultaneously, as shown in Eq. (7).

$$\begin{aligned} U_{i,j} &= \frac{1}{24}(7S_1 + 2S_2 + 2S_3 + S_4) + U_{i,j}, \\ U_{i+1,j} &= \frac{1}{24}(2S_1 + 7S_2 + S_3 + 2S_4) + U_{i+1,j}, \\ U_{i,j+1} &= \frac{1}{24}(2S_1 + S_2 + 7S_3 + 2S_4) + U_{i,j+1}, \\ U_{i+1,j+1} &= \frac{1}{24}(S_1 + 2S_2 + 2S_3 + 7S_4) + U_{i+1,j+1}. \end{aligned} \tag{7}$$

By adding a weighted parameter ω to Eq. (7), the implementation of the 4EGSOR iterative method can be shown as

$$\begin{aligned} U_{i,j}^{k+1} &= \frac{\omega}{24}(7S_1 + 2S_2 + 2S_3 + S_4) + (1-\omega)U_{i,j}^{k}, \\ U_{i+1,j}^{k+1} &= \frac{\omega}{24}(2S_1 + 7S_2 + S_3 + 2S_4) + (1-\omega)U_{i+1,j}^{k}, \\ U_{i,j+1}^{k+1} &= \frac{\omega}{24}(2S_1 + S_2 + 7S_3 + 2S_4) + (1-\omega)U_{i,j+1}^{k}, \\ U_{i+1,j+1}^{k+1} &= \frac{\omega}{24}(S_1 + 2S_2 + 2S_3 + 7S_4) + (1-\omega)U_{i+1,j+1}^{k}. \end{aligned} \tag{8}$$

As shown in Eq. (7) and Eq. (8), with 4EGSOR, four node points are computed simultaneously in one loop of iteration. Thus, greatly speed up the computation of all points in the environment.

## VI. Experiments and Results

The experiment considered various size of static environment, i.e. 128x128, 256x256, and 512x512, that consists of a goal point, several starting points and varying setup of walls and obstacles. Initially, the boundaries (walls) and obstacles were fixed with high temperature values. Goal point was set to very low temperature. All other free spaces were set to zero temperature value. Then, the iteration process was run on Intel Core 2 Duo CPU running at 3GHz speed with 1GB of RAM to compute temperature values numerically at all points in the environment. The iteration process was terminated when there was no more changes in temperature values, where it converged to a specified very small value, i.e. $1.0^{-10}$. The highest precision of solution for Eq. (1) was required to reduce the occurrence of flat area, hence would speed up the searching algorithm during path planning construction of the mobile robot from starting point to goal point. Table I shows the number of iterations, maximum error and elapsed time in (m:s:ms) for computing all temperature values in the environment. Clearly, 4EGSOR performs much faster than the previous methods as shown in Figure 2.

Once the temperature values were obtained, the searching algorithm would make use of them to guide its exploration. In the previous work, the path can be generated successfully even without LBBC, if the environment space was simple and sparse in which the gradient from start points to goal point are smooth, as shown in Figure 3(a). However, the searching algorithm failed to reach the goal point when the horizontal wall was extended. As shown in Figure 3(b), only one path was successfully generated, whereas the other two start points got stuck in the flat region. By employing LBBC, the searching would be able to escape from flat region and continue its exploration by utilizing *follow-wall* behaviour to reach the goal point, see Figure 3(c). As shown in Figure 3(d), the temperature values of the walls and the generated paths are raised up for visualization purpose. The lowest temperature indicates the goal point. All other areas are almost flat due to very small difference in temperature values, except for the area close to the goal point.

## VII. CONCLUSION

In conclusion, the simulation demonstrates that LBBC provides robust exploration for the robot. Meanwhile, complete search offered by numerical technique via block iterative method is indeed very attractive and feasible for solving difficult robot path planning problem. As shown in Fig. 2, 4EGSOR proved to be very fast compared to the previous iterative methods. Table I shows that 4EGSOR performed much better than EGSOR, in which the number of iteration was reduced by more than 30%. In the future work, we would consider faster numerical implementation by employing half-sweep iteration, Muthuvalu & Sulaiman [27, 28] and Akhir et al. [29], to compute the solution of Laplace's equation for robot path planning

TABLE I. Performance comparison of several iterative methods in varying sizes of environment.

| | Iterative Methods | Size of environments | | |
|---|---|---|---|---|
| | | 128x128 | 256x256 | 512x512 |
| Number of iterations | GS | 21552 | 78522 | 281220 |
| | SOR | 1314 | 5059 | 18699 |
| | EGSOR | 962 | 3768 | 13982 |
| | 4EGSOR | 640 | 2609 | 9783 |
| Maximum error | GS | $0.9993^{-10}$ | $0.9988^{-10}$ | $0.9996^{-10}$ |
| | SOR | $0.9952^{-10}$ | $0.9985^{-10}$ | $0.9998^{-10}$ |
| | EGSOR | $0.9897^{-10}$ | $0.9983^{-10}$ | $0.9998^{-10}$ |
| | 4EGSOR | $0.9848^{-10}$ | $0.9983^{-10}$ | $0.9986^{-10}$ |
| Elapsed time (m:s:ms) | GS | 0:21:297 | 6:28:235 | 49:18:32 |
| | SOR | 0:1:344 | 0:24:641 | 7:11:188 |
| | EGSOR | 0:1:109 | 0:19:828 | 5:39:250 |
| | 4EGSOR | 0:0:765 | 0:13:485 | 3:46:328 |

GS: Gauss-Seidel; SOR: Successive Over-Relaxation;
EGSOR: Explicit Group with SOR; EGSOR9L: EGSOR on Nine-Point Laplacian.

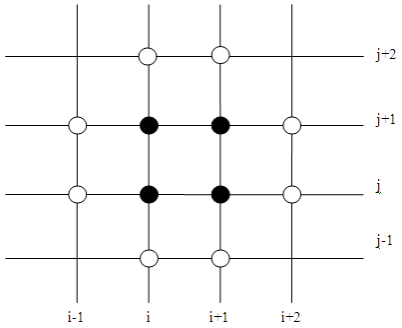



Figure 1: Illustration of block iteration method based on four points.

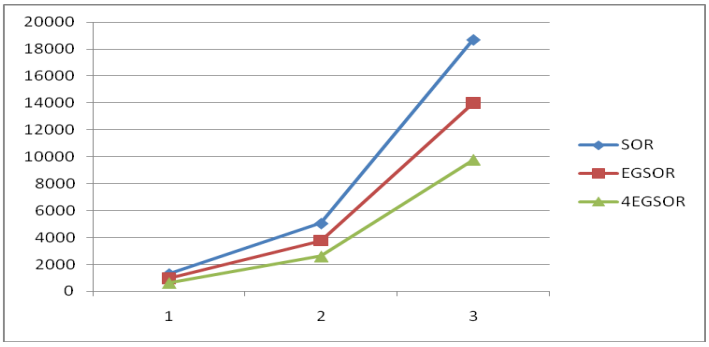
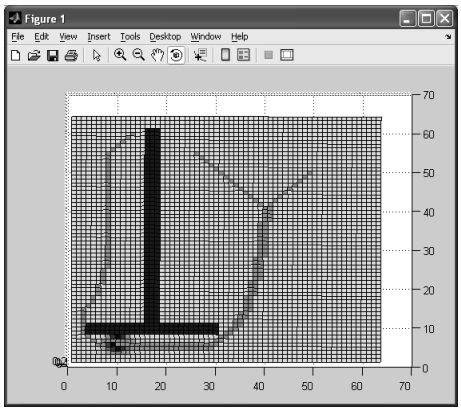
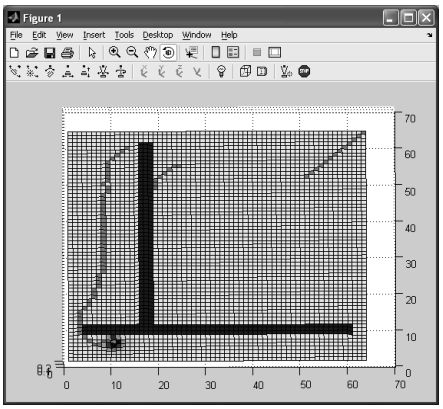


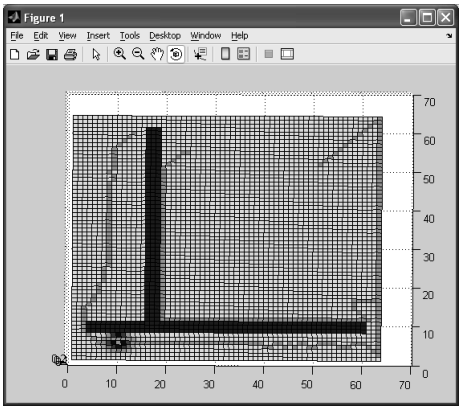

Figure 2: Number of iterations against sizes of environment for various iterative methods.



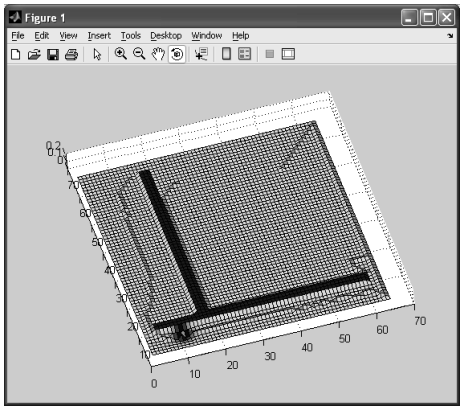

(a)




(b)




(c)




(d)

Figure 3: (a) Path is successfully generated in a simple and sparse environment, from three starting points to a goal point. (b) The path generation process failed to reach the goal point when the length of horizontal wall is extended twice to the right. (c) With LBBC, the algorithm simply utilized the follow-wall behaviour to escape from flat region and keep moving to find the goal point. (d) The 3D view of the environment and generated path from three start points to a goal point.

## REFERENCES


[1] Khatib, O. 1985. Real time obstacle avoidance for manipulators and mobile robots. IEEE Transactions on Robotics and Automation 1:500–505.

[2] Koditschek, D. E. 1987. Exact robot navigation by means of potential functions: Some topological considerations. Proceedings of the IEEE International Conference on Robotics and Automation: 1-6.

[3] Connolly, C. I., Burns, J. B., & Weiss, R. 1990. Path planning using Laplace's equation. Proceedings of the IEEE International Conference on Robotics and Automation: 2102–2106.

[4] Akishita, S., Kawamura, S., & Hayashi, K. 1990. Laplace potential for moving obstacle avoidance and approach of a mobile robot. Japan-USA Symposium on flexible automation, A Pacific rim conference: 139–142.

[5] Connolly, C. I., & Gruppen, R. 1993. On the applications of harmonic functions to robotics. Journal of Robotic Systems, 10(7): 931–946.

[6] Sasaki, S. 1998. A Practical Computational Technique for Mobile Robot Navigation. Proceedings of the IEEE International Conference on Control Applications: 1323-1327.Connolly, C. I., & Gruppen, R. 1993. On the applications of harmonic functions to robotics. Journal of Robotic Systems, 10(7): 931–946.

[7] Silva, E.P., Engel, P.M, Trevisan, M. & Idiart, M.A.P. 2002. Exploration method using harmonic functions. Robotics and Autonomous Systems, Elsevier. Vol. 40 (2002), Page(s): 25 – 42.

[8] Kazemi, M & Mehrandezh, M. 2004. Robotic navigation using harmonic function-based probabilistic roadmaps. Proc. of the IEEE International Conference on Robotics and Automation, 2004 (ICRA '04).

[9] Rosell, J. & Iniguez, P. 2005. Path planning using Harmonic Functions and Probabilistic Cell Decomposition. Proc. of the IEEE International Conference on Robotics and Automation, 2005 (ICRA2005), Apr 18 – 22. Page(s): 1803 – 1808. ISBN: 0-7803-8914-X.

[10] Daily, R. & Bevly, D.M. 2008. Harmonic Potential Field Path Planning for High Speed Vehicles. In the proceeding of American Control Conference, Seattle, June 11-13, 4609-4614.Evans, D. J. 1985. Group Explicit Iterative methods for solving large linear systems. Int. J. Computer Maths., 17: 81-108.

[11] Garrido, S., Moreno, L., Blanco, D. & Monar, F.M. 2010. Robotic Motion Using Harmonic Functions and Finite Elements. Journal of Intelligent and Robotic Systems archive. Volume 59, Issue 1, July 2010. Pages 57 – 73.

[12] Szulczyński, P., Pazderski, D. & Kozłowski, K. 2011. Real-Time Obstacle Avoidance Using Harmonic Potential Functions. Journal of Automation, Mobile Robotics & Intelligent Systems. Volume 5, No 3, 2011.

[13] R. A. Brooks. 1986. A robust layered control system for a mobile robot, IEEE Journal of Robotics and Automation, 2:(14-23).

[14] R. C. Arkin. 1998. Behaviour-based robotics, Bradford Books.

[15] Saudi, A. & Hallam, B. 2004. A Tale of Two Behaviour-based Robots. Mindstorms vs Technic. In proc. Of IEEE Region 10 Conference (TENCON 2004), 21-24 Nov. 2004. Page(s): 479 - 482 Vol. 4. ISBN: 0-7803-8560-8.

[16] Evans, D.J & Yousif, W. S. 1986. Explicit Group Iterative Methods for solving elliptic partial differential equations in 3-space dimensions. Int. J. Computer Maths., 18:323-340.

[17] Ibrahim, A.. 1993. The Study of the Iterative Solution Of Boundary Value Problem by the Finite Difference Methods. PhD Thesis. Universiti Kebangsaan Malaysia.

[18] Sulaiman, J., Hasan, M.K. & Othman, M. 2007. Red-Black EDGSOR Iterative Method Using Triangle Element Approximation for 2D Poisson Equations. In. O. Gervasi & M. Gavrilova (Eds). Computational Science and Its Application 2007. Lecture Notes in Computer Science (LNCS 4707): 298-308. Berlin: Springer-Verlag.

[19] Saudi, A. & Sulaiman, J. 2009. Block Iterative Method for Robot Path Planning. The 2nd Seminar on Engineering and IT 2009 (SEIT09), Kota Kinabalu, July 7 – 8, 2009.

[20] Saudi, A. & Sulaiman, J. 2010. Block Iterative Method using Nine-Point Laplacian for Robot Path Planning. European Journal of Scientific Research. ISSN 1450-216X Vol.43 No.2 (2010), pp.204-211.

[21] Saudi, A. & Sulaiman, J. 2010. Numerical Technique for Robot Path Planning using Four Point-EG Iterative Method. In proc. of the 2010 Int. Symposium on Information Technology (ITSim), Page(s): 831 – 836. ISBN: 978-1-4244-6715-0.

[22] Saudi, A. & Sulaiman, J. 2012. Robot Path Planning Using Four Point-Explicit Group Via Nine-Point Laplacian (4EG9L) Iterative Method. Journal of Procedia Engineering Volume 41, 2012, Pages 182–188. Int. Symposium on Robotics and Intelligent Sensors 2012 (IRIS 2012). ISSN: 1877-7058.

[23] Saudi, A. & Sulaiman, J. 2012. Laplacian Behaviour-Based Control for Robot Path Planning using Full-Sweep Successive Over-Relaxation via Nine-Point Laplacian (FSSOR9L). International Journal of Applied

Science and Technology, Vol. 2 No. 3; March 2012, pp: 255 - 261. ISSN 2221-0997.

[24] Young, D.M. 1971. Iterative solution of large linear systems. London: Academic Press.

[25] Young, D.M. 1972. Second-degree iterative methods for the solution of large linear systems. Journal of Approximation Theory. 5:37-148.

[26] Young, D.M. 1976. Iterative solution of linear systems arising from finite element techniques. In: Whiteman, J.R. (Eds.). The Mathematics of Finite Elements and Applications II: 439-464. London: Academic Press.

[27] Muthuvalu, M. S., & Sulaiman, J. 2011. Half-Sweep Arithmetic Mean method with composite trapezoidal scheme for solving linear Fredholm integral equations. Applied Mathematics and Computation. 217(12):5442-5448. ISSN: 0096-3003.

[28] Muthuvalu, M. S., & Sulaiman, J. 2012. Half-Sweep Geometric Mean Iterative Method for the Repeated Simpson Solution of Second Kind Linear Fredholm Integral Equations. Proyecciones Journal of Mathematics, 31(1): 65-79. ISSN: 0717-6279.

[29] Akhir, M.K.M, Othman, M., Sulaiman, J., Majid Z.A, & Suleiman, M. 2012. Half-Sweep Iteration for Solving Two-Dimensional Helmholtz Equations. International Journal of Applied Mathematics and Statistics, Vol. 29(5):101-109. ISSN: 0973-7545.