VFAST Transactions on Mathematics
http://vfast.org/journals/index.php/VTM@ 2024 ISSN(e): 2309-0022, ISSN(p): 2411-6343
Volume 12, Number 1, January-June, 2024 *pp: 323-335*

Parallel Numerical Solution of 2D Electrostatics Poisson Equation on Different Mesh Partitioning Schemes

Shakeel Ahmed Kamboh[1*], **Faiza Khanam**[1], **Nadeem Naeem**[2], **Sajida Parveen**[3], **Sakina Kamboh**[4], **Safina Kamboh**[5]

[1*]Department of Mathematics and Statistics, Quaid-e-Awam University of Engineering, Science and Technology, 67480, Nawabshah (Pakistan); [2]Department of Electronic Engineering, QUEST, Science and Technology, 67480, Nawabshah (Pakistan); [3]Department of Software Engineering,QUEST, 67480, Nawabshah (Pakistan); [4]Department of Statistics, University of Sindh, 76080, Jamshoro (Pakistan); [5]USPCAS-W, Mehran University of Engineering and Technology, Jamshoro

Keywords: *2D electrostatics Poisson equation, Finite difference method, parallel computing, numerical simulation, electric potential and electric field.* ***Subject Classification***:*Mathematical Modeling.*

Journal Info:
Submitted:
May 15, 2024
Accepted:
June 27, 2024
Published:
June 30, 2024

Abstract The ideas of parallelism for the large scale problems or problems with dense meshes have gained much attention in last few decades. The key goal of applying the parallelization is to reduce the computational time. In this paper; the 2D finite difference mesh partitioning schemes and their effect on performance of parallel numerical solution is evaluated. The main objective was to investigate the mesh partitioning schemes for less computational time and high speedup. For testing and implementation purpose a 2D electrostatics Poisson's equation with Dirichlet and Neumann boundary conditions applied on a 2D cross section of Electrohydrodynamic (EHD) planar ion-drag micropump is used to simulate the electric potential and electric field on a parallel system. The performance of the 7 different mesh partitioning schemes (PS) in terms of computational time, speedup, efficiency and communication cost was evaluated. It was revealed that among the seven different partitioning schemes the PS-3 (two-way or tile partitioning) is found the best scheme for the parallel numerical simulation of the problem. Moreover, the parallel algorithm remains more efficient on $P = 2$ to $P = 8$ workers while for $P > 8$ the efficiency of the algorithm may drop because of the high communication time.

***Correspondence Author Email Address:**
shakeel.maths@yahoo.com
DOI: 10.21015/vtm.v12i1.1847

1 Introduction

The parallel numerical simulations of various problems in science and engineering are mainly used to reduce the computational time. While the design of the parallel algorithms lies on the foremost step called the partitioning of domain into small parts (alternatively the mesh partitioning, also called graph partitioning). The better parallel performance can be achieved by selecting the most feasible mesh partitioning scheme. However, the selection of best possible mesh partitioning scheme is challenging as it depends upon the size and type of the problem and the inter processor communications. Most of the studies have been done on finite element meshes which require very complicated formulation and difficult implementation. Where as the finite difference method based partitioning schemes are very easy to implement on parallel computers as such meshes could be partitioned among the processors with the common almost equal number of mesh points with the reduced communication cost.

In the method, the geometric domain is discretized in the form of a mesh, where the core of the computation is comprised of the interactions of each mesh point with its immediate neighbors. In a parallel computing environment, such meshes should be partitioned among the processors with the common objectives of load balance (e.g., almost equal number of mesh points per processors) and reduced communication cost [1, 9]. The partitioning methods search for a balanced partition of the mesh points among a given number of parts with the objective of minimizing the number of interacting pairs that are assigned to different processors. In general, one uses an imbalance parameter in the problem dentition to define an allowable upper bound for the part weights [9]. The optimal partitioning of irregular and nonstructural graphs is the key to efficient scientific simulation on high performance parallel machines. In literature, different mesh partitioning techniques are reported having their own pros and cons. For instances; the spatial mesh partitioning scheme exploit geometry to capture global replacement of mesh nodes using spatial coordinates, yet may not be suitable descriptor of mesh nodes proximity. Non-spatial mesh partitioning uses neighborhood of mesh nodes for local connectivity, it employs the greedy approach for partitioning but can be vulnerable to local optima [25].

Single-level mesh partitioning is simple to implement and fast for small meshes but is slow for large meshes [13]. Multi-level mesh partitioning for large meshes in which the refinement phase improves quality of partitioning [7]. However, it is highly complex in implementation due to coarsening and uncoarsening phases [17]. Graph based mesh partitioning employs well-developed graph theoretic approaches in which complex irregular FEM meshes are mapped to graphs for transversing and searching. This technique could be costly in dynamic mesh partitioning approach [16, 22, 26]. The tree-based mesh partitioning is commonly employed in dynamic mesh partitioning and mesh reordering. It assists in the partitioning meshes of tree data structure; however the mapping of mesh to a tree is time consuming [17]. A serial mesh partitioning works on single machine in which small meshes can be partitioned sequentially requiring no parallel overheads. But this type of partitioning is not viable for large meshes requiring more storage resources [2]. Parallel mesh partitioning is appropriate for large graphs for fast partitioning; it utilizes available processing power and memory while the load balancing in an evolving distributed mesh is complicated [11].

A single objective mesh partitioning takes the advantages of well established optimization theory and easy to find the optimal solution. It is not appropriate for multi-objective and multi-constraint problems. As an alternative of single-objective partitioning, multi-objective partitioning helps in finding an optimal solution in large and difficult problems; but lead to the complicated implementation. [20] Simple mesh

partitioning is easy to program, to modify and debug and is also assumed to consume less time but may compromise the quality of partitioning. Complex mesh partitioning produces good partitioning results but complicated and requires longer time for execution [13, 19]. A Mesh reordering improves performance for distributed irregular meshes by increasing cache hit rate but may not provide significant results for regular meshes [17]. However, simpler geometric methods have proven to be highly effective for numerical simulations, while providing reasonably good decompositions for mesh-based solvers [23]. Some other partitioning schemes such as multiphase graph partitioning scheme that attempt to minimize interprocessor communication costs subject to the constraint that each component of the load is balanced is discussed by [5]; the initial partitioning phase, where a partitioning of the coarsest graph is quickly computed. This can be performed by a task decomposition scheme [6]. The coarsest graph is assembled on a single processor and then broadcast to all of the processors. Each processor then computes the same bisection of this graph concurrently [18]. A hyper graph partitioning studied by [21] which is a generalization of a graph, where the set of edges is replaced by a set of hyperedges. A hyperedge extends the notion of an edge by allowing more than two vertices to be connected by a hyperedge. Similarly, the element-based partitioning focuses on partitioning elements in finite element meshes, while the node-based partitioning focuses on partitioning nodes. For example, in parallel solutions of structural dynamics, implicit solution methods often require element-based partitioning [10, 15]. Some other applications of parallelization of numerical methods have recently been reported. To carry out the numerical analysis faster, an indigenous parallel FVM code was developed using OpenMP paradigm for numerical analysis of conjugate heat transfer and fluid flow. OpenMP parallelization of the FVM code provides a maximum speedup of up to 1.5 for considered conditions [12]. [24] Describe PyNumero, an open-source, object-oriented programming framework in Python for structured nonlinear programming problems (NLP's) using the Message Passing Interface (MPI). The parallel solution of a two-dimensional PDE optimal control problem was obtained and it was shown that the proposed provides nearly perfect scaling to more than $1,000$ cores for large matrix-vector dot products and structured linear systems. A parallel of high-order gas-kinetic scheme (HGKS) code was developed for large-scale computation using direct numerical simulation (DNS) [3]. The standard tests were presented to validate the parallel scalability, efficiency, accuracy and robustness of parallel implementation. Meanwhile, based on the kinetic formulation HGKS shows advantage for supersonic turbulent flow simulation with its accuracy and robustness.

An original numerical modelization approach for Moving Load (ML) beam problems based FEM on implemented on C++ parallel computing with the purposes of performing efficient numerical analyses was presented [8]. The proposed idea may serve as a guideline paradigm in adventuring the computational challenges involved in the present mechanical research context. In [4] the parallel implementations of the fourth-order Runge-Kutta method (RK4) for sparse matrices on Graphics Processing Unit (GPU) architecture with Compute Unified Device Architecture (CUDA) was investigated. The result demonstrates that CUDA has a significant advantage and performance as compared with the MPI implementation since the computational cost is tiny. Similarly, a parallel finite element method based on two-grid discretizations for the 2D/3D Navier–Stokes equations with damping has been worked in [27].

From the literature search it was observed that most of the studies have been done on finite element meshes which require very complicated formulation and difficult implementation. However, the finite difference method based partitioning schemes are very easy to implement on parallel computers as such meshes could be partitioned among the processors in a regular way. The choice of simple mesh partitioning in finite difference method could be further extended by exploring the different ways of partitioning

the given mesh, such as diagonal and window partitioning which is rarely found in literature. Thus the aim of this study is the investigation of the effect of different simple mesh partitioning schemes on parallel performance and it is expected that the outcome of this study will enable to decrease the computational time on dense and refined meshes.

2 Methodology

Consider a cross section of a prototype of planar ion-drag micropump whose all geometric configurations and boundary conditions are shown in the following Figure 3.1.

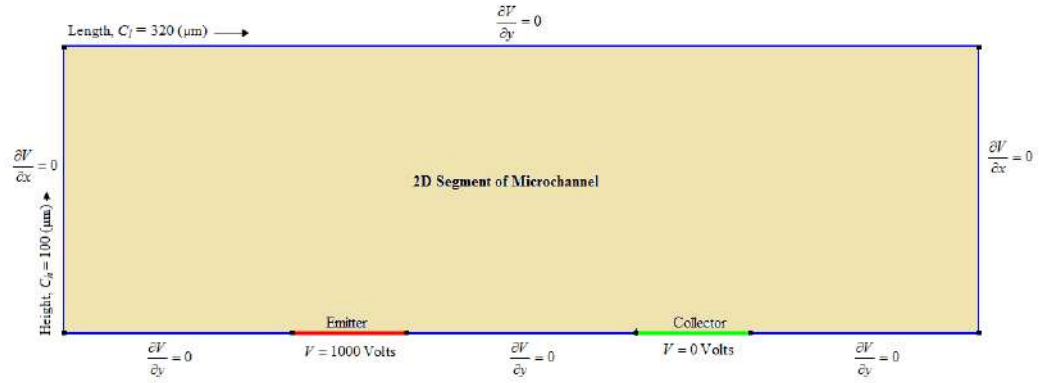

Figure 1. Computational domain (a cross section) of the EHD ion-drag micropump

The distribution of electric potential in EHD ion-drag pumping is governed by the following 2D elliptic partial differential equation specifically called the electrostatics Poisson's equation [14].

$$\nabla^2.V = -\frac{q_e}{\epsilon} \tag{1}$$

or in Cartesian form:

$$\frac{\partial^2 V}{\partial x^2} + \frac{\partial^2 V}{\partial y^2} = -\frac{q_e}{\epsilon} \tag{2}$$

also the application of electric potential generates the electric field in micropump which is given by $\vec{E} = -\nabla.V$ where V is the electric potential field (V), q_e the space charge density (C/m^3), ϵ the relative permittivity (F/m), $\vec{E}$ the electric field (V/m). The initial and boundary conditions for the above problem are defined as below; $V(x, y) = 0$, $V(Emitter) = 1000$, $V(collectro) = 0$ and $\frac{\partial V}{\partial n}(Walls) = 0$ where n is the normal vector. In the 2D case, Eq. (2) is discretized using central finite difference approximations for the key output variable V. The new form of the equation, expressed by the finite difference schemes, is:

$$\frac{V_{i+1,j} - 2V_{i,j} + V_{i-1,j}}{\Delta x^2} + \frac{V_{i,j+1} - 2V_{i,j} + V_{i,j-1}}{\Delta y^2} = -\frac{q_{ei,j}}{\epsilon} \tag{3}$$

where (i, j) are the indices of mesh nodes; (h, k) are the step size along (h, k) respectively. A schematic of the finite difference mesh of the problem is shown in the following Figure 2.

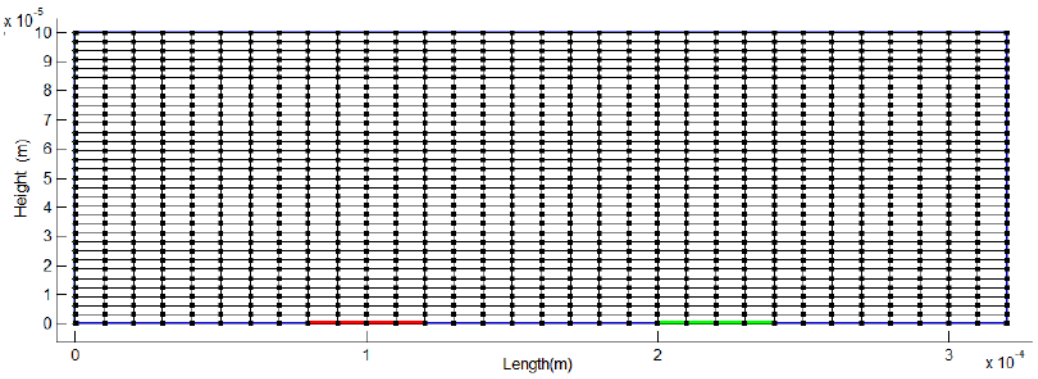

Figure 2. A schematic of the discretization of the computational domain into a mesh

After the discritization of the domain the mesh partitioning strategies will be applied. The mesh partitioning will be acquired by splitting the domain into the different sub-domains as $D_1, D_2, ..., D_P$. The sub-domains must satisfy the condition.

$\bigcup_{t=1}^{p} D_i = D$ and $D_i \cap D_j = \{\}, i, j = 1, 2, 3 \ldots, P$ (4)

where P is the number of selected workers used to solve the problem in parallel computation. The mesh partitioning of the problem is proposed in the different ways as listed below:

- PS-1: Horizontal mesh partitioning (1d partitioning-partitioning the points along y-axis)
- PS-2: Vertical mesh partitioning (1d partitioning-partitioning the points along x-axis)
- PS-3: Two way partitioning (2d partitioning-partitioning the points along x-axis and y-axis)
- PS-4: Diagonal partitioning (Top Left-Bottom Right partitioning)
- PS-5: Off diagonal partitioning (Top Right-Left-Bottom partitioning
- PS-6: Two way diagonal partitioning
- PS-7: Window Partitioning

In order to implement the parallel algorithm the most straightforward approach may be the data parallel approach in which the portions of complete mesh are distributed among the parallel workers such that each worker computes the solution on its own data and finally the results are collected from each and every worker. In addition to distribution of the sub domains the dependencies are treated. For that purpose, the message passing schemes are required to share the boundaries of distributed domain among the parallel workers. Therefore, the distribution and message passing schemes related to the central finite difference method are formulated properly.

Let $I+1$ mesh data points lie on any axis and required to distribute among P workers then $\lfloor (I+1)/P \rfloor = I_2$ points are assigned to all workers equally. But if $r = mod(I+1, p)$then r < P data points are remaining behind then it is required to find the maximum possible number of workers that may distribute r data points equally. Assume that there are w_{max} workers are going to utilize this data equally then must satisfy the modular expression mod $(r, w_{max}) = 0$, where $1 \leq w_{\max} \leq P-1$

Once, w_{max} is found the one can easily distribute rest of the data points to w_{max} workers equally. This states that first workers will have I_2+1 data points and other $P-w_{max}$ workers will have I_2 data points. This distribution scheme is generalized for each individual worker and described by the global indices of the distribution data as follows:

$$\begin{cases} 1+(p-1)\left(l_2+1\right) \le i \le l_2+1+(p-1)\left(l_2+1\right), & \text{when } p \le w_{\max} \\ w_{\max}+1+(p-1)l_2 \le i \le w_{\max}+l_2+(p-1), & p > w_{\max} \end{cases} \tag{5}$$

where p is the index of worker and i is the global index of mesh points along the partitioned axis.

It is also attempted to formulate the schemes for sharing the neighboring boundaries among the parallel workers. Assume that ϕ is a mapping for sending the data from p^{th} worker to its left $(p-1)^{th}$ or $(p+1)^{th}$ worker that sends the neighborhood boundary $V(x_{nbhd}, y)$, x_{nbhd} may be the first or the last boundary of p^{th} worker. It can be expressed mathematically as follows:

$$\Phi(p, p-1) = V\left(x_{\text{nhbd}}, y\right) \tag{6}$$

$$\Phi(p, p+1) = V\left(x_{\text{nhbd}}, y\right) \tag{7}$$

Eq. (6) and Eq. (7) are mappings for sending the mesh data to the left and the right workers respectively. The general for of these schemes is given below;

$$\begin{cases} \Phi(p, p+1) = V\left(l_2+1, y\right), \text{ if } 1 < p \le w_{\max} \\ \Phi(p, p+1) = V\left(l_2, y\right), \text{ if } w_{\max} < p < P \\ \Phi(p, p-1) = V(1, y), \text{ if } 1 < p \le P \end{cases} \tag{8}$$

Similarly, assume that ψ is the mapping for receiving the mesh data from p^{th} worker from left $(p-1)^{th}$) or right $(p+1)^{th}$) worker that receives the neighborhood boundary $V(x_{nbhd}, y)$. It can be represented mathematically as given below:

$$\Psi(p-1, p) = V\left(x_{\text{nbhd}}, y\right) \tag{9}$$

$$\Psi(p+1, p) = V\left(x_{\text{nbhd}}, y\right) \tag{10}$$

$$\begin{cases} \Psi(p+1, p) = V(1, y), \text{ if } 1 \le p < P \\ \Psi(p-1, p) = V\left(l_1+1, y\right), \text{ if } 2 \le p \le w_{\max} \\ \Psi(p-1, p) = V\left(l_1, y\right), \text{ if } w_{\max} < p \le P \end{cases} \tag{11}$$

Eq. 8 and Eq.11 are used for all above seven different mesh partitioning schemes. After defining the distribution and message passing schemes the next step is to solve the Equation (3) on each worker W and set the error tolerance on local worker and then start the iterations until the solution converges to the predefined error. The numerical solution algorithm is implemented on shared memory systems by writing the user defined codes on MATLAB parallel computing interfaces.

The performance of the parallel numerical solution algorithm is evaluated at different grid sizes and time step where the speedup and efficiency of the algorithms is analyzed. The performance metrics are defined as:

$t_s(sec)$ is the sequential time taken to solve the problem on single worker, $t_p(sec)$ is the parallel time taken to solve the problem on P parallel workers, $t_c(sec)$ is the communication time taken to solve the problem on P parallel workers, $SP = \frac{t_s}{t_p}$ is the parallel speedup achieved on P parallel workers, $EP = \frac{S_p}{P}$ is the parallel efficiency achieved on P parallel workers.

3 RESULTS AND DISCUSSION

Initially, the above Eq. (3) is solved iteratively using FDM by writing a MATLAB code on a single worker of MATLAB. The obtained simulation profiles of the electric potential and electric field look realistic, satisfying

the boundary conditions and shown in the following Figure 3-6. The Figure 3 exhibits the simulation of electric potential in the domain; Figure 4 shows the contour plot of the electric potential with level curves; Figure 5 shows the gradient of V (electric field) with the proportional field line and the Figure 6 reveals the normalized electric field vector plot showing the movement of charges. Theses simulation profiles are quite useful to understand and predict the physical behavior of the micropump. Once the sequential solution of the problem is done the same problem is attempted to solve on more than one processor (workers). The solution with each of the above seven proposed partitioning schemes is obtained with varied mesh size and number of workers. The implementation is done on MATLAB by writing user defined codes for each mesh partitioning scheme and its performance is evaluated.

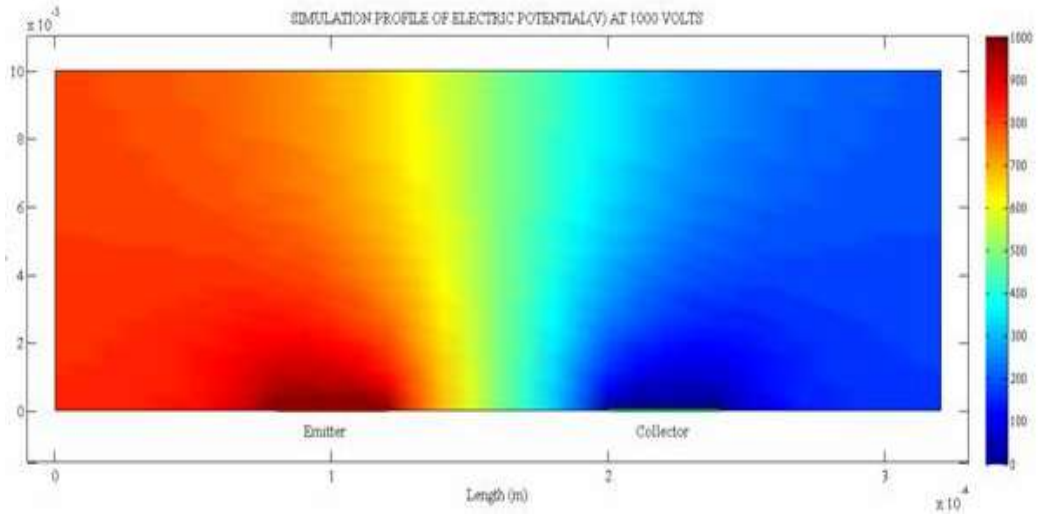

Figure 3. Local distribution of electric potential in the cross section of ion-drag micropump

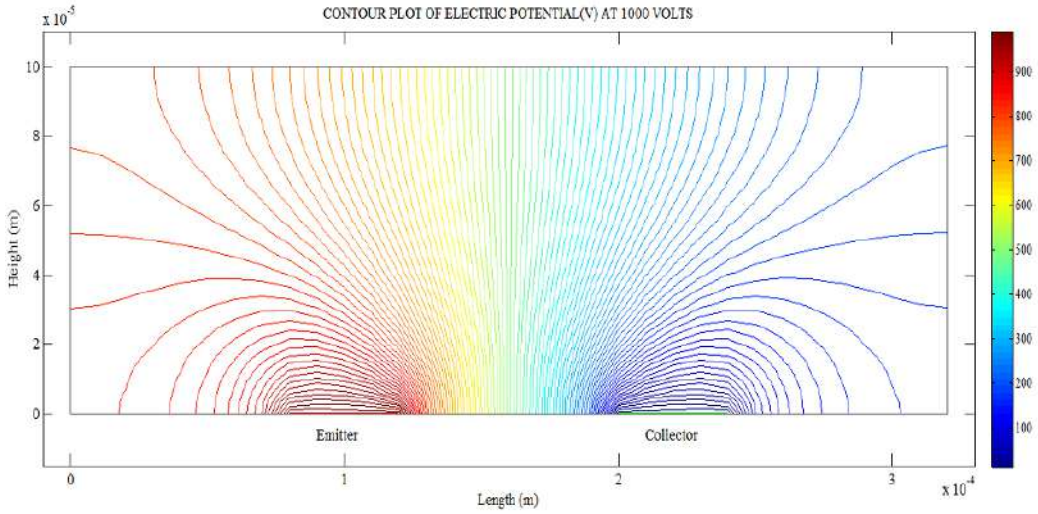

Figure 4. Contour levels of electric potential in the cross section of ion-drag micropump

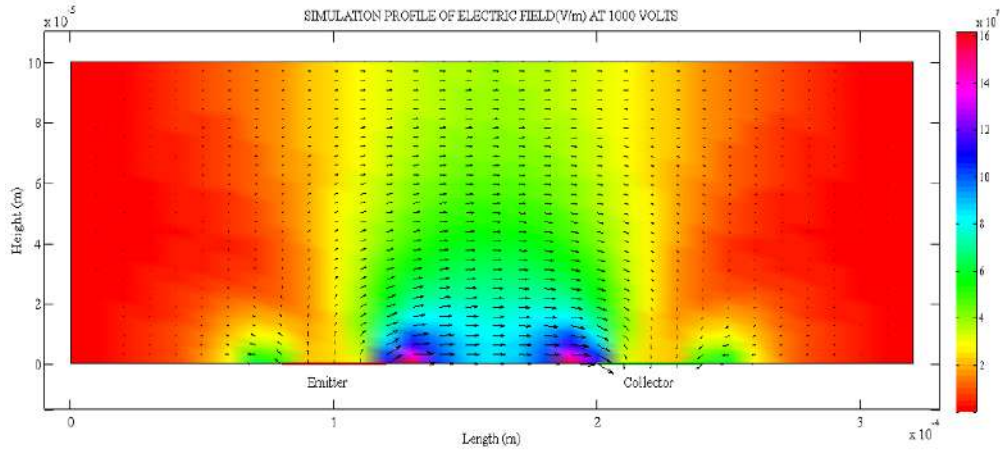

Figure 5. Local electric field (magnitude) distribution in the cross section of ion-drag micropump

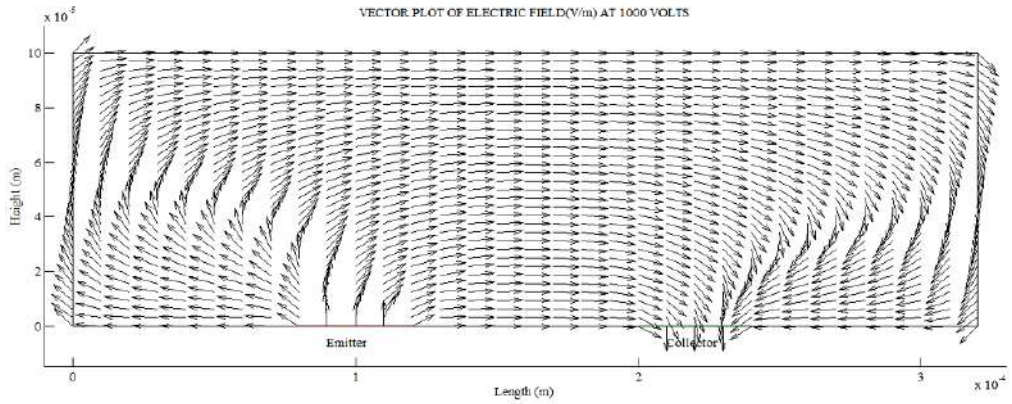

Figure 6. Local electric field (vector lines) distribution in the cross section of ion-drag micropump

The simulation results in above Figures 3-6 have their physical significance for various applications. For instance, the electric potential and electric field under the given space charge density is coupled with the hydrodynamic equations that generate the drag force which is responsible for the pumping of fluid.

To see the reduction in computing time the parallelization of the problem is run on 2, 4, 8, 12 and 16 parallel workers for each mesh partitioning schemes independently. The parallel computing time was noted for each of the seven different mesh partitioning schemes and is show in the Figure 7. It is found that among the all seven partitioning schemes the PS-3 scheme takes less computing time as compared to others. Moreover, the parallel computing time appeared as of exponential order. Similarly, the communication time with respect to different number of workers and partitioning schemes has been analyzed and shown in the following Figure 8. The behavior of communication time is also exponentially increasing as the number of workers increases to share the data among the processors. From the figure, it can be seen that among the all seven partitioning schemes the PS-3 scheme takes less communication computing time as compared to others. Speed up is an important performance metric that is mainly used to see how fast the parallel algorithm works as compared to sequential algorithm. The speed up of the problem with respect to different mesh partitioning schemes is analyzed and is shown in Figure 9. The maximum

speed up is achieved for PS-3. It can be concluded that among all the workers P = 8 workers may be the best choice for large mesh sizes.

Parallel efficiency is also an important performance metric that is mainly used to see how efficiently the parallel algorithm works as compared to sequential algorithm.

The efficiency of the proposed parallel algorithm with respect to different mesh partitioning schemes is analyzed and exhibited in Figure 10. Figure shows that PS-3 remain more efficient up to P = 8 workers and then drops down as the number of workers increases. Thus, it can be deduced that the PS-3 type parallel partitioning may be more feasible as it provides better results. After the PS-3 the PS-6 (two way diagonal partitioning) and PS-7 (window partitioning) schemes are found better as compared to other schemes.

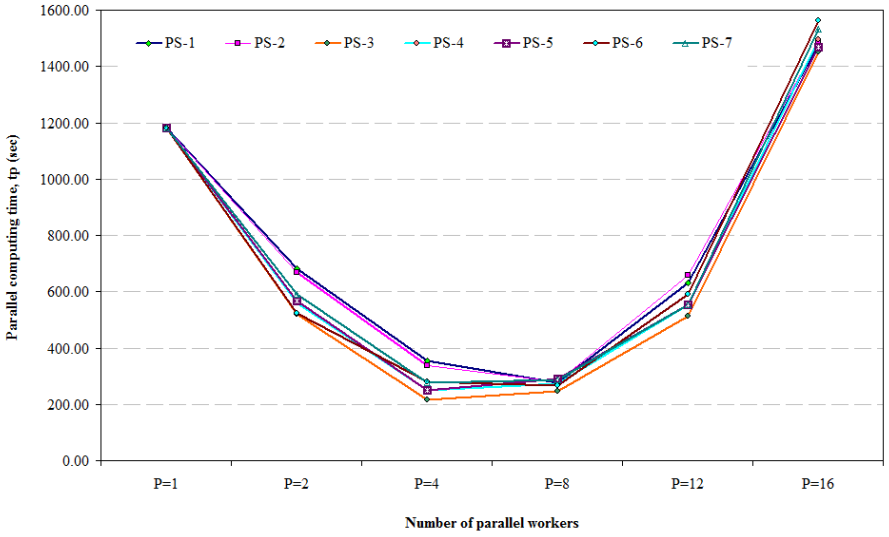

Figure 7. Parallel computing time at different mesh partitioning schemes

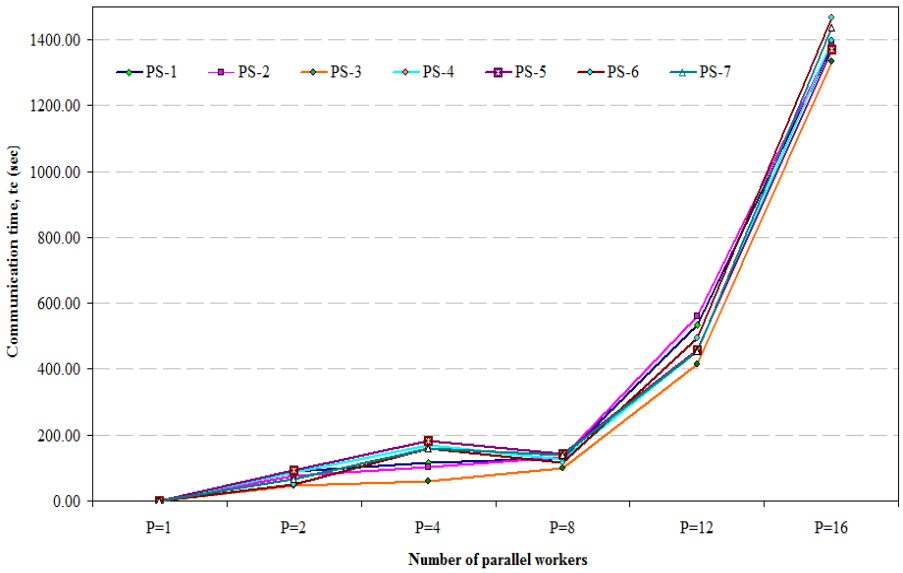

Figure 8. Communication time of different mesh partitioning schemes

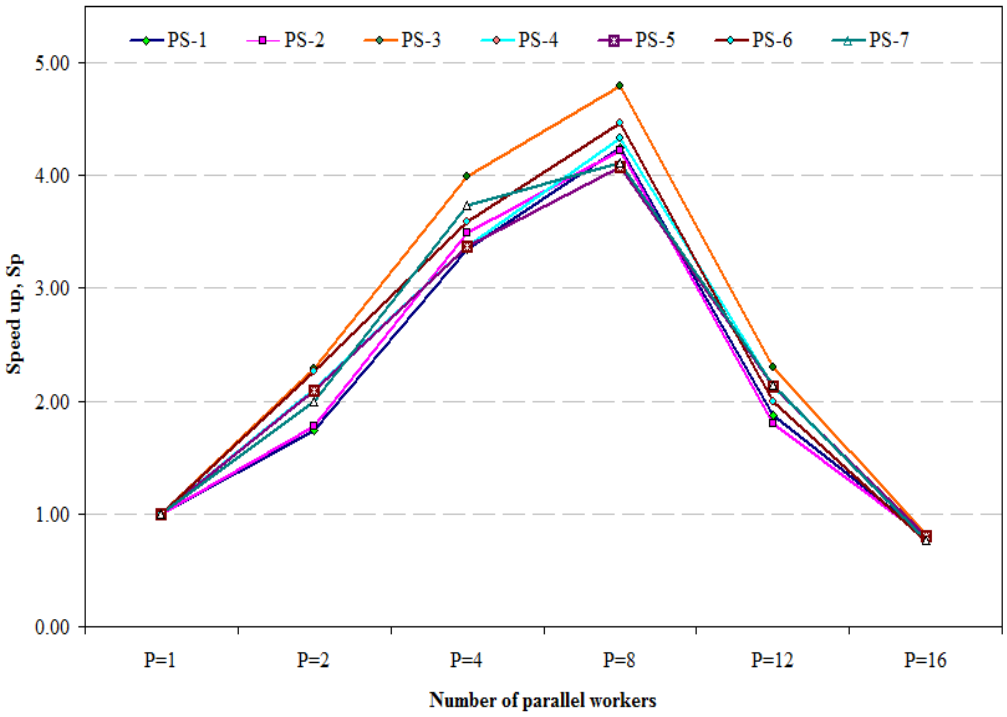

Figure 9. Speedup of different mesh partitioning schemes

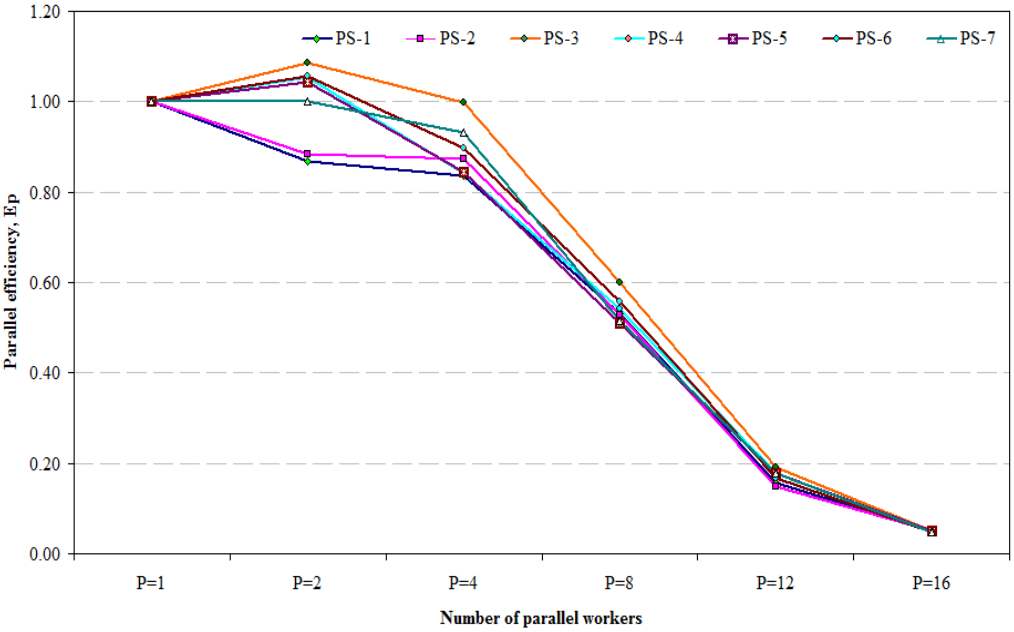

Figure 10. Efficiency of different mesh partitioning schemes

4 CONCLUSION AND FUTURE WORK

The parallel mesh partitioning schemes for the numerical solution of 2D Poisson equation were worked out by using data parallel approach. The seven different types partitioning schemes were implemented on shared memory parallel system using MATLAB. It was revealed that among the seven schemes the PS-3 (two way or tile partitioning) is found the best scheme for the parallel numerical simulation of the problem considered in the study. The parallel algorithm remains more efficient on $P = 2$ to $P = 8$ parallel workers while for $P > 8$ the efficiency drops because of the high communication time. The maximum speed up can be achieved by PS-3 at $P = 8$ workers for 80×80 mesh size. The outcomes of the proposed research provide exciting directions for future research related to the field, for instances; the efficiency may be increased by reducing the communication cost among the parallel workers. The algorithm may provide more speedup if implemented on cluster of computers. The proposed method could be modified for other finite difference schemes used to solve 2D Poisson's equation. The proposed partitioning schemes can be tested for the parallel performance of the other classical partial different equations.

5 Author Contributions

Shakeel Ahmed Kamboh:Mathematical formulation of the problem and Finite difference mesh partitioning schemes, **Faiza Khanam**:Literature review and solution of the Poisson equation, **Nadeem Naeem**:Interpretation of the electrostatics results, **Sajida Parveen**:Suggestions for the parallel computing of the problem , **Sakina Kamboh**: Analyzed data, **Safina Kamboh6**:Editing and proof reading of the graphs.

6 Compliance with Ethical Standards

It is declare that all authors don't have any conflict of interest. It is also declare that this article does not contain any studies with human participants or animals performed by any of the authors. Furthermore, informed consent was obtained from all individual participants included in the study.

7 Funding Information

No Funding

8 Author Information

ORCID:
Shakeel Ahmed Kamboh: 0000-0002-3468-1663
Faiza Khanam: 0009-0001-9769-1201
Nadeem Naeem: 0000-0002-6465-8843
Sajida Parveen: 0009-0005-9760-3370
Sakina Kamboh: 0009-0001-8845-3949
Safina Kamboh: 0009-0000-1020-921X

References

[1] Afzal, A., Ansari, Z. and Ramis, M. [2020], 'Parallelization of numerical conjugate heat transfer analysis in parallel plate channel using openmp', *Arabian Journal for Science and Engineering* **45**(11), 8981–8997.

[2] Bichot, C.-E. and Siarry, P. [2013], *Graph partitioning*, John Wiley & Sons.

[3] Cao, G., Pan, L. and Xu, K. [2022], 'High-order gas-kinetic scheme with parallel computation for direct numerical simulation of turbulent flows', *Journal of Computational Physics* **448**, 110739.

[4] Chakkour, T. [2024], 'Parallel computation to bidimensional heat equation using mpi/cuda and fftw package', *Frontiers in Computer Science* **5**, 1305800.

[5] Devine, K. D., Boman, E. G., Heaphy, R. T., Bisseling, R. H. and Catalyurek, U. V. [2006], Parallel hypergraph partitioning for scientific computing, *in* 'Proceedings 20th IEEE International Parallel & Distributed Processing Symposium', IEEE, pp. 10–pp.

[6] Devine, K. D., Boman, E. G. and Karypis, G. [2006], Partitioning and load balancing for emerging parallel applications and architectures, *in* 'Parallel processing for scientific computing', SIAM, pp. 99–126.

[7] Farhat, C. [1988], 'A simple and efficient automatic fem domain decomposer', *Computers & Structures* **28**(5), 579–602.

[8] Froio, D., Verzeroli, L., Ferrari, R. and Rizzi, E. [2021], 'On the numerical modelization of moving load beam problems by a dedicated parallel computing fem implementation', *Archives of Computational Methods in Engineering* **28**, 2253–2314.

[9] Grandjean, A. and Uçar, B. [2014], On partitioning two dimensional finite difference meshes for distributed memory parallel computers, *in* '2014 22nd Euromicro International Conference on Parallel, Distributed, and Network-Based Processing', IEEE, pp. 9–16.

[10] Hajjar, J. F. and Abel, J. F. [1988], 'Parallel processing for transient nonlinear structural dynamics of three-dimensional framed structures using domain decomposition', *Computers & structures* **30**(6), 1237–1254.

[11] Hendrickson, B. and Leland, R. [1995], 'An improved spectral graph partitioning algorithm for mapping parallel computations', *SIAM Journal on Scientific Computing* **16**(2), 452–469.

[12] HSIEH, S.-H., Paulino, G. H. and Abel, J. F. [1997], 'Evaluation of automatic domain partitioning algorithms for parallel finite element analysis', *International Journal for Numerical Methods in Engineering* **40**(6), 1025–1051.

[13] Hussain, M., Abid, M., Ahmad, M. and Hussain, S. [2013], 'A parallel 2d stabilized finite element method for darcy flow on distributed systems', *World Appl Sci J* **27**(9), 1119–1125.

[14] Kamboh, S. A., Kalhoro, Z. A., Abro, K. A. and Labadin, J. [2017], 'Simulating electrohydrodynamic ion-drag pumping on distributed parallel computing systems', *Indian Journal of Science and Technology* **10**(24), 1–5.

[15] Karypis, G. [2003], Multilevel hypergraph partitioning, *in* 'Multilevel Optimization in VLSICAD', Springer, pp. 125–154.

[16] Karypis, G. and Kumar, V. [1996], Parallel multilevel k-way partitioning scheme for irregular graphs, *in* 'Proceedings of the 1996 ACM/IEEE Conference on Supercomputing', pp. 35–es.

[17] Karypis, G. and Kumar, V. [1998*a*], 'A fast and high quality multilevel scheme for partitioning irregular graphs', *SIAM Journal on scientific Computing* **20**(1), 359–392.

[18] Karypis, G. and Kumar, V. [1998*b*], 'A fast and high quality multilevel scheme for partitioning irregular graphs', *SIAM Journal on scientific Computing* **20**(1), 359–392.

[19] Karypis, G. and Kumar, V. [1998*c*], Multilevel algorithms for multi-constraint graph partitioning, *in* 'SC'98: Proceedings of the 1998 ACM/IEEE Conference on Supercomputing', IEEE, pp. 28–28.

[20] Kumar, V., Grama, A., Gupta, A. and Karypis, G. [1994], *Introduction to parallel computing*, Vol. 110, Benjamin/Cummings Redwood City, CA.

[21] Kumar, V., Karypis, G. and Schloegel, K. [1999], 'A new algorithm for multi-objective graph partitioning'.

[22] Pellegrini, F. [2011], 'Current challenges in parallel graph partitioning', *Comptes Rendus Mécanique* **339**(2-3), 90–95.

[23] Pothen, A., Simon, H. D. and Liou, K.-P. [1990], 'Partitioning sparse matrices with eigenvectors of graphs', *SIAM journal on matrix analysis and applications* **11**(3), 430–452.

[24] Rodriguez, J. S., Parker, R. B., Laird, C. D., Nicholson, B. L., Siirola, J. D. and Bynum, M. L. [2023], 'Scalable parallel nonlinear optimization with pynumero and parapint', *INFORMS Journal on Computing* **35**(2), 509–517.

[25] Sanders, P. and Schulz, C. [2013], Think locally, act globally: Highly balanced graph partitioning, *in* 'International Symposium on Experimental Algorithms', Springer, pp. 164–175.

[26] Warren, M. S. and Salmon, J. K. [1993], A parallel hashed oct-tree n-body algorithm, *in* 'Proceedings of the 1993 ACM/IEEE conference on Supercomputing', pp. 12–21.

[27] Wassim, E., Zheng, B. and Shang, Y. [2024], 'A parallel two-grid method based on finite element approximations for the 2d/3d navier–stokes equations with damping', *Engineering with Computers* **40**(1), 541–554.