Computational time analysis of the numerical solution of 3D electrostatic Poisson’s equation
Shakeel Ahmed Kamboh, Jane Labadin, Andrew Ragai Henri Rigit, Tech Chaw Ling, Khuda Bux Amur, and Muhammad Tayyab Chaudhary

Citation: AIP Conference Proceedings 1660, 070015 (2015); doi: 10.1063/1.4915733
View online: http://dx.doi.org/10.1063/1.4915733
View Table of Contents: http://scitation.aip.org/content/aip/proceeding/aipcp/1660?ver=pdfcov
Published by the AIP Publishing

Articles you may be interested in
Efficient solution of Poisson’s equation with free boundary conditions

Self-consistent numerical solution to Poisson’s equation in an axisymmetric Malmberg-Penning trap

Laplace and Poisson equation solution by relax3d
Comput. Phys. 8, 469 (1994); 10.1063/1.168506

Numerical solution of Poisson’s equation in polyatomic molecules

The Solution of Torsion Problems by Numerical Integration of Poisson's Equation
J. Appl. Phys. 11, 283 (1940); 10.1063/1.1712773
Computational Time Analysis of the Numerical Solution of 3D Electrostatic Poisson’s Equation

Shakeel Ahmed Kamboha, Jane Labadin, Andrew Ragai Henri Rigit, Tech Chaw Ling, Khuda Bux Amur and Muhammad Tayyab Chaudhary

Abstract. 3D Poisson’s equation is solved numerically to simulate the electric potential in a prototype design of electrohydrodynamic (EHD) ion-drag micropump. Finite difference method (FDM) is employed to discretize the governing equation. The system of linear equations resulting from FDM is solved iteratively by using the sequential Jacobi (SJ) and sequential Gauss-Seidel (SGS) methods, simulation results are also compared to examine the difference between the results. The main objective was to analyze the computational time required by both the methods with respect to different grid sizes and parallelize the Jacobi method to reduce the computational time. In common, the SGS method is faster than the SJ method but the data parallelism of Jacobi method may produce good speedup over SGS method. In this study, the feasibility of using parallel Jacobi (PJ) method is attempted in relation to SGS method. MATLAB Parallel/Distributed computing environment is used and a parallel code for SJ method is implemented. It was found that for small grid size the SGS method remains dominant over SJ method and PJ method while for large grid size both the sequential methods may take nearly too much processing time to converge. Yet, the PJ method reduces computational time to some extent for large grid sizes.

Keywords: Computational time; Poisson’s equation, EHD ion-drag micropump; sequential and parallel algorithms.

PACS: 47.65.-d

INTRODUCTION

3D Poisson’s equation is commonly used to model the diffusion of different physical and engineering phenomena such as the gravitation, fluid flow, elasticity, heat conduction, electromagnetism, and many more. In electrostatics; simulating the distribution of electric potential for a given charge distribution described by the charge density function is an important practical problem. A special application of electrostatics Poisson’s equation is the simulation of electric potential in an electrohydrodynamically driven ion-drag micropump. Simulations obtained by solving the Poisson’s equation are useful to understand the behavior of the electric potential and electric field in such devices. Subsequently, provides the better ways to design the devices before their real fabrication.

Different analytical and numerical methods have been used to find the solution of Poisson’s equation. Since most of the physical problems are three dimensional, analytical methods may lead to tedious computations and uses more computer memory [1]. In this case, the numerical iterative methods are preferred and implemented efficiently on computers. Among various numerical methods the finite difference method (FDM) is simple and easy to implement on computers [2]. Poisson's equation can be discretized using FDM; that results into the system of linear equations containing large number of unknowns lying on the mesh of computational domain. These unknowns for corresponding mesh nodes can be approximated iteratively by using stationary (Jacobi, Gauss-Siedel and SOR types) or non-stationary (Conjugate gradient and generalised minimal residual (GMRES) types) iterative methods. A recent review of such methods can be found in [3]. In most of the cases, the discretized Poisson’s equation lead to a system of linear equations with diagonally dominant coefficient matrix for which the stationary methods are more suitable. These methods are simple to implement on sequential computers and they are also used as preconditioners for the non-stationary methods.
Often, SGS converges to true solution in less number of iterations than SJ. On the other hand, PJ iteration may be better in speed even with more iterations, since its algorithm is inherently parallel thus, can exploit the advantage of simultaneous parallel computation [4]. However, such assumptions may not applicable to all kinds of problems with different data sizes because there are many cases where SJ fails to converge but SGS converges [5]. The objective of this work is to analyze the computational time required by both SJ and SGS methods with respect to different grid sizes and the speedup of PJ method over standard SGS method. It will be investigated that at what extent the PJ method is preferable than SGS for a given data size. To achieve this goal, first the discretized Poisson’s equation is solved by SJ and SGS methods implemented on MATLAB. Next, Jacobi method is parallelized using parallel domain partitioning technique and implemented on MATLAB parallel computing environment based on a single program multiple data (SPMD) model. At the end, the processing time for each method is given and analyzed.

The paper begins with the Introduction Section highlighting the problem and main objective of this study. Next the finite difference discretization of the 3D Poisson’s equation and the implementation of its solution by SJ and SGS methods, and the PJ method are presented in the Numerical Solution Methodology Section. The numerical simulation results and the analysis of the computational time are discussed in the Section of Results and Discussion. Finally, the Conclusion Section summarizes briefly the findings of this paper and gives directions for future work.

NUMERICAL SOLUTION METHODOLOGY

The Poisson’s equation is an elliptic type partial differential equation and is one of the basic governing equations of Electrohydrodynamics [6]. Finding the electric potential and the electric field distributions is an important practical problem in EHD ion-drag pumping. The general form of 3D electrostatics Poisson’s equation in Cartesian coordinates is given as follows,

$$\frac{\partial^2 V}{\partial x^2} + \frac{\partial^2 V}{\partial y^2} + \frac{\partial^2 V}{\partial z^2} = -\frac{q_e}{\varepsilon},$$  \hspace{1cm} (1)

where $V$ is the electric potential (Volts), $q_e$ the space charge density (C/m$^3$) and $\varepsilon$ the relative permittivity (F/m), also both $V$ and $q_e$ are the functions of space coordinates $(x, y, z)$. This equation describes the steady-state electric potential in any 3D media with the boundaries where external electric potential is applied on the electrodes. In order to obtain the numerical solution, Eq. (1) is discretized using central finite difference scheme and is given by,

$$\frac{V_{i+1,j,k} - 2V_{i,j,k} + V_{i-1,j,k}}{h_1^2} + \frac{V_{i,j+1,k} - 2V_{i,j,k} + V_{i,j-1,k}}{h_2^2} + \frac{V_{i,j,k+1} - 2V_{i,j,k} + V_{i,j,k-1}}{h_3^2} = -\frac{q_{e,i,j,k}}{\varepsilon},$$  \hspace{1cm} (2)

where $i$, $j$ and $k$ represent the nodes location in the discretized mesh while $h_1$, $h_2$ and $h_3$ are the step sizes (increments) in $x$, $y$ and $z$ directions respectively. If there are $l$, $m$, $n$ discrete cells along $x$, $y$ and $z$ axis respectively then there will be $l+1$, $m+1$ and $n+1$ grid points along respective axis. The charge density values at each node are predefined constants and their description can be found in [7]. Note that Eq. (2) represents a system of linear equations in unknowns $V$, and could be manipulated to express in the standard matrix form $AV = B$. However, its worth to compute all unknowns recursively without operating and storing the coefficient matrix $A$. Therefore, Eq (2) is expressed explicitly for $V_{i,j,k}$ and given as follows:

$$V_{i,j,k} = \frac{1}{2\varepsilon\left(h_1^2 + h_2^2\right)h_3^2 + h_1^4h_2^2 + h_1^2h_2^4}\left(\frac{q_{e,i,j,k}h_1^2}{h_1^2 + h_2^2} + \left(V_{i+1,j,k} + V_{i-1,j,k}\right)h_1^2 + \left(V_{i,j+1,k} + V_{i,j-1,k}\right)h_2^2\right).$$  \hspace{1cm} (3)

Equation (3) is applied to a prototype of EHD ion-drag micropump whose basic configurations labeled with both Dirichlet and Neumann boundary conditions are shown in Figure 1. It is attempted to solve this equation numerically by two classical iterative methods; Jacobi and Gauss-Seidel methods. If $r$ represents any iteration to run the algorithms of Jacobi and SGS methods then Eq. (3) can be expressed by Eq. (4) and Eq. (5) respectively.

\begin{align*}
\text{Eq. (4):} & \quad V_{i,j,k}^{(r)} = \frac{1}{2\varepsilon\left(h_1^2 + h_2^2\right)h_3^2 + h_1^4h_2^2 + h_1^2h_2^4}\left(\frac{q_{e,i,j,k}h_1^2}{h_1^2 + h_2^2} + \left(V_{i+1,j,k}^{(r-1)} + V_{i-1,j,k}^{(r-1)}\right)h_1^2 + \left(V_{i,j+1,k}^{(r-1)} + V_{i,j-1,k}^{(r-1)}\right)h_2^2\right), \\
\text{Eq. (5):} & \quad V_{i,j,k}^{(r)} = \frac{1}{2\varepsilon\left(h_1^2 + h_2^2\right)h_3^2 + h_1^4h_2^2 + h_1^2h_2^4}\left(\frac{q_{e,i,j,k}h_1^2}{h_1^2 + h_2^2} + \left(V_{i+1,j,k}^{(r)} + V_{i-1,j,k}^{(r)}\right)h_1^2 + \left(V_{i,j+1,k}^{(r)} + V_{i,j-1,k}^{(r)}\right)h_2^2\right).
\end{align*}
In addition to the SJ and SGS methods, the Jacobi method is parallelized using data parallelization to achieve the speedup. Since Jacobi method can be parallelized perfectly as all the results of each iteration depend on the results of previous iteration. Whereas, the SGS method is intrinsically sequential and is quite difficult to parallelize as compared to Jacobi method. The domain decomposition method [8-9] is applied on Eq. (4) to parallelize the Jacobi method and the basic steps are listed as follows:

Step: 0 initialize the parallel system and computational domain with all necessary information.

Step: 1 partition the computational domain \( D \) into sub-domains \( D_1, D_2, \ldots, D_p \) and assign each domain to individual processor \( P_1, P_2, \ldots, P_p \) respectively. It is appropriate to partition the domain along the axis having more discrete cells.

Step: 2 start process with iteration \( r = 0 \) to \( r = I \), and predefined error tolerance, where \( I \) refers to the maximum number of iterations required to converge the solution.

Step: 3 share the neighborhood slices between the adjacent sub-domains and update the data on boundaries as there are data dependencies in Eq. (4). This step requires necessary message passing among the processors.

Step: 4 solve Eq. (4) locally on each processor \( P_1, P_2, \ldots, P_p \) over distributed domain \( D_1, D_2, \ldots, D_p \) respectively.

Step: 5 check for convergence locally on each individual processor.

Step: 6 check for convergence globally on all processors. If the solution is converged then stop and gather the all local solutions from all processors. Otherwise, repeat the Steps 2-6.

FIGURE 1. Computational domain \( D \), a segment of ion-drag micropump with basic configurations and boundary conditions.
Implementation and Testing of the Sequential and Parallel Numerical Algorithms

This section evaluates the implementation and testing of the above numerical methods for the analysis of computational time. The sequential and parallel numerical solution algorithms are implemented on MATLAB with the same computing resources used by [6]. Both the sequential and parallel algorithms are tested for different grid sizes \( l \times m \times n \) as given in Table (1); increasing the grid size decreases the cell size consequently refines the mesh of computational domain. First, the SJ and SGS methods are run for each grid size with predefined error tolerance of 0.001. Sequential computing \( t_s \) (sec) and the maximum number of iterations \( I_r \) required to converge are listed in Table (1). After that, the parallel system is initialized for cluster of 2, 4, and 8 processors \( p \) and the parallel version of Jacobi method is run and tested for the same data sizes as used in sequential implementation. The numerical results obtained on parallel systems were found correct and converging in the same number of iterations as in sequential. Parallel computing time \( t_p \) for each of the data sizes is noted and given in the TABLE (1) below:

<table>
<thead>
<tr>
<th>Cell size</th>
<th>Grid size</th>
<th>Sequential Jacobi (SJ)</th>
<th>Sequential Gauss-Seidel (SGS)</th>
<th>Parallel Jacobi (PJ) (Parallel time, ( t_p ))</th>
</tr>
</thead>
<tbody>
<tr>
<td>( h_1 )</td>
<td>( h_2 )</td>
<td>( h_3 )</td>
<td>( l )</td>
<td>( m )</td>
</tr>
<tr>
<td>20</td>
<td>13.3</td>
<td>10</td>
<td>16</td>
<td>20</td>
</tr>
<tr>
<td>13.3</td>
<td>10.0</td>
<td>10</td>
<td>8</td>
<td>122.27</td>
</tr>
<tr>
<td>6.67</td>
<td>4.5</td>
<td>4</td>
<td>5</td>
<td>28555.68</td>
</tr>
<tr>
<td>3.33</td>
<td>2.2</td>
<td>2</td>
<td>5</td>
<td>67156.17</td>
</tr>
<tr>
<td>2.5</td>
<td>1.5</td>
<td>1</td>
<td>6</td>
<td>107178.92</td>
</tr>
</tbody>
</table>

### RESULTS AND DISCUSSION

In this section, the numerical solution of the 3D Poisson’s equation and computing time taken by the SJ, SGS and PJ methods are presented and analyzed. Before comparing and analyzing the computational time its worth to check the numerical feasibility of applying both the Jacobi and SGS methods. For a typical grid size FIGURE 2 and FIGURE 3 shows the numerical simulation of electric potential in EHD ion-drag micropump obtained by using the SJ and SGS methods. The simulation patterns are significantly same verifying the correctness of both the methods. To check the difference between the numerical solutions of both the methods the results are compared at each grid point and shown in FIGURE 4. A good agreement in the numerical results can be seen except for minor differences at some data points.

**FIGURE 2.** Numerical simulation of electric potential by Jacobi method (grid size 32x40x10).

**FIGURE 3.** Numerical simulation of electric potential by SGS method (grid size 32x40x10).
The sequential computing time, $t_s$, and parallel computing time $t_p$, as given in Table (1) is represented graphically in Figure 4. It is obvious from the results that the PJ can be preferred for sufficiently large grid sizes. In view of the fact, that the results for small grid sizes are not understandable from Figure 4. Therefore, a more clear depiction is represented by 100% stack column bar diagram. The fraction (percentage) of computing time for each grid size is depicted. It is revealed that for small grid size the SGS method is dominant over both SJ and PJ. But as for as the grid size increases the PJ method appears better than both sequential versions of Jacobi and Gauss-Seidel methods. Also, in case of small grid sizes the PJ may perform better by utilizing few processors, say $p=2$. In order to test the speedup of PJ over SJ and SGS methods their sequential time is divided by the parallel time. The speedup of PJ over SJ and SGS methods are exhibited by Figure 7 and Figure 8 respectively. The speedup of PJ over SJ is not fine for small grid sizes and trends show that increasing the number of processor decreases the speedup. It is because of the communication time is greater than the computational time. Whereas, for the large data sizes the speed up increases by some extend with respect to the number of processors. Similarly, the speedup of PJ over SGS is not satisfactorily unless the data size is very large. It is revealed that PJ provides more speedup over SJ than SGS. From the analysis of the computing time and speedup it can be concluded that for small data size the SGS is better than SJ and PJ but for sufficiently large data sizes the PJ could be preferred than SJ and SGS.
The analysis of computational time taken by SJ, SGS and PJ methods was presented for the numerical solution of 3D Poisson’s equation. The results of the present study show that the computing time for solving such problems depends on the number of grid points (data sizes) and the maximum number of iterations. SJ method is slower than SGS method since SGS converges in less number of iterations than SJ. Yet, in case of large number of grid points such as 128x160x40, both the methods may take high computing time regardless of number of iterations. It is revealed that for small data size the SGS is better than SJ and PJ but for sufficiently large data sizes the PJ reduces computing time to some extent and could be preferred than SJ and SGS. Also, PJ method provides more speedup over SJ than that of SGS. The numerical solution by PJ method could be even much faster if the communication cost is not a matter. Unfortunately, with the recent advances in technology the processing speed of computers is increasing progressively but the communication speed is not increasing significantly as compared to the speed of processors. Future work will focus on the reduction of communication time by handling the dependencies patterns in PJ method.
ACKNOWLEDGMENTS

The authors gratefully acknowledge financial support from the University of Malaya and collaboration for this work with Universiti Malaysia Sarawak. The authors would also like to thank both the universities for providing full access to the resources used in the conduct of this study.

REFERENCES