Benchmarks of 3D Laplace Equation Solvers in a Cubic Configuration for Streamer Simulation

Joseph-Marie PLEWA¹, Olivier DUCASSE¹, Philippe DESSANTE², Carolyn JACOBS², Olivier EICHWALD¹, Nicolas RENON³, Mohammed YOUSFI¹, ⁴

¹Université de Toulouse, UPS, INPT, LAPLACE (Laboratoire Plasma et Conversion d’Energie) Toulouse Cedex 9 F-31062, France
²Group of Electrical Engineering-Paris (GeePs), UMR 8507 CNRS, CentraleSupelec, Universite Paris-Sud—UPMC, 11 rue Joliot-Curie, Plateau de Moulon, 91192 Gif-sur-Yvette Cedex F-91192, France
³CALMIP, Université de Toulouse-CNRS-INPT-INSA-UPS, UMS 3667, France
⁴CNRS, LAPLACE, Toulouse F-31062, France

Abstract The aim of this paper is to test a developed SOR R&B method using the Chebyshev accelerator algorithm to solve the Laplace equation in a cubic 3D configuration. Comparisons are made in terms of precision and computing time with other elliptic equation solvers proposed in the open source LIS library. The first results, obtained by using a single core on a HPC, show that the developed SOR R&B method is efficient when the spectral radius needed for the Chebyshev acceleration is carefully pre-estimated. Preliminary results obtained with a parallelized code using the MPI library are also discussed when the calculation is distributed over one hundred cores.

Keywords: numerical methods for elliptic equations, high performance computing, 3D streamer simulation, SOR, IDR, BiCGSTAB

PACS: 52.65.−y, 52.80.Hc

DOI: 10.1088/1009-0630/18/5/16

(Some figures may appear in colour only in the online journal)

1 Introduction

Generally, when a short high-pulsed voltage is applied between asymmetric electrodes, the resulting breakdown corona discharge regime is characterized by the propagation of several streamers presenting a large branching structure [1]. However, for a DC high voltage condition, the branching structure rapidly disappears and a stabilized monofilament discharge occurs, with the successive development of a primary and a secondary streamer [2]. This DC micro-discharge can be simulated with a 2D(r, z) cylindrical grid domain that reproduces the symmetry of the monofilament [3,4]. For a pulse voltage condition, a complete 3D simulation is required in order to capture the natural tree structure of the discharge [5]. In the last decades, increasing computer power has permitted the 3D simulation of micro-discharges using classical first order fluid models and/or hybrid calculations [6–9].

The 3D simulation is very time consuming because in the discharge model both hyperbolic and elliptic equations must be solved at each time step of the streamer evolution. Special attention should be paid to the efficiency of the elliptic equation solvers in terms of both precision and simulation time because their contribution on the total discharge computation time can be greater than 90%.

The present work is devoted to the comparison of a specific 3D solver developed in our group [10] with three other pre-selected solvers proposed in the open source LIS library. The simulation conditions and a brief description of the solvers are given in section 2. In section 3, the efficiency of each solver is first tested with a single core configuration and a preliminary result is given on a multi-cores configuration using the MPI library of the High Performance Computer EOS [11]. EOS is a BULLx DLC system composed of 612 computing nodes, each equipped with two Intel IvyBridge processors at 2.8 GHz, totaling 12240 computing cores latest generation and 39.1 TB of memory. One processor has 10 cores and 25 MB memory cache. The peak power of this supercomputer is 274 TFLOPS for a power consumption of 244 kW at the LINPACK test.

2 Simulation conditions and solver description

The elliptic solvers are tested to calculate the electrical potential, solution of the classical Laplace equation.
The 3D domain is a cube of with an edge length of 1 millimeter divided into \( n^3 \) equidistant discretization points, \( n \) being equal to 50, 100 or 200 for different calculations. Dirichlet boundary conditions are applied on the lower and upper electrodes (0 kV and 6 kV respectively) while zero Neumann conditions are applied on the others lateral surfaces.

After the discretization on the 3D Cartesian domain, the elliptic Laplace equation can be expressed in the form of a linear matrix system:

\[
Ax = b, \tag{1}
\]

where \( A \) is a sparse matrix (of \( n^3 \times n^3 \) dimension), \( x \) the potential vector and \( b \) a specific vector involving the potential boundary conditions. \( x \) and \( b \) are of length \( n \).

In order to solve this linear matrix system, we use in the present paper iterative solvers. Iterative methods can be very efficient, especially for time dependent problems where the solution at a previous time \( t \) can be used as an initial estimation for the new iteration process, leading to the solution at time \( t + \Delta t \). In this work, we compare the efficiency of three methods selected from the open source LIS library [12] (IDR(s), BiCGSTAB(l) and SOR(\( \omega \))) with a 3D solver developed in our group. This solver is based on the classical Successive Over Relaxation Method (SOR(\( \omega \))) using the Red and Black (R&B) technique and the Chebyshev accelerator algorithm [13].

In order to introduce the principal elements of the SOR(\( \omega \)) method, we consider a decomposition of matrix \( A \) as

\[
A = L + D + U, \tag{2}
\]

where \( D \) is the diagonal part of \( A \) and \( L \) and \( U \) the lower and upper triangles of \( A \) (with zero on the diagonal). The classical iterative Gauss-Seidel method can then be written in the form:

\[
x^{k+1} = x^k - (L + D)^{-1}r^k, \tag{3}
\]

where \( x^k \) is an approximation of solution \( x \) at iteration \( k \) and \( r^k \) the residual vector with:

\[
x^k = Ax^k - b. \tag{4}
\]

The SOR(\( \omega \)) method extends upon Eq. (3) by adding the over relaxation parameter \( \omega \in [0, 2] \) which increases the convergence speed of approximation \( x^{(k+1)} \) towards \( x \). The iteration step is therefore rewritten as:

\[
x^{k+1} = x^k - \omega(L + D)^{-1}r^k. \tag{5}
\]

Of course, the crucial part of the SOR(\( \omega \)) algorithm is the judicious choice of the \( \omega \) value. Practically, when the SOR(\( \omega \)) is developed under Red and Black ordering (as in a 3D checkerboard), the optimal \( \omega \) value can be calculated from the Chebyshev acceleration method using the spectral radius \( \rho \) [13] (i.e. the largest absolute eigenvalue of the matrix \( M = -(L + U)D^{-1} \)). In this case, the difficulty leads in the calculation of this maximum absolute eigenvalue of a large matrix \( M \) of dimension \( n^3 \times n^3 \) (\( n \) could be on the order of several hundred points). At this stage, it is worth noticing that the spectral radius depends only on the coefficients of matrix \( M \) and, as a consequence, only on the coefficients of matrix \( A \). Therefore, for a given matrix \( A \) and vector \( b \) on the right hand side of Eq. (1), initial tests of the convergence speed can be undertaken in order to find, by successive iterations, the value of the spectral radius or the over relaxation parameter with or without the inclusion of the Chebyshev acceleration. In the present paper, two SOR algorithms were tested: a home developed SOR R&B using the spectral radius to initiate the Chebyshev acceleration and the SOR(\( \omega \)) method proposed in the LIS library. As will be shown in section 3, the convergence speed of the SOR R&B solver is very sensitive to the initial value of the spectral radius \( \rho \).

The BiCGSTAB(l) and IDR(s) algorithms are part of the family of general recursive Krylov type solvers. \( x^k \) and \( r^k \) are estimated at iteration \( k \) from a recursive function estimated through specific linear combinations of vectors \( b, Q_1b, Q_2b, \ldots, Q_{k-1}b \) where \( Q \) could be equal to \( A, A^T, P^{-1}A, \ldots \) (\( P \) being a preconditioned matrix like \( L + D \) in the Gauss-Seidel method). The integer parameters \( l \) and \( s \) are used to estimate the optimal linear combinations (e.g. in BiCGSTAB(l)) algorithms. \( l \) stands for the degree of minimal residual polynomials involved in the calculation of \( r^k \). Detailed information can be found in Refs. [14] and [15] for the BiCGSTAB(l) and IDR(s) algorithms respectively. As \( l \) and \( s \) are integer parameters that must be fixed, initial tests were undertaken in order to determine the best values for increasing the convergence speed at our specific simulation conditions.

For all tested iterative solvers, the convergence criterion is defined as follows:

\[
\frac{\| r_k \|}{\| b \|} < \varepsilon, \tag{6}
\]

where \( \| \cdot \| \) stands for the norm 2 of the corresponding vectors. \( \varepsilon \) is the chosen convergence precision, varying from \( 10^{-8} \) to \( 10^{-6} \). Furthermore, (i) the maximum number of iterations was limited to 9000 which means that the solver failed when the solution has not converged towards the chosen precision after a maximum of 9000 iterations and (ii) the initial potential values \( x^{k=0} \) used to initiate the iteration processes were set to 3 kV. Fig. 1 displays a simplified flowchart summarizing the successive steps involved in the solution of the electrical potential in our domain of study.

### 3 Results and analysis

#### 3.1 Optimal parameters determination for the LIS solvers

As previously explained, the efficiency of the three chosen methods from the LIS library can be enhanced
if the optimal values of parameters $s$, $l$ or $\omega$ are pre-estimated. Therefore, preliminary tests were undertaken in order to choose the best values permitting the optimization of the numerical convergence of each solver for our specific simulation conditions. Table 1 summarizes the resulting optimal values which enhance the efficiency of the methods in terms of precision and computing time.

![Fig.1 Successive steps to solve the electrical potential from Laplace equation](image)

**Table 1.** Optimal parameters obtained at our simulation conditions for the three chosen methods selected from the LIS library

<table>
<thead>
<tr>
<th>Number of points</th>
<th>Convergence precision $\varepsilon$</th>
<th>BIcgSTAB parameter $l$</th>
<th>IDR parameter $s$</th>
<th>SOR parameter $\omega$</th>
</tr>
</thead>
<tbody>
<tr>
<td>50</td>
<td>$10^{-6}$</td>
<td>2</td>
<td>3</td>
<td>1.93</td>
</tr>
<tr>
<td></td>
<td>$10^{-7}$</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>$10^{-8}$</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>100</td>
<td>$10^{-6}$</td>
<td>2</td>
<td>3</td>
<td>1.96</td>
</tr>
<tr>
<td></td>
<td>$10^{-7}$</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>$10^{-8}$</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>200</td>
<td>$10^{-6}$</td>
<td>2</td>
<td>3</td>
<td>1.98</td>
</tr>
<tr>
<td></td>
<td>$10^{-7}$</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>$10^{-8}$</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

It may be noted that the optimal values of parameters $l$ (for BIcgSTAB($l$) algorithm) and $s$ (for IDR($s$) algorithm) do not depend on the number of discretization points or the convergence precision, while the optimal value of parameter $\omega$ (for SOR($\omega$) algorithm) clearly depends on the number of discretization points.

Plasma Science and Technology, Vol.18, No.5, May 2016

It should also be noted that in this last case, a non-optimal value of $\omega$ can drastically increase the computing time or lead to a calculation, which does not converge before the maximum number of iterations is reached (9000). This phenomenon is also observed for our developed SOR R&B algorithm. As a consequence, a detailed analysis of the SOR R&B convergence speed as a function of the initial value of the spectral radius $\rho$ was performed and is presented in the next subsection.

### 3.2 Sensitivity study of the SOR R&B convergence speed to the spectral radius value

The optimum value of the spectral radius $\rho$ induces the fastest convergence for a given convergence precision, which also means the lowest computing time. In order to optimize the convergence speed, we analyze the number of iterations required for the SOR R&B to converge as a function of three parameters: the convergence precision $\varepsilon$, the number of discretization points $n^3$, and the value of the spectral radius $\rho$ used to initiate the Chebyshev accelerator algorithm. Figs. 2, 3, and 4 show the results obtained for $n=50$, 100, and 200, $\varepsilon=10^{-6}$, $10^{-7}$, and $10^{-8}$, and $\rho$, varying from 0.97 up to 1. Following these preliminary tests, the optimum $\rho$ values obtained in our simulation conditions are listed in Table 2.

![Fig.2 Spectral radius diagnostic for $n=50$ points](image)

For all studied combinations of the number of discretization points and the accuracy, the results show that the convergence speed is very sensitive to the magnitude of the injected spectral radius. In our simulation conditions, the convergence speedup increases considerably when the spectral radius tends towards 1. However, the number of iterations needed to converge increases considerably when the spectral radius exceeds the optimum value given in Table 2. One can also notice that as expected [13], the higher the number of discretization points, the closer $\rho$ is to one. For $n=50$, the optimal $\rho$ value depends on the chosen precision, while this is no longer the case for $n=100$ and 200. The tests were repeated three times and the same results were
obtained each time, indicating that it is not a random phenomenon.

Table 2. Optimal spectral radius for a fixed mesh and convergence precision

<table>
<thead>
<tr>
<th>Number of points</th>
<th>Convergence precision</th>
<th>Iteration number</th>
<th>Optimal spectral radius</th>
</tr>
</thead>
<tbody>
<tr>
<td>n</td>
<td>ε</td>
<td></td>
<td></td>
</tr>
<tr>
<td>50</td>
<td>10^{-6}</td>
<td>125</td>
<td>0.99809999</td>
</tr>
<tr>
<td></td>
<td>10^{-7}</td>
<td>145</td>
<td>0.99810999</td>
</tr>
<tr>
<td></td>
<td>10^{-8}</td>
<td>203</td>
<td>0.99867999</td>
</tr>
<tr>
<td>100</td>
<td>10^{-6}</td>
<td>245</td>
<td>0.99949999</td>
</tr>
<tr>
<td></td>
<td>10^{-7}</td>
<td>281</td>
<td>0.99949999</td>
</tr>
<tr>
<td></td>
<td>10^{-8}</td>
<td>322</td>
<td></td>
</tr>
<tr>
<td>200</td>
<td>10^{-6}</td>
<td>522</td>
<td>0.99988909</td>
</tr>
<tr>
<td></td>
<td>10^{-7}</td>
<td>603</td>
<td></td>
</tr>
<tr>
<td></td>
<td>10^{-8}</td>
<td>691</td>
<td></td>
</tr>
</tbody>
</table>

These preliminary tests clearly indicate that a careful diagnostic must be undertaken in order to find the optimal ρ value for a given configuration in order to exploit the SOR R&B algorithm under the best conditions. Nevertheless, once determined for the given simulation conditions (i.e. spatial mesh repartition, number of discretization points and boundary conditions), the pre-determined optimal value of ρ can be used for all the values of the right hand side b vector in Eq. (1).

This is important because in the case of streamer simulations, the b vector is modified at each time step by taking into account the space charge evolution.

3.3 Algorithm efficiency comparisons to solve the Laplace equation using a single core computing

Figs. 5, 6 and 7 show the efficiency of the four selected solvers in terms of computing time versus precision for n=50, 100, and 200 respectively. The computing time required to converge increases when the precision decreases and the number of points increases. The results indicate that the general trends of BiCGSTAB(2) and SOR R&B are very similar but BiCGSTAB(2) converges slightly faster for greater precision. IDR(3) remains competitive only for n=50, but the number of iterations needed to converge is remarkably stable whatever the chosen precision.

It is interesting to note that a gain of 100 in the precision (i.e. by reducing the precision from 10^{-6} to 10^{-8}) produces a very small increase of computing time, especially for the SOR R&B method and BiCGSTAB(l). For example, in the case of n=200 in Fig. 7, the SOR R&B method requires only 1.3 times more computing time to converge from ε=10^{-6} to 10^{-8}. For this same method, with ε=10^{-8}, it can be seen that multiplying the number of cells by height increases the computing time by a factor of about 16. Indeed, from 50^3 to 100^3 (≈8×50^3) and from 100^3 to 200^3 (≈8×100^3), the computing time for SOR R&B increases from 0.28 s to 4.6 s (≈16.4×0.28 s) and from 4.6 s to 75 s (≈16.3×4.6 s).

In all tested situations, the BiCGSTAB(l) algorithm proposed in the LIS library remains the more efficient (except for n=200 and ε=10^{-6}, closely followed by the developed SOR R&B algorithm.)
3.4 Preliminary tests of parallel SOR R&B implementation using MPI library

While essential, studies using a single core are insufficient. The only way to reduce the computing time when the number of points increases is through the use of parallelization techniques in High Performance Computing (HPC). In this situation, in addition to the intrinsic effectiveness of the various algorithms for a single core, it is also important for their ability to be efficiently allocated on multi-cores without losing their performance. Following our previous tests discussed in section 3.3, we focus our first study on the resolution of potential by a parallelized SOR R&B method using the MPI library for multiple cores management. In our case, the initial cube domain is divided into slices along the vertical axis. Each slice, having the same dimension, is allocated to a core and the potential values at the borders of slices are communicated between cores. Initial tests have been undertaken for \( n = 200 \) with one hundred cores, which means that each slice is \( 200^2 \times 2 \) in size. The gain of calculation time resulting from the parallelized SOR R&B method can be seen in Fig. 8.

This first result shows that in moving from a single core to one hundred cores, the mean calculation time (for all precisions) is divided by a factor of approximately 18. For example, with \( \varepsilon = 10^{-8} \), the single core program requires 75 s whereas only 4 s are needed when one hundred cores are used. Theoretically, i.e., using Amdahl’s law with a fully parallelized algorithm, multiplying the number of cores by a factor of one hundred should induce a time saving of the same proportion until the time of communication exchange between cores can be neglected. Our first results suggest that the communication exchange time cannot be neglected. In order to improve the speedup, the parallelized MPI program could be coupled with OpenMP techniques. This hybridization can be very efficient depending on the HPC architecture.

4 Conclusion and perspectives

The comparison between several iterative solvers for elliptic equations indicated that the SOR R&B method using the Chebyshev accelerator algorithm and the BiCGSTAB(2) method demonstrate the best efficiency in terms of precision and computing time for the solution of the Laplace equation in tested configurations. However, in order to have a reduced computing time with the SOR R&B method, an estimation of the optimal spectral radius value is essential. Moreover, and as expected, the use of MPI parallel computing reduces the calculation time. Parallel computing is the best way to solve domains with a significant number of nodal points. In the studied configuration, the SOR R&B method can be improved by coupling MPI programming with OpenMP, while accounting for the HPC architecture. Future work will be focused on the influence of mesh size and on the distribution domain over a larger number of cores to see the effects of these parameters on the computing time for several iterative solvers (mainly BiCGSTAB(2) and SOR R&B) and also a non-iterative solver like MUMPS for example. The ef-
ficiency of the methods selected will also be evaluated for the case where a specific space charge density crosses the cubic domain. In this case, the Poisson equation replaces the Laplace equation in the calculations.

Acknowledgments

This work was performed using HPC resources from CALMIP (Grant 2011-[P1053]) [11] and is supported by the French Agence Nationale de la Recherche under Project REMOVAL ANR-12-BS09-0019-1.

References


E-mail address of corresponding author
DUCASSE Olivier: ducasse@laplace.univ-tlse.fr