The 8th European Congress on Computational Methods in Applied Sciences and Engineering
ECCOMAS Congress 2022
5-–9 June 2022, Oslo, Norway

PARALLEL FINITE VOLUME CODE FOR PLASMA WITH UNSTRUCTURED ADAPTIVE MESH REFINEMENT

I. Kissami[1], S. Maazioui[1,2,3] F. Benkhaldoun[3]

[1] Mohammed VI Polytechnic University, Lot 990 Hay Moulay Rachid, Benguerir 43150, Morocco. e-mail: imad.kissami@um6p.ma and souhail.maazioui@um6p.ma

[2] Civil engineering Department, Ecole Mohammadia d'Ingénieurs, Mohammed V University, Rabat 10050, Morocco

[3] LAGA, Université Paris 13, 99 Av J.B. Clement, 93430 Villetaneuse, France. email: fayssal@math.univ-paris13.fr

Key words: Ionization waves, Finite Volume method, Unstructured mesh, Mesh refinement, MUMPS, METIS, Parallel algorithms.

Abstract. The present paper describes a parallel unstructured-mesh Plasma simulation code based on Finite Volume method. The code dynamically refines and coarses mesh for accurate resolution of the different features regarding the electron density. Our purpose is to examine the performance of a new Parallel Adaptive Mesh Refinement (PAMR) procedure introduced on the ADAPT platform, which resolves of a relatively complicated system coupling the flow partial differential equations to the Poisson's equation. The implementation deals with the MUMPS parallel multi-frontal direct solver and mesh partitioning methods using METIS to improve the performance of the framework. The standard MPI is used to establish communication between processors. Performance analysis of the PAMR procedure shows the efficiency and the potential of the method for the propagation equations of ionization waves.

1 Introduction

Computational fluid dynamics, commonly known by the acronym "CFD", has advanced rapidly over the last two decades and is today recognized as a valuable tool in all aspects of science, technology and industry. Nowadays, hundreds of software packages are used by engineers across a wide range of application and research areas. The ultimate aim of CFD is to provide a description of fluids flows, at different levels of complexity, that must be at the same time economical and sufficiently complete. Recently, the availability of high-performance computing hardware had led to an upsurge of interest in more sophisticated mathematical models and increasingly complex and extensive computer simulations into the wider industrial community.

While experiments undertaken to investigate streamer propagation [1, 2, 3] are costly and require sophisticated equipment, performing simulations often conduct an inquiry into the effect of a given process using relatively simple models. Mostly, numerical models of streamer discharges are formulated by coupling the electrostatic to the motion of charged particles (electrons, positive and negative ions) which lead to a system of equations coupling a set of convection–diffusion–reaction equations for charged particles to stationary Poisson's equation for

the electric field. However, numerical computing for studying streamer propagation remains very expensive even with use of current vector supercomputers. Numerical simulation of ionization waves often presents difficulties due to their nonlinear form, presence of source terms, and coupling between the electron density and electric field. In addition, the difficulty in these models comes from the presence of both diffusion and convection terms which needs special discretization in some parts of the domain.

Distributed parallel computer codes have recently been successfully employed in large scale analysis for the numerical simulation of the streamer propagation in [4, 5], when the authors tackled the parallelization of 2D and 3D simulations of the streamer propagation in a static grid using the parallel solver MUMPS [6], METIS [7] for graph partitioning and the standard MPI [8] to ensure the communications between processors. Kumar et al. [9] have also successfully studied similar problems to those presented in this paper. The associated linear systems have been solved using iterative Gmres and CG solvers [10]. One difference is that in our work we consider direct methods based on LU decomposition using the MUMPS solver.

In the other hand, an important development has been the use of adaptive mesh refinement, so that a fine mesh is only placed where it is required, see for example [11, 12, 13, 14]. Structured-grid models can be adapted to more complex geometries [15], but this is often challenging. This has motivated the development of finite element discharge simulations on unstructured grids, see e.g. [16, 17, 18]. An alternative way to handle complex geometries is to apply finite volume methods to unstructured grids, as is done with the nonPDPSIM code [19, 20]. In one of our earlier works, a multilevel refinement analysis on unstructured discretization procedures was used in [21] to present an effective and accurate method in order to address this complex physical phenomenon, but still require heavy computational resources in 3D case.

In [22], six 2D fluid codes from different groups have been compared on the simulation of streamer propagation, and show that considering both AMR and MPI/OpenMP parallelization methods can significantly speed up numerical calculations. However, there is not computer code using PAMR with FV method. In this paper, we focus on the PAMR for unstructured-grid using FV method for resolving the transport equation coupled with Poisson's equation in the general case, and particularly the Streamer propagation. This work presents investigations, results, and conclusions that will no doubt be useful to scientific community and engineers who are concerned with the various aspects of computational fluid mechanics.

The paper is organized as follows. In section 2, we introduce some basic information about ADAPT platform and its positioning. In section 3, we discuss how the resulting discrete system, for the evolution equation coupled with Poisson's equation, can be effectively solved directly by combining AMR and MPI-Based parallelization in order to take advantage of those massively parallel clusters available to LAGA scientists. The implementation strategy of parallelization is also explained in details. Section 4 includes numerical results and examples that prove the efficiency of our resolution method. Finally, concluding remarks are summarized in section 5.

2 The ADAPT framework

2.1 Overview

ADAPT [21] and [23] is a C++ code which treats several physical phenomena, such as Combustion, Shallow Water, Navier Stokes, Streamer propagation. The present code consists

on two versions using Finite Volume/Element method; the first one is serial with AMR, however the second one is parallel but using fixed mesh.

Running the 3D Streamer code using the first version, takes more than 3 weeks to give a good results. The parallel version using pre-refined mesh (local refined mesh in the discharge area) require up to 8h using 512 MPI cores. Unfortunately this cannot work in all cases, especially if we want to perform branching tests [24], where the refinement zone is not necessarily known. Given this purpose, we decided to mix the AMR procedure to the MPI-Based parallelization. This paper is an important step towards this direction, it represents 2D results of streamer equations using PAMR.

2.2 Working environment

To realize all the experiments, both sequential and parallel, we worked on the MAGI cluster [1] which is located at the University of Paris 13 (Villetaneuse, France). This machine contains about 50 Dell FC430 blades with 10 Hyper-Threading dual processor cores (40 cores per blade), and 64 GB RAM per blade interconnected by InfiniBand.

3 Parallel approach

3.1 Governing equations

As a starting point, we consider the evolution equation coupled with the Poisson equation, written as:

$$\begin{cases} \dfrac{\partial u}{\partial t} + F(V,u) = S, \\ \\ \Delta P = b. \end{cases} \tag{1}$$

given that $F(V,u) = div(u.\overrightarrow{V}) - \Delta u$, $S = 0$, and $V = \overrightarrow{\nabla} P$, the previous system yields :

$$\begin{cases} \dfrac{\partial u}{\partial t} + div(u.\overrightarrow{V}) = \Delta u \\ \\ \Delta P = b. \end{cases} \tag{2}$$

The first equation is discretized using the finite volume method on an unstructured triangular mesh. The time-integration of the transport equation is performed using an explicit scheme. The discretized form of Poisson's equation leads to a linear algebraic system. We obtain the following set of equations:

$$\begin{cases} u_i^{n+1} = u_i^n - \dfrac{\Delta t}{\mu_i} \underbrace{\sum_{j=1}^{m} u_{ij} \overrightarrow{V}_{ij} \overrightarrow{n}_{ij} |\sigma_{ij}|}_{Rez\_conv} + \dfrac{\Delta t}{\mu_i} \underbrace{\sum_{j=1}^{m} \overrightarrow{\nabla} u_{ij} \overrightarrow{n}_{ij} |\sigma_{ij}|}_{Rez\_dissip} \\ \\ \mathbf{A}.\overrightarrow{P}^n = \overrightarrow{b}^n \end{cases} \tag{3}$$

[1] https://www-magi.univ-paris13.fr/doku.php

where m is the number of faces of volume μ_i, $\overrightarrow{n}_{ij}$ is the unit normal vector on the face σ_{ij} (face between volumes μ_i and μ_j) and $|\sigma_{ij}|$ is the face's length. Other variables denoted by subscript ij represent variables on the face σ_{ij}. **A** is a large sparse matrix which contains coefficients that depend only on the grid topology.

3.2 Domain decomposition and linear system $A.P = b$ solver

The 2D adaptive unstructured mesh in Figure 1 was decomposed into four sub-domains using the **METIS** algorithm. These partitions have approximately the same size, which is a good property since the workload is balanced on the homogeneous processors of our platform. To solve the linear system, the parallel solver **MUMPS** is used. The current approach with **MUMPS** shows significant advantages by avoiding problems related to pre-conditioners, and requiring fewer iterations to converge to the solution.

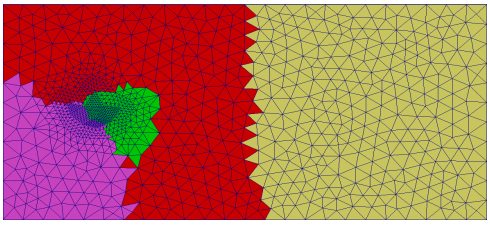

Figure 1: Decomposition of the adaptive discretization into 4 sub-domains using the METIS algorithm.

# of cells	8 MPI cores			16 MPI cores			32 MPI cores			64 MPI cores		
	cells	halos	neigh.	cells	halos	neigh.	cells	halo	neigh.	cells	halo	neigh.
27112 ($t = 0$)	3388	307	5	1671	291	6	836	200	6	434	171	8
46150 ($t = 52\ ns$)	6029	409	5	3038	395	6	1519	301	6	759	225	8

Table 1: Metis partitioning statistics using AMR

The table 1 summarize the statistics of metis partitioning for the AMR grid (grids taken from the simulation in section 4 at $t = 0$ after the first refinement and at the final physical time $t = 52\ ns$), using 8 to 64 MPI cores. For each case we represent information for the partition with the highest number of halo cells; the number of cells, the number of halo cells and the number of neighbor subdomains. This study will allows to us estimating the cost of communication part and takes a look at the right number of MPI cores needed to have the best ratio MPI core/CPU time for running this kind of problem. The statistics evince that using 8 MPI cores, the number of halo cells represents less than 9% in the chosen partition and the number of neighbor subdomain is 5 which means that the communication part wont be important. Unfortunately, using 64 MPI cores, the number of halo cells and neighbors increase and represents more than 30%, which denotes that the communication will take a large part.

3.3 Parallel discretization of the evolution equation

3.3.1 Parallel computation of u_{ij} on face σ_{ij}

The convective flux u_{ij} in the equation 3 is computed using simple upwind scheme extended by a Van Leer's type MUSCL algorithm along with Barth-Jespersen limiter [25].

$$u_{ij} = \begin{cases} u_i & if(\overrightarrow{u_{ij}} \,.\, \overrightarrow{n_{ij}}) \geq 0 \\ u_j & in \;\; other \;\; case, \end{cases} \tag{4}$$

One introduces $\tilde{u_i}$, $\tilde{u_j}$ in the equation 4 instead of u_i, u_j

$$\tilde{u_i} = u_i + \psi_i.(\overrightarrow{\nabla} u_i.\overrightarrow{r_i})$$
$$\tilde{u_j} = u_j + \psi_j.(\overrightarrow{\nabla} u_j.\overrightarrow{r_j})$$

where $\overrightarrow{\nabla} u_i$, $\overrightarrow{\nabla} u_j$ are gradients of u on cells T_i, T_j. We can compute the gradients assuming that u is a piecewise linear function and its value u_i is in the center of gravity of the cell T_i. This linear function is computed by the least square method which include all neighboring cells of T_i (neighbors with at least one common vertex).

Figure 2 represents all neighbor cells needed to compute the gradient of u. The inner information are represented by u_j (j={1,...,6}) and halo cells by u_h (h={1,...,8}).

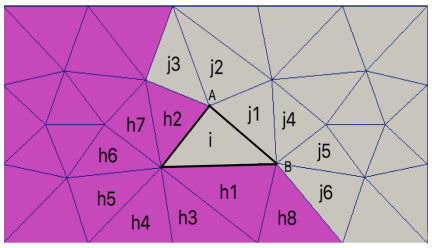

Figure 2: Computation of ∇u_i in mesh splitted into two subdomains

3.3.2 Parallel computation of the gradient ∇u_{ij} on face σ_{ij}

The gradient $\overrightarrow{\nabla} u_{ij}$ is computed using the diamond cell appearing in Figure 3 constructed by connection of centers of gravity (i, j) of cells T_i, T_j which share the face σ_{ij} and its endpoints A, B, the gradient can be written as follow

$$\overrightarrow{\nabla} u_{ij} = \frac{1}{2\mu(D_{\sigma_{ij}})}[(u_A - u_B)\overrightarrow{n}_{LR}|\sigma_{LR}| + (u_j - u_i)\overrightarrow{n}_{ij}|\sigma_{ij}|] \tag{5}$$

In Figure 2 we illustrate all information needed to compute de gradient $\overrightarrow{\nabla} u_{ij}$ on the face σ_{ij}. The algorithm 2 details how the $\overrightarrow{\nabla} u_{ij}$ is computed on face σ_{ij}.
The values of u_A, u_B on the face σ_{ij} end points are computed using the least square method. The algorithm 3 shows how it is done.

Algorithm 1: Compute the gradient $(u)_x$ on cell

```
1  C : number of cells;
2  I_xx, I_yy,D : computed using mesh;
3  A_x = sqrt(I_xx), A_y = sqrt(I_yy);
4  for i:=1 to C do
5  |  for j:=1 to inner cells neighbor by node do
6  |  |  J_x = J_x + (A_x * (u[j] - u[i]));
7  |  |  J_y = J_y + (A_y * (u[j] - u[i]));
8  |  end
9  |  for h:=1 to halo cells neighbor by node do
10 |  |  J_x = J_x + (A_x * (u[h] - u[i]));
11 |  |  J_y = J_y + (A_y * (u[h] - u[i]));
12 |  end
13 |  (u)_x[i] = ((A_x^2 * J_x - A_x * A_y * J_y)) / D
14 |  (u)_y[i] = ((A_y^2 * J_y - A_x * A_y * J_x)) / D
15 end
```

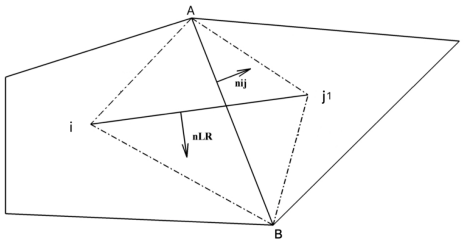

Figure 3: Illustration of control volumes and Diamond cell used in the space discretization.

3.4 Our PAMR procedure

To improve the performance of ADAPT, we incorporate an AMR using the gradient of the solution u, which has shown its effectiveness in this kind of equation [26]. Algorithm 4 summarizes the MPI-Based parallelization procedure using AMR. It provides a piece of pseudocode that shows how this approach is done according to the coupling of evolution equation with Poisson's equation. We can observe that most parts of the code are parallel, except reading and splitting mesh procedure at the beginning and during the mesh refinement. Indeed, at this step, we compute the criterion which depends on the gradient of the solution u and we refine/coarse mesh. The mesh refinement strategy is performed on the master processor to avoid both unbalancing partitions and the difficulty of conformity due to multilevel refinement [21]. To solve the linear system, the matrix A in equation (3) is computed at each refinement step because it depends only on the mesh. The halo information sent each iteration are; electron density, ion density, electric field and velocity field. These communications are performed using MPI_Alltoallv routine.

Algorithm 2: Compute ∇u_{ij} on face ij

F:number of faces;
$\overrightarrow{\nabla} u$:Vector2d;
for *k:=1* **to** F **do**
  A:first node of face;
  B:second face node;
  i:center of gravity of cell T_i;
  j:center of gravity of cell T_j;
  $mes = \frac{1}{2\mu(D_{\sigma_{ij}})}$;
  if *Inner faces* **then**
    $\overrightarrow{\nabla} u(k) = mes * (u_A - u_B)\overrightarrow{n}_{LR}|\sigma_{LR}| + (u_j - u_i)\overrightarrow{n}_{ij}|\sigma_{ij}|$;
  end
  else if *Halo faces* **then**
    $\overrightarrow{\nabla} u(k) = mes * (u_A - u_B)\overrightarrow{n}_{LR}|\sigma_{LR}| + (u_h - u_i)\overrightarrow{n}_{ih}|\sigma_{ih}|$;
  end
end

Algorithm 3: Compute value on node A (u_A)

u_A:double;
N :number of nodes;
Alpha :weight coming from the least square method;
for *n:=1* **to** N **do**
  for *c:=1* **to** *inner cells around node n* **do**
    $u_A(n) += Alpha(c) * u_j(c)$;
  end
  for *m:=1* **to** *halo cells around node* **do**
    $u_A(n) += Alpha(m) * u_h(m)$;
  end
end
return u_{node};

Our contribution will serve to enhance the performance of 3D simulations, in which we consider the same type of equations and strategy. It is worth noting that the structure of the 3D grid will be more complex because tetrahedra are used instead of triangles. Using the adaptive 3D version may take up to 3 weeks to give results.

Algorithm 4: Algorithm of parallel ADAPT

```
1  W:double;
2  if rank==0 then /* for the master processor                    */
3  |  Read mesh data;
4  |  Split the mesh with METIS;
5  |  Distribute mesh information (connectivity & coordinates) to all processors;
6  end
7  W=0;
8  for each rank do /* for each processor                         */
9  |  Construct the local grid (information required for FV discretization);
10 |  Initialize conditions and create constants;
11 |  Send the information of halo cells to neighbor subdomains;
12 |  Apply boundary conditions;
13 end
14 for each iteration do
15 |  for all rank do /* for all processors                       */
16 |  |  Gather electric density information to the master;
17 |  |  Compute refinement Criteria and mark element to refine/coarse;
18 |  |  Send the marked information to the master process;
19 |  end
20 |  if rank==0 then
21 |  |  Adapt the mesh: Refine/coarse where necessary;
22 |  |  Split and distribute mesh information to all processors;
23 |  end
24 |  for each rank do
25 |  |  Construct the new local grid;
26 |  |  Construct submatrix of linear system;
27 |  end
28 |  for all rank do
29 |  |  Solve linear system using MUMPS;
30 |  end
31 |  for each rank do
32 |  |  Send the information of halo cells to neighbor subdomains;
33 |  |  Apply boundary conditions;
34 |  |  Compute fluxes of convection, diffusion and source term;
35 |  |  Update solution :
   |  |    W^{n+1} = W^n + Δt * (rez_conv + rez_dissip + rez_source);
36 |  |  Save results in parallel way using Paraview;
37 |  end
38 end
```

4 Numerical examples

4.1 Equations for streamer propagation

Here, we are interested in a simple two-dimensional model for negative streamer investigated in [21] among others. Since the charged particles can be considered as continuum, the particles are dealt with in form of densities and we can use continuum mechanics for them. Hence as mentioned above, this model is able to capture the streamer propagation and consists of a convection-diffusion-reaction equation for the electron density along coupled with the Poisson's equation for the electric potential. In what follows, we use boldface notation to denote vectors. Hence, for the electron density n_e, the ion density n_i, the electric potential V and the electric field $\vec{E}$, we solve the following equations

$$\begin{aligned} \frac{\partial n_e}{\partial t} + div(n_e \overrightarrow{v_e} - De\overrightarrow{\nabla} n_e) &= S_e, \\ \frac{\partial n_i}{\partial t} &= S_e, \\ \Delta V = -\frac{e}{\epsilon}(n_i - n_e) \ , \\ \vec{E} = -\overrightarrow{\nabla} V \ , \end{aligned} \tag{6}$$

Here, we solve the test example of discharge propagation in a homogeneous electric field described in [21]. Hence, we solve equations (6) in the rectangular domain $[0,1]\times[0,0.5]$ subjected to a plane anode with $V = 25000\ volts$ on the left and a plane cathode with $V = 0\ volts$ on the right whereas periodic boundary conditions are imposed on the upper and lower walls. Initially, $n_e(x,y,0) = n_i(x,y,0) = 10^{16} \times e^{-\frac{(x-0.2)^2+(y-0.25)^2}{\sigma^2}} + 10^9$, with $\sigma = 0.01$. All our simulations are carried on the MAGI cluster.

Reproduction of the plasma discharge's phenomenon requires a very fine mesh (more than 1M cells in 2D using fixed mesh). In our test case, we assume that the results on the very fine mesh are the reference solutions.

The AMR is illustrated in figures 4 and 5, the evolution of refined zone follows the same streamer propagation trajectory.

Relative errors are calculated and summarized in Table 2. The minimum grid size of 1.5 nm is used for both reference grid and AMR grid to have the same Δt. The number of cells in AMR grid is 4110 at $t = 0$. Figure 6 depicts the numerical results obtained using the PAMR procedure with 32 MPI cores, distribution of electron densities n_e and the velocity fields $\vec{v}_e$ at two different times. The PAMR procedure for the finite volume method resolved this case successfully and accurately captured the physical features similarly to the first example. Here, the criteria for mesh refinement is based on the gradient of the ion density n_e and the source term S_e.

In table 4, a comparison is made between the parallel version of 2D streamer using fixed mesh and AMR one. We can observe that using 32 MPI cores, the computational time using adaptive approach is 1147 ($\sim$19 minutes) instead of 1944 ($\sim$ 40 minutes) using fixed mesh and 256 MPI cores. The CPU time elevation using 64 MPI cores can be explained by the fact that :

- the maximum cells number during the overall simulation does not exceed 46150, means that the cells number by partition is less than 760,

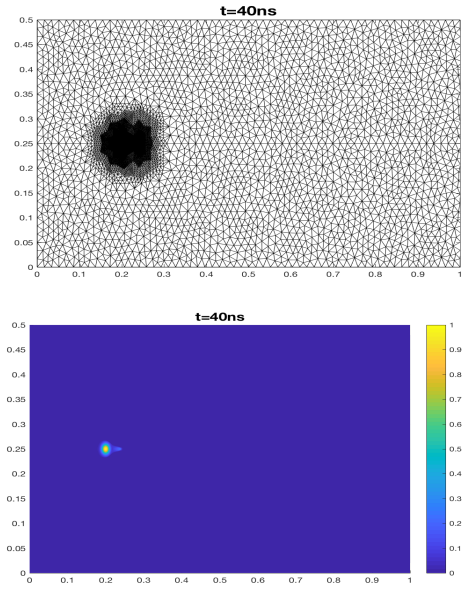

Figure 4: Adaptive mesh evolution (top) and Electron density distributions for propagation of a discharge (bottom) at times $t = 40\ ns$.

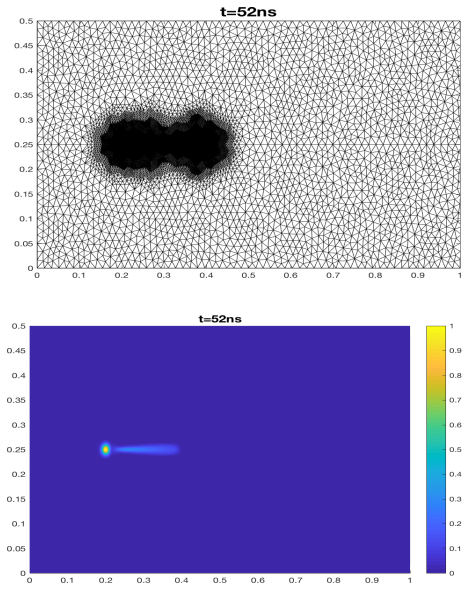

Figure 5: Adaptive mesh evolution (top) and Electron density distributions for propagation of a discharge (bottom) at times $t = 52\ ns$.

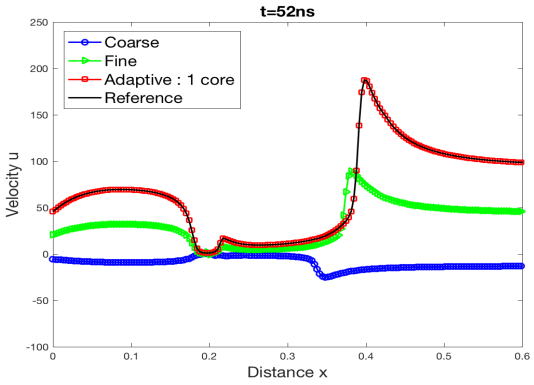

Figure 6: Cross-section at $x = 0.25\ cm$ of the velocity at time 52 ns.

	Min N_{cells}	Max N_{cells}	Min cell size	Error in n_e	Error in $\vec{v}_e$	Timing (s)
Coarse	4.36×10^4	4.36×10^4	47 nm	0.3104	1.4747	6860
Fine	2.97×10^5	2.97×10^5	5.3 nm	0.0924	0.4387	31315
Adaptive	4.11×10^3	4.61×10^4	1.5 nm	0.0013	0.0041	18340
Reference	1.19×10^6	1.19×10^6	1.5 nm	—	—	291600

Table 2: A summary of simulation setting, relative errors and computational times (in seconds) for the Streamer propagation at time $t = 52\ ns$ using single MPI core.

- the halo cells number increase (see table 3).

# of cells	Partitions			
	8	16	32	64
27112 ($t = 0$)	1725	2891	4368	8202
46150 ($t = 52\ ns$)	2453	4261	6391	10703

Table 3: Halo cells vis-a-vis Partitions

The communication cost will cover the computational one.

Table 5 shows the computational time spent on each part added in the new approach. The results shows a perfect scalability in the submatrices update part, a good scalability in the local grid construction, and a small computational time increase in the mesh generation part (timing of mesh refinement + timing of mesh partitioning + timing of mesh distribution) due to the management of halo cells which increases by increasing the number of partitions.

# of cores	Timing (s)	
	Adaptive mesh	Fixed mesh
1	18340	291600
2	9968	153473
4	5394	81000
8	3056	41657
16	1834	21130
32	1147	11215
64	1210	5832
128	-	3240
256	-	1944

Table 4: Computational times for The Streamer propagation using PAMR

# of cores	Mesh generation	Local grid	Submatrix
1	52.5	381.4	54.1
2	50.5	222.1	26.7
4	52.4	117.8	12.8
8	54.1	68.12	6.22
16	58.2	38.65	3.13
32	63.1	26.36	1.43
64	71.7	23.71	0.67

Table 5: Computational times in (s) for mesh generation, local grid construction and submatrix update

5 Conclusion

We have proposed an efficient PAMR procedure for finite volume solution of ionization waves in planar electric fields (The percentage of serial part in the total computational time takes less than 6% using 64 partitions). Numerical results have been presented for the propagation of ionization waves in a rectangular domain. The originality of this work lies in the running of parallel code using AMR. The workflow is realized: (1) using external tools such as METIS for mesh partitioning in order to enable the computational load balance for the global assembly, (2) using MPI to assure communication, (3) using MUMPS solver to solve the system of linear equations. (4) using adaptive procedure to avoid the use of very fine mesh. In summary, the most important advantage of the presented schemes is the potential to produce highly accurate solutions at low computational cost.

We conclude that the campaign of numerical tests yields expected results, meaning that our parallel implementation goes into the right direction at the moment and thus contributes significantly to employment of such methods. Overall, this approach seems to be a suitable tool for solving complex multidimensional streamer problems which require a huge calculation time. These findings can be exploited to achieve accurate 3D simulation of the streamer discharge with a reasonable computation cost, which allow for more realistic real-life geometries analysis.

This part is the subject of an ongoing investigation that is to be published in our future works.

REFERENCES

[1] P. A. Vitello, B. M. Penetrante, and J. N. Bardsley. Simulation of negative-streamer dynamics in nitrogen. *Phys. Rev. E*, 49:5574–5598, Jun 1994.

[2] Ute Ebert, Sander Nijdam, Chao Li, Alejandro Luque, Tanja Briels, and Eddie van Veldhuizen. Review of recent results on streamer discharges and discussion of their relevance for sprites and lightning. *Journal of Geophysical Research: Space Physics*, 115(A7), 2010.

[3] J. Fořt, J. Karel, D. Trdlička, F. Benkhaldoun, I. Kissami, J.-B. Montavon, K. Hassouni, and J. Zs. Mezei. Finite volume methods for numerical simulation of the discharge motion described by different physical models. *Advances in Computational Mathematics*, 45(4):2163–2189, Aug 2019.

[4] Imad Kissami, Christophe Cérin, Fayssal Benkhaldoun, and Gilles Scarella. Towards parallel CFD computation for the ADAPT framework. In Jesús Carretero, Javier García Blas, Ryan K. L. Ko, Peter Mueller, and Koji Nakano, editors, *Algorithms and Architectures for Parallel Processing - 16th International Conference, ICA3PP 2016, Granada, Spain, December 14-16, 2016, Proceedings*, volume 10048 of *Lecture Notes in Computer Science*, pages 374–387. Springer, 2016.

[5] Imad Kissami, Christophe Cérin, Fayssal Benkhaldoun, and Gilles Scarella. Parallelization of the ADAPT 3d streamer propagation code. In *2016 IEEE Intl Conference on Computational Science and Engineering, CSE 2016, and IEEE Intl Conference on Embedded and Ubiquitous Computing, EUC 2016, and 15th Intl Symposium on Distributed Computing and Applications for Business Engineering, DCABES 2016, Paris, France, August 24-26, 2016*, pages 242–245. IEEE Computer Society, 2016.

[6] Patrick Amestoy, Iain S. Duff, Jean-Yves L'Excellent, and Jacko Koster. MUMPS: A general purpose distributed memory sparse solver. In *Applied Parallel Computing, New Paradigms for HPC in Industry and Academia, 5th International Workshop, PARA 2000 Bergen, Norway, June 18-20, 2000 Proceedings*, pages 121–130, 2000.

[7] George Karypis and Vipin Kumar. A fast and high quality multilevel scheme for partitioning irregular graphs. *SIAM J. Sci. Comput.*, 20(1):359–392, December 1998.

[8] Message P Forum. Mpi: A message-passing interface standard. Technical report, USA, 1994.

[9] Pawan Kumar, Stefano Markidis, Giovanni Lapenta, Karl Meerbergen, and Dirk Roose. High performance solvers for implicit particle in cell simulation. *Procedia Computer Science*, 18:2251–2258, 2013.

[10] Youcef Saad and Martin H. Schultz. Gmres: A generalized minimal residual algorithm for solving nonsymmetric linear systems. *SIAM Journal on Scientific and Statistical Computing*, 7(3):856–869, 1986.

[11] C. Montijn, W. Hundsdorfer, and U. Ebert. An adaptive grid refinement strategy for the simulation of negative streamers. *Journal of Computational Physics*, 219(2):801–835, 2006.

[12] S. Pancheshnyi, P. Ségur, J. Capeillère, and A. Bourdon. Numerical simulation of filamentary discharges with parallel adaptive mesh refinement. *Journal of Computational Physics*, 227(13):6574–6590, 2008.

[13] Jannis Teunissen and Ute Ebert. Simulating streamer discharges in 3d with the parallel adaptive afivo framework. *Journal of Physics D: Applied Physics*, 50(47):474001, oct 2017.

[14] Max Duarte, Zdeněk Bonaventura, Marc Massot, Anne Bourdon, Stéphane Descombes, and Thierry Dumont. A new numerical strategy with space-time adaptivity and error control for multi-scale streamer discharge simulations. *Journal of Computational Physics*, 231(3):1002–1019, 2012. Special Issue: Computational Plasma Physics.

[15] Sebastien Celestin, Zdenek Bonaventura, Barbar Zeghondy, Anne Bourdon, and Pierre Ségur. The use of the ghost fluid method for poisson's equation to simulate streamer propagation in point-to-plane and point-to-point geometries. *Journal of Physics D: Applied Physics*, 42(6):065203, feb 2009.

[16] G.E Georghiou, R Morrow, and A.C Metaxas. An improved finite-element flux-corrected transport algorithm. *Journal of Computational Physics*, 148(2):605–620, 1999.

[17] Olivier Ducasse, Liberis Papageorghiou, Olivier Eichwald, Nicolas Spyrou, and Mohammed Yousfi. Critical analysis on two-dimensional point-to-plane streamer simulations using the finite element and finite volume methods. *IEEE Transactions on Plasma Science*, 35(5):1287–1300, 2007.

[18] M. Zakari, H. Caquineau, P. Hotmar, and P. Ségur. An axisymmetric unstructured finite volume method applied to the numerical modeling of an atmospheric pressure gas discharge. *Journal of Computational Physics*, 281:473–492, 2015.

[19] Zhongmin Xiong and Mark J Kushner. Surface corona-bar discharges for production of preionizing UV light for pulsed high-pressure plasmas. *Journal of Physics D: Applied Physics*, 43(50):505204, dec 2010.

[20] Zhongmin Xiong and Mark J Kushner. Atmospheric pressure ionization waves propagating through a flexible high aspect ratio capillary channel and impinging upon a target. *Plasma Sources Science and Technology*, 21(3):034001, apr 2012.

[21] F. Benkhaldoun, J. Fořt, K. Hassouni, and J. Karel. Simulation of planar ionization wave front propagation on an unstructured adaptive grid. *J. Comput. Appl. Math.*, 236:4623–4634, 2012.

[22] B Bagheri, J Teunissen, U Ebert, M M Becker, S Chen, O Ducasse, O Eichwald, D Loffhagen, A Luque, D Mihailova, J M Plewa, J van Dijk, and M Yousfi. Comparison of six simulation codes for positive streamers in air. *Plasma Sources Science and Technology*, 27(9):095002, sep 2018.

[23] Fayssal Benkhaldoun, Jaroslav Fořt, Khaled Hassouni, Jan Karel, Gilles Scarella, and David Trdlička. A full 3-d dynamically adaptive unstructured grid finite-volume approach to simulate multiple branching in streamer propagation. *IEEE Transactions on Plasma Science*, 42(10):2420–2421, 2014.

[24] Sander Nijdam, Jannis Teunissen, and Ute Ebert. The physics of streamer discharge phenomena. *Plasma Sources Science and Technology*, 29(10):103001, nov 2020.

[25] Jiri Blazek. *Computational fluid dynamics: principles and applications.* Butterworth-Heinemann, 2015.

[26] I. Elmahi, F. Benkhaldoun, R. Borghi, and S. Raghay. Ignition of fuel issuing from a porous cylinder located adjacent to a heated wall: a numerical study. *Combust. Theory Modelling*, 8:789–809, 2004.