Available online at www.sciencedirect.com

ScienceDirect

Journal of Computational Physics 227 (2007) 898–918

JOURNAL OF COMPUTATIONAL PHYSICS

www.elsevier.com/locate/jcp

An asynchronous scheme with local time stepping for multi-scale transport problems: Application to gas discharges

Thomas Unfer [a,*], Jean-Pierre Boeuf [a], François Rogier [b], Frédéric Thivet [b,c]

[a] *LAPLACE, Université Paul Sabatier, 118 Route de Narbonne, 31062 Toulouse Cedex, France*
[b] *ONERA, 2 Avenue Edouard Belin, 31400 Toulouse, France*
[c] *Université de Toulouse, ISAE, 10 Avenue Edouard Belin 31055 Toulouse, France*

Received 29 November 2006; received in revised form 4 June 2007; accepted 22 July 2007
Available online 3 August 2007

Abstract

This paper presents an asynchronous integration scheme with local time stepping for transport problems. The concept consists in associating refresh time tags to the interface fluxes between cells and to the source terms within the cells rather than to the cell themselves. This scheme is less diffusive numerically than its synchronous equivalent. This method is very effective in terms of computation time for problems with localized sharp minima in the CFL condition. The method is then applied to dielectric barrier discharges for aerodynamic flow control.

Keywords: Asynchronous; Transport problem; Local time stepping; Gas discharges; CFL condition

1. Introduction

In many physical systems, multi-scale phenomena and steep gradients make the integration of transport equations a crucial numerical issue in terms of accuracy and computation time. This is particularly true for the physics of discharges in gas, where slow heavy species are coupled to fast electrons through Poisson's equation. The time scales of the discharge formation are governed by the transport, chemical reactions and dielectric relaxation.

Two types of numerical integration methods are commonly used in the modeling of gas discharge. Fully explicit methods: such methods require that the time steps satisfy three criterions which are:

- The CFL condition with respect to the fastest velocity (which is typically the electron velocity).
- The stability constraint due to the integration of the kinetic scheme (typically ionization).
- The dielectric relaxation time.

[*] Corresponding author. Tel.: +33 5 61 55 66 68.
E-mail address: unfer@laplace.univ-tlse.fr (T. Unfer).

0021-9991/$ - see front matter © 2007 Elsevier Inc. All rights reserved.
doi:10.1016/j.jcp.2007.07.018

Explicit schemes give accurate results but at a really expensive computational time-cost.

Semi-implicit schemes: semi-implicit transport-Poisson solvers have been developed with the exponential scheme (Scharfetter and Gummel). In this method the time step is no longer limited by the CFL conditions or the dielectric relaxation time [11]. However the scheme is generally quite diffusive and remain first order accurate. In some cases, the dynamics of the plasma formation is not well reproduced and large errors occurs. Here, we propose an explicit time stepping technique locally adapted, in terms of the stability condition, for each species.

Similar approaches have been developed and are briefly discussed below. Berger and Oliver introduced the adaptive mesh refinement [1] for block-structured meshes in which a time synchronization between large and fine grids is proposed. This technique has been successfully applied to various physical processes and particularly to computational fluid dynamics. Local time stepping algorithms have also been studied in electromagnetic compatibility (EMC) applications. Simplectic schemes (see [2]) or multisymplectic schemes have been implemented for Maxwell solvers (or more generally in model derived from an Hamilton's principle). Thus, the local time steps have to be chosen as a fraction of the global time step.

Recently a new approach presented by Omelchenko and Karimabadi [3] introduced an asynchronous scheme for drift–diffusion transport. In this paper we propose a different approach for asynchronous time stepping by associating independent time tags directly to the interface fluxes between the cells and to the source terms within the cells. In this approach the cells synchronization is replaced by an interface synchronization for the fluxes. The treatment of the source terms does not require any synchronization. We also chose to always retain the local CFL condition (no "idle" state or flux capacitor), which leads to a significant numerical diffusion reduction. The critical point for speeding up an asynchronous method in terms of computation time is the searching algorithm for the most urgent time tag to be treated, and some alternative options are presented here. From a theoretical viewpoint, Dawson and Kirby showed the convergence of periodically synchronized higher order schemes in [4]. The present paper gives a short proof of the convergence of first order asynchronous scheme.

We propose an asynchronous, conservative, stable scheme which is computationally attractive for problems with locally sharp minima in the stability conditions.

The efficient transport/Poisson coupling for an asynchronous scheme is a crucial issue for both accuracy and computation time in gas discharge modeling. An original filtering technique is proposed here to resolve this part.

This paper is organized as follows: Section 2 describes the asynchronous scheme in detail and gives elements of proof of its convergence. Section 3 shows a numerical comparison with a standard explicit synchronous scheme. This section aims at validating the concept for a simple test case and at illustrating the application domain of the method. Section 4 focuses on the gas discharge application illustrated with its numerical application to dielectric barrier discharge (DBD) for aerodynamic flow control. Followed by the conclusion in Section 5.

2. Numerical integration method for the advection equation

2.1. Concept

Explicit integration techniques face a severe time step restriction known as the Courant–Friedrich–Levy (CFL) condition. For the system to remain numerically stable this condition has to be imposed over the whole mesh, using classical methods. As a result the integration time step of the system is limited by the minimum of the CFL conditions all over the domain.

Let us consider a conservation law in 1D for simplicity

$$\frac{\partial n}{\partial t} + \frac{\partial \Gamma}{\partial x} = S \tag{1}$$

$\Gamma = nv(x)$ with $v(x) \geqslant 0$ a given advection velocity and S some source term

In this case the first-order upwind scheme over a regular mesh would be:

$$\begin{cases} n_\mathrm{i}^{k+1} = n_\mathrm{i}^k - \frac{\Delta t}{\Delta x}(n_\mathrm{i}^k v_{i+\frac{1}{2}} - n_{i-1}^k v_{i-\frac{1}{2}}) + S_\mathrm{i}\Delta t \\ t_{k+1} = t_k + \Delta t \end{cases} \tag{2}$$

n_i are the density values at the cell centers $x_i = i\Delta\ x$, $v_{i+\frac{1}{2}}$ is the velocity values taken at the cell interfaces $x_{i+\frac{1}{2}} =\ i + \frac{1}{2})\Delta x$.

For the standard scheme Δt is the minimum of the CFL condition all over the mesh.

The asynchronous scheme assumes that each flux or source term is updated independently according to a refresh time tag with respect to a global simulation clock. Over some user defined time step the solution is built as follows:

$$\begin{cases} n_\mathrm{i}^{k+1} = n_\mathrm{i}^k - \frac{1}{\Delta x}(\sum_p \Delta t_{\Gamma+}^p \Gamma_{i+\frac{1}{2}}(t_p) - \sum_q \Delta t_{\Gamma-}^q \Gamma_{i-\frac{1}{2}}(t_q)) + \sum_r \Delta t_S^r S_\mathrm{i}(t_r) \\ t_{k+1} = t_k + \Delta t_\mathrm{output} \end{cases} \tag{3}$$

For the scheme to be stable each local time step is limited by the local value of the CFL condition.

2.2. Algorithm

Each mesh cell has its own value, value time tag, and variation rate. The idea is to store the value of each term (fluxes or source term) of the variation rate and let them evolve independently:

$$\frac{\partial n}{\partial t_\mathrm{i}} = S_\mathrm{i} - \frac{\Gamma_{i+\frac{1}{2}}}{\Delta x} + \frac{\Gamma_{i-\frac{1}{2}}}{\Delta x} = \frac{\partial n^S}{\partial t_\mathrm{i}} + \frac{\partial n^{\Gamma+}}{\partial t_\mathrm{i}} + \frac{\partial n^{\Gamma-}}{\partial t_\mathrm{i}}$$

Various time variables are needed: $t_\mathrm{simulation}$ is the current discrete simulation time, t_i^n is the time tag associated with the stored value of n_i, t_i^Γ is the refresh time tag of Γ_i and t_i^S is the refresh time tag of S_i.

During the simulation, $t_\mathrm{simulation}$ jumps discretely from the most urgent refresh time tag to the next most urgent.

2.2.1. Initialization

(1) Initialize all density and fluxes.
(2) Initialize all refresh time tags to the initial time.

2.2.2. Proceed until the simulation time is completed

(1) Find the most urgent flux or source term to be refreshed.
(2) $t_\mathrm{simulation}$ becomes this most urgent refresh time tag.
(3) Compute the value of the density which variation rate will evolve or which are needed for flux computation (only a few cells are involved).
- $n_j = n_j + \left(\frac{\partial n^S}{\partial t_j} + \frac{\partial n^{\Gamma+}}{\partial t_j} + \frac{\partial n^{\Gamma-}}{\partial t_j}\right)(t_\mathrm{simulation} - t_j^n)$
- $t_j^n = t_\mathrm{simulation}$

with $j \in [i, i+1]$ for first order flux, $j \in [i-1, i+2]$ for second order flux (e.g. MUSCL) or $j = i$ for source term
(4) Compute the new value of the flux or source term and update its variation rate according to:
- $\frac{\partial n^{\Gamma+}}{\partial t_\mathrm{i}} = -\frac{\Gamma_{i+\frac{1}{2}}}{\Delta x}$
- $\frac{\partial n^{\Gamma-}}{\partial t_{i+1}} = \frac{\Gamma_{i+\frac{1}{2}}}{\Delta x}$ or $\frac{\partial n^S}{\partial t_\mathrm{i}} = S_\mathrm{i}$

(5) Compute the new refresh time tag of this flux or source term according to the local CFL condition or reaction kinetic $t_\mathrm{i}^\Gamma = t_\mathrm{i}^\Gamma + \Delta t_\mathrm{CFL}$ when updating a flux or $t_\mathrm{i}^S = t_\mathrm{i}^S + \Delta t_S$ when updating a source

(6) If the solution is to be monitored before the next most urgent refresh time tag, build the solution at the output time tag and store it.

$$n_{\mathrm{i}}^{\mathrm{output}} = n_{\mathrm{i}} + \left(\frac{\partial n^S}{\partial t_{\mathrm{i}}} + \frac{\partial n^{\Gamma+}}{\partial t_{\mathrm{i}}} + \frac{\partial n^{\Gamma-}}{\partial t_{\mathrm{i}}}\right)(t^{\mathrm{output}} - t_{\mathrm{i}}^{n}) \text{ for all cells}$$

Notes:

- step 3. serves two purposes: integrating the previous variation before the variation rate is updated and updating the density values needed for flux or source term computation
- step 6. occurs at user defined time steps when the solution is to be monitored, it has no effect on the actual density variables within the algorithm. Because the synchronized solution is never present in memory it has to be constructed at the output time tag.

2.3. Comparison with the original asynchronous scheme (see [3])

Our approach is to always use the local CFL as refresh time steps in order to minimize numerical diffusion (see 2.7). On the other hand the innovative "d*f*" and "flux capacitor" concepts introduced by Omelchenko and Karimabadi leads to the definition of a threshold perturbation level below which a perturbation is not propagated.

Our goal is to develop simulation codes in the field of gas discharges modeling. One of the key phenomena in this field is electron avalanching which leads to an exponential growth of charged particles. In this precise context defining a threshold of electron density variation is not a good approach. That is why we compare our scheme with the original one with the "d*f*" threshold set to zero on an uniform 1D drift problem with no source over a N cell mesh. This problem is obviously synchronous. The original scheme would call N times a single cell. For each cell i it means computing both incoming and outgoing fluxes then updating the derivative of cell i and half derivatives of cells $i-1$ and $i+1$. In this way the original scheme is conservative. For the uniform drift problem it means computing $2N$ fluxes and $2N$ derivatives for each time step. Our scheme would call $N+1$ times a single flux. For each flux it means computing the flux and two half derivatives (except at boundaries). This scheme is also conservative (see Section 2.6.1). For the uniform drift problem it means globally for each time step computing $N+1$ fluxes and N derivatives. Consequently the original scheme computes twice more fluxes than this scheme for given time steps. Flux computation represents a heavy CPU load when using second order method in space such as MUSCL method. Besides when using the local CFL condition the velocity used to compute the time step corresponds exactly to the one used to compute the flux.

2.4. Most urgent time tag searching algorithm

2.4.1. Binary tree strategy

Finding the most urgent refresh time tag is a crucial programming issue for the algorithm to be attractive in terms of CPU time. In [5] Karimabadi et al. showed that for particles in cell simulation the best algorithm developed so far is a priority queue coded as a heap-sorted dynamically re-sizable array. This implementation outpaces "binary trees" implementation in a worst case when full sorting a random data set. Because our algorithm does not rely on the "d*f*" and "flux capacitor" concepts introduced in [3] there is no "wake-up call" of any flux or source term. Consequently in our approach there is no need to perform a full sorting of the time tags. Furthermore the number of elements to keep sorted is constant and well-known in advance. So, in this precise case the binary tree may compete fairly with the priority queue. We kept the minimal heap data structure but coded it as a binary tree. In these conditions the tree remains always almost sorted except for the root value which has to be replaced within the branches leading to a O(log(n)) algorithm where n is the element number in the tree. This simple structure can be optimally stored in memory in order to minimize cache misses (see Fig. 2).

2.4.2. Discrete time scheduler

Section 2.7 shows that numerical diffusion is minimum when local time stepping is close to the local CFL condition. An interesting trade-off between precision and CPU time can be made if one accepts that each local time step is some discrete multiple of a minimal time increment. The idea is to take some upper and lower bound to the local time steps:

$$\Delta t \in [\Delta t_{\min}, m\Delta t_{\min}] \tag{4}$$

The core data structure of the algorithm is a circular array of m elements of the "task" data type (see Fig. 1). By circular array we mean that the mth element of the array is "followed" by the first element. A "task" consist in a time tag and a pointer to a pending action stack (fluxes or source terms to be refreshed at this precise time tag). The current task is identified by using its rank within the array. Finding the most urgent time tag is then a O(1) operation: accessing the top element of the pending action stack at current task. Replacing the new event within the scheduler is also a O(1) operation: if the next occurrence of this action is in $i\Delta t_{\min}$, this action as to be replaced on the top of the pending action stack located at "current task + i" in the circular array. When the current task has an empty action stack, then the next task becomes the current task, and the former current task is freed within the array and becomes a new task at $t_{\text{task}}^{\text{last}} + \Delta t_{\min}$.

The discrete time scheduler is a O(1) algorithm to perform the most urgent time tag searching. Parameters of the scheduler $[\Delta t_{\min}, m]$ have to be oversized compared to the time steps of the simulation, for instance in the gas discharge simulations (see Section 4.3) the scheduler parameters are $\Delta t_{\min} = 5 \times 10^{-14}$ and $m = 2 \times 10^6$.

2.5. Memory management

2.5.1. Time tag tree

The basic data structure of the time tag tree is a time tag, a code identifying the element and two pointers to the branches. The tree is allocated in memory as showed on Fig. 2 so that memory cache miss occurrence is mitigated. When cache misses occur it is always to the same direction, up or down from the root until the correct position is reached.

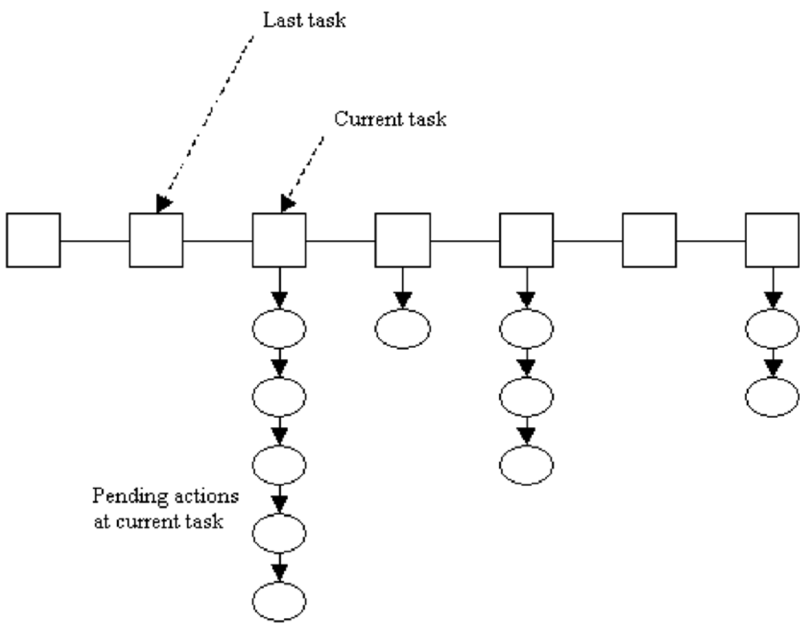

Fig. 1. Discrete time scheduler.

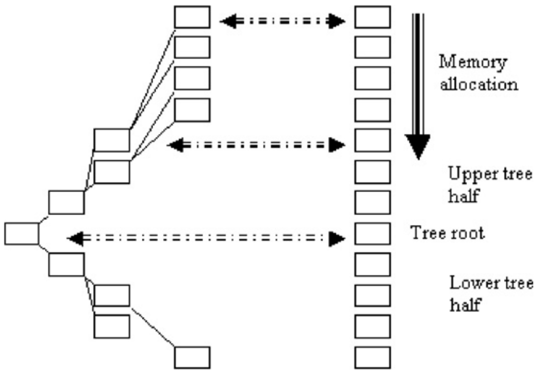

Fig. 2. Tree storage in memory.

2.5.2. Data storage

Classical synchronous schemes take advantage of cache memory because treatment loop generally follow data storage of large matrices in memory. This is not the case for asynchronous scheme because they keep jumping from one area of the matrices to the other. So cache misses for asynchronous schemes are much likelier to occur. In order to minimize this phenomenon a data structure as been implemented where all mesh data is collocated. This means that for single mesh coordinates this structure holds density, density time tag, outgoing flux, variation rate components, source term, etc in the same memory area.

2.6. Convergence of the first-order upwind asynchronous scheme

2.6.1. Conservativeness

Consider the first-order upwind asynchronous scheme with no source term and zero flux at the boundaries $\Gamma_{\frac{1}{2}} = \Gamma_{n+\frac{1}{2}} = 0$. The scheme is clearly conservative:

$$\sum n_{\mathrm{i}}^{t+\Delta t} = \sum n_{\mathrm{i}}^{t} + \Delta t \sum \left(\frac{\partial n^{\Gamma+}}{\partial t_{\mathrm{i}}} + \frac{\partial n^{\Gamma-}}{\partial t_{\mathrm{i}}} \right) = \sum n_{\mathrm{i}}^{t} \tag{5}$$

because by definition of the algorithm

$$\frac{\partial n^{\Gamma-}}{\partial t_{i+1}} = \frac{\Gamma_{i+\frac{1}{2}}}{\Delta x} = -\frac{\partial n^{\Gamma+}}{\partial t_{\mathrm{i}}} \tag{6}$$

2.6.2. Stability

With the notations of Fig. 3, the scheme can be written between t_q and t_{q+1} assuming any updates of $\Gamma_{i-\frac{1}{2}}$.

$$\begin{aligned} n_{\mathrm{i}}^{t_{q+1}} &= n_{\mathrm{i}}^{t_q} - \frac{1}{\Delta x}\left[\Delta t_q n_{\mathrm{i}}^{t_q} v_{i+\frac{1}{2}} - \tau n_{i-1}^{t_p} v_{i-\frac{1}{2}} - \sum n_{i-1}^{t_{p+k}} v_{i-\frac{1}{2}} \Delta t_{p+k} - n_{i-1}^{t_{p+k+1}} v_{i-\frac{1}{2}} (t_{q+1} - t_{p+k+1})\right] \\ n_{\mathrm{i}}^{t_{q+1}} &\geqslant n_{\mathrm{i}}^{t_q}\left(1 - \frac{\Delta t_q}{\Delta x} v_{i+\frac{1}{2}}\right) \end{aligned} \tag{7}$$

Eq. (7) shows that the local CFL condition insures the positivity of the scheme, so stability as well because it is conservative.

2.6.3. Consistency under local CFL condition

Suppose that the drift velocity is smooth and the local time steps are proportional to the local CFL condition i.e.:

$$\Delta t_q = \frac{k_{\mathrm{CFL}} \Delta x}{v_{i-\frac{1}{2}}} \simeq k_{\mathrm{CFL}} \frac{\Delta x}{v}\left(1 - \frac{\partial v}{\partial x}\frac{\Delta x}{2v}\right) k_{\mathrm{CFL}} \epsilon]0,1[\tag{8}$$

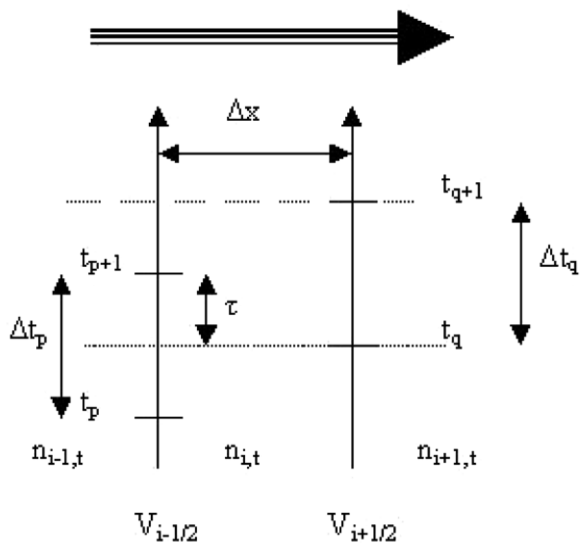

Fig. 3. Time/Space diagram of the asynchronous scheme assuming positive drift velocity.

$$\Delta t_p = \frac{k_{\mathrm{CFL}}\Delta x}{v_{i+\frac{1}{2}}} \simeq k_{\mathrm{CFL}}\frac{\Delta x}{v}\left(1+\frac{\partial v}{\partial x}\frac{\Delta x}{2v}\right) = \Delta t_q\left(1+\frac{\partial v}{\partial x}\frac{\Delta x}{v}\right) \tag{9}$$

For $\Delta x \leqslant v\frac{\partial x}{\partial v}$ the following condition is imposed on the local time steps: $|\Delta t_p - \Delta t_q| \leqslant \Delta t_q$, this means that three cases have to be considered:

- First case: $t_p \leqslant t_q \leqslant t_{p+1} \leqslant t_{q+1}$

$$\begin{aligned}
n_{\mathrm{i}}^{t_{q+1}} &= n_{\mathrm{i}}^{t_q} - \frac{1}{\Delta x}\left[\Delta t_q n_{\mathrm{i}}^{t_q} v_{i+\frac{1}{2}} - \tau n_{i-1}^{t_p} v_{i-\frac{1}{2}} - n_{i-1}^{t_{p+1}} v_{i-\frac{1}{2}}(\Delta t_q - \tau)\right]\\
n_{\mathrm{i}}^{t_q} v_{i+\frac{1}{2}} &= n_{\mathrm{i}}^{t_q} v_{\mathrm{i}} + n_{\mathrm{i}}\frac{\partial v}{\partial x}\frac{\Delta x}{2}\\
n_{\mathrm{i}}^{t_p} v_{i-\frac{1}{2}} &= n_{\mathrm{i}}^{t_q} v_{\mathrm{i}} - n_{\mathrm{i}}^{t_q}\frac{\partial v}{\partial x}\frac{\Delta x}{2} - v_{\mathrm{i}}\frac{\partial n}{\partial x}\Delta x - v_{\mathrm{i}}\frac{\partial n}{\partial t}\ \ \Delta t_p - \tau)\\
n_{\mathrm{i}}^{t_{p+1}} v_{i-\frac{1}{2}} &= n_{\mathrm{i}}^{t_q} v_{\mathrm{i}} - n_{\mathrm{i}}^{t_q}\frac{\partial v}{\partial x}\frac{\Delta x}{2} - v_{\mathrm{i}}\frac{\partial n}{\partial x}\Delta x + v_{\mathrm{i}}\frac{\partial n}{\partial t}\tau\\
n_{\mathrm{i}}^{t_{q+1}} &= n_{\mathrm{i}}^{t_q} - \Delta t_q\left[\frac{\partial(nv)}{\partial x} + v_{\mathrm{i}}\frac{\partial n}{\partial t}\frac{\Delta t_p - \Delta t_q)}{\Delta t_q\Delta x}\tau\right]\\
\frac{n_{\mathrm{i}}^{t_{q+1}} - n_{\mathrm{i}}^{t_q}}{\Delta t_q} &+ \frac{\partial(nv)}{\partial x} = -v_{\mathrm{i}}\frac{\partial n}{\partial t}\frac{\Delta t_p - \Delta t_q)}{\Delta t_q\Delta x}\tau
\end{aligned} \tag{10}$$

Hence there is consistency since $\Delta t_p - \Delta t_q = \mathrm{O}(\Delta\ t_q\Delta x)$ and $\tau = \mathrm{O}(\Delta t_q)$.

- Second case: $t_p \leqslant t_q \leqslant t_{p+1} \leqslant t_{p+2} \leqslant t_{q+1}$

$$\begin{aligned}
n_{\mathrm{i}}^{t_{q+1}} &= n_{\mathrm{i}}^{t_q} - \frac{1}{\Delta x}\left[\Delta t_q n_{\mathrm{i}}^{t_q} v_{i+\frac{1}{2}} - \tau n_{i-1}^{t_p} v_{i-\frac{1}{2}} - \Delta t_p n_{i-1}^{t_{p+1}} v_{i-\frac{1}{2}} - \ \ \Delta t_q - \tau - \Delta t_p) n_{i-1}^{t_{p+2}} v_{i-\frac{1}{2}}\right]\\
n_{\mathrm{i}}^{t_{p+2}} v_{i-\frac{1}{2}} &= n_{\mathrm{i}}^{t_q} v_{\mathrm{i}} - n_{\mathrm{i}}^{t_q}\frac{\partial v}{\partial x}\frac{\Delta x}{2} - v_{\mathrm{i}}\frac{\partial n}{\partial x}\Delta x + v_{\mathrm{i}}\frac{\partial n}{\partial t}\ \ \tau + \Delta t_p)\\
n_{\mathrm{i}}^{t_{q+1}} &= n_{\mathrm{i}}^{t_q} - \Delta t_q\left[\frac{\partial(nv)}{\partial x} + \frac{v_{\mathrm{i}}}{\Delta t_q\Delta x}\frac{\partial n}{\partial t}\ \ \ \Delta t_p - \Delta t_q)\tau + \Delta t_p\tau + \Delta t_p\ \ \Delta t_p - \Delta t_q))\right]
\end{aligned} \tag{11}$$

In this case, necessary $\tau \leqslant \Delta t_q - \Delta t_p$ so $\tau = \mathrm{O}(\Delta t_q\Delta x)$, which gives consistency.

- Third case: $t_p \leqslant t_q \leqslant t_{q+1} \leqslant t_{p+1}$

$$\begin{aligned} n_{\mathrm{i}}^{t_{q+1}} &= n_{\mathrm{i}}^{t_q} - \frac{1}{\Delta x}\left[\Delta t_q n_{\mathrm{i}}^{t_q} v_{i+\frac{1}{2}} - \Delta t_q n_{i-1}^{t_p} v_{i-\frac{1}{2}}\right] \\ n_{\mathrm{i}}^{t_{q+1}} &= n_{\mathrm{i}}^{t_q} - \Delta t_q\left[\frac{\partial(nv)}{\partial x} + v_{\mathrm{i}}\frac{\partial n}{\partial t}\frac{\Delta t_p - \tau}{\Delta x}\right] \end{aligned} \tag{12}$$

In this case, necessary $\Delta t_q \leqslant \tau \leqslant \Delta t_p$ so $\Delta t_p - \tau = \mathrm{O}(\Delta t_q \Delta x)$, which gives consistency.

As a result the first-order asynchronous scheme is convergent. The local CFL condition hypothesis is essential for the convergence of the scheme.

2.7. Numerical diffusion

When writing second order Taylor development of the first-order asynchronous scheme and isolating the numerical diffusion coefficient, one obtains for case two:

$$D_{\mathrm{async}} = v^2\left(\frac{\Delta t_q}{2} + v\frac{\Delta t_p - \tau)^2\tau + \tau^2\ \Delta t_q - \tau)}{2\Delta t_q \Delta x} - \frac{\Delta t_p - \Delta t_q)\tau}{\Delta t_q} - \frac{\Delta x}{2v}\right) \tag{13}$$

The first and last terms also appear when calculating the numerical diffusion coefficient for the synchronous scheme giving rise to $D_{\mathrm{sync}} = \frac{v\Delta x}{2}(1 - \frac{v\Delta t}{\Delta x})$. In contrast for the asynchronous scheme those two terms cancel each other at least at first order in Δx when $k_{\mathrm{CFL}} = 1$. The third term is also second order in Δx, only the second term as to be evaluated knowing that to first order$\Delta t_q \simeq \Delta t_p$ and $(\Delta t_q - \tau)\tau \leqslant \frac{\Delta t_q^2}{4}$ with $0 \leqslant \tau \leqslant \Delta t_q$. Then the asynchronous diffusion coefficient is lower than:

$$D_{\mathrm{async}} \leqslant \frac{v\Delta x}{8} \tag{14}$$

This means that if the velocity variation over the mesh is greater than 25%, the asynchronous scheme becomes less diffusive than its synchronous equivalent. On a physical viewpoint the numerical diffusion in the synchronous scheme can be seen as a consequence of the fact that in the area where the CFL number is lower than one, information travels faster than it should. The asynchronous scheme cancels this effect to first order because information cannot go faster than the fastest local wave corresponding to the local CFL. However the asynchronous scheme introduces diffusion due to the local desynchronization τ (second term in Eq. (13)).

Notes:

- case one and three are corner cases. Nevertheless they lead to no numerical diffusion coefficient to first order in Δx.
- the estimation of the asynchronous numerical diffusion coefficient correspond to the worst case $\tau = \frac{\Delta t_q}{2}$. Actually τ keeps varying in time so the average value of numerical diffusion should be lower.

3. Test case: comparison with a classical synchronous method

A 1D advection equation is considered with periodic boundary conditions and unit advection velocity on a non regular mesh over an unit domain:

$$\frac{\partial n}{\partial t} + \frac{\partial n}{\partial x} = 0 \tag{15}$$

The N mesh point grid follows some polynomial law with parameter α defining the slope at $x = 0.5$

$$x_{\mathrm{i}} = (4 - 4\alpha)\left(\frac{i-1}{N-1} - \frac{1}{2}\right)^3 + \alpha\left(\frac{i-1}{N-1} - \frac{1}{2}\right) + \frac{1}{2} \tag{16}$$

Test function is defined at $t = 0$ as follows:

$$n(x) = \begin{cases} 1 & \text{if } |x - \frac{1}{2}| < \frac{1}{2} \\ 0.01 & \text{else} \end{cases} \tag{17}$$

A first-order in time second order in space (MUSCL, minmod limiter) scheme is being used. After 20 periods results from the asynchronous scheme are compared with results from a standard synchronous method which time step is equal to the global CFL condition. For the asynchronous scheme the local time step is equal to $\Delta t(x) = 0.49\frac{\Delta x}{v(x)}$, whereas the global time step is equal to $\Delta t = 0.49 \min\left\{\frac{\Delta x}{v(x)}\right._x$. Precision is evaluated with respect to the analytical solution $n_\mathrm{i}^\mathrm{ref}$ using the errors defined in (18) and (19).

$$E_1^\mathrm{async} = \frac{\sum((n_\mathrm{i}^\mathrm{async} - n_\mathrm{i}^\mathrm{ref})^2 \Delta x_\mathrm{i})}{\sum((n_\mathrm{i}^\mathrm{ref})^2 \Delta x_\mathrm{i})} \tag{18}$$

$$E_1^\mathrm{sync} = \frac{\sum((n_\mathrm{i}^\mathrm{sync} - n_\mathrm{i}^\mathrm{ref})^2 \Delta x_\mathrm{i})}{\sum((n_\mathrm{i}^\mathrm{ref})^2 \Delta x_\mathrm{i})} \tag{19}$$

Computation time performance is evaluated with the ratio R_cputime defined as: computation time of the synchronous scheme divided by computation time of the asynchronous scheme. The sensitivity to both α and cell number N_cell is presented on Tables 1 and 2. The maximum local space step variation $\max\ \frac{\delta\Delta x}{\Delta x}) = \max_\mathrm{i}\left(\frac{\Delta x_{i+1} - \Delta x_\mathrm{i}}{\Delta x_\mathrm{i}}\right)$ corresponding to the combination of α and N_cell is characteristic of the mesh distortion. Comparison is made between tree based asynchronous scheme, discrete time scheduled asynchronous scheme and synchronous scheme. As an illustrative example one test case is plotted on Fig. 4.

Table 1 shows that for a fully homogeneous problem the asynchronous scheme behavior is equivalent to the synchronous scheme but is slower by a factor of about 2.5 to 3.5 depending on sorting. For non uniform grids the synchronous error E_1^sync increases quite faster than the asynchronous error E_1^async. The asynchronous scheme is less diffusive numerically, when using discrete time scheduling numerical diffusion is hardly worsened. In the meantime the R_cpu ratio becomes more favorable to the asynchronous scheme. This means that for a given mesh non uniformity the asynchronous method becomes more time effective than the synchronous method. The asynchronous method reduces significantly numerical diffusion with respect to the synchronous scheme even when using second order flux discretization in space.

Table 2 shows that when refining the mesh, the CPU time gain is lowered when using tree-based sorting whereas it increases when using discrete time scheduling. Refining the mesh leads using the minimum time step on fewer cells, thus reducing computations. One expects that the R_cpu ratio should grow. However tree-based sorting cost increases in $\log(n)$ so the global balance is negative. To the contrary discrete time scheduling is not affected by cell number and R_cpu increases. The asynchronous method is most effective for locally sharp minimum of the CFL condition. In other words it is well adapted for localized phenomenon or locally refined grids and sharp gradient capture.

Table 1
$N_\mathrm{cell} = 200$

α	1	0.5	0.1	0.05	0.01
$\max(\frac{\delta\Delta x}{\Delta x})$	0%	1.7%	5.3%	7.9%	18.8%
E_1^async(tree)	2.58×10^{-3}	3.71×10^{-3}	4.75×10^{-3}	5.35×10^{-3}	5.20×10^{-3}
E_1^async(sched)	2.58×10^{-3}	4.67×10^{-3}	4.82×10^{-3}	5.39×10^{-3}	5.74×10^{-3}
E_1^sync	2.58×10^{-3}	1.54×10^{-2}	7.48×10^{-2}	8.39×10^{-2}	9.04×10^{-2}
$t_\mathrm{cpu}^\mathrm{async}$(tree), in s	0.891	1.094	2.203	3.079	6.656
$t_\mathrm{cpu}^\mathrm{async}$(sched), in s	0.609	0.859	1.703	2.391	5.453
$t_\mathrm{cpu}^\mathrm{sync}$, in s	0.265	0.406	1.641	3.187	15.360
R_cpu(tree)	0.297	0.371	0.745	1.035	2.308
R_cpu(sched)	0.435	0.473	0.964	1.333	2.817

Table 2
$\alpha = 0.01$

N_{cell}	200	500	1000	2000	5000
max $\frac{\delta\Delta x}{\Delta x}$)	18.8%	7.1%	3.5%	1.7%	0.7%
E_1^{async}(tree)	5.20×10^{-3}	2.19×10^{-3}	1.08×10^{-3}	4.97×10^{-4}	2.12×10^{-4}
E_1^{async}(sched)	5.74×10^{-3}	2.22×10^{-3}	9.79×10^{-4}	4.51×10^{-4}	1.92×10^{-4}
E_1^{sync}	9.04×10^{-2}	4.99×10^{-2}	3.15×10^{-2}	1.98×10^{-2}	1.06×10^{-2}
$t_{\text{cpu}}^{\text{async}}$(tree), in s	7	45	190	813	5497
$t_{\text{cpu}}^{\text{async}}$(sched), in s	5	33	131	536	3270
$t_{\text{cpu}}^{\text{sync}}$, in s	15	96	389	1570	9854
R_{cpu}(tree)	2.31	2.14	2.04	1.93	1.79
R_{cpu}(sched)	2.82	2.91	2.97	3.02	3.01

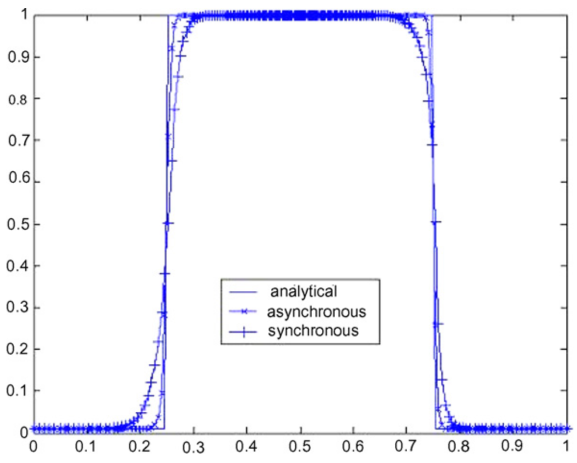

Fig. 4. Comparison after 20 periods for $\alpha = 0.01$ and $N_{\text{cell}} = 200$.

4. Application to gas discharge models

4.1. Model equations for a collisional discharge

A collisional discharge (i.e. where the charged particle mean free paths are much smaller than the discharge dimensions) is well described by the following fluid equations (e is for electrons, i for ions):

- Continuity equation

$$\frac{\partial n_e}{\partial t} + \vec{\nabla} \cdot \vec{\Gamma}_e = S \tag{20}$$

$$\frac{\partial n_i}{\partial t} + \vec{\nabla} \cdot \vec{\Gamma}_i = S \tag{21}$$

- Momentum equation in drift–diffusion form

$$\vec{\Gamma}_e = \mu_e(E)\left(-n_e\vec{E} - \frac{k_B T_e}{e}\vec{\nabla} n_e\right) \tag{22}$$

$$\vec{\Gamma}_i = \mu_i(E)\left(n_i\vec{E} - \frac{k_B T_i}{e}\vec{\nabla} n_i\right) \tag{23}$$

- Source terms

$$S = S_{\text{ionization}} + S_{\text{recombination}} \tag{24}$$

$$S_{\text{ionization}} = \alpha(\widetilde{E})\|\vec{\Gamma}_{\text{e}}\| \tag{25}$$

with $\widetilde{E} = \frac{-\vec{E}\cdot\vec{\Gamma}_{\text{e}}}{\|\vec{\Gamma}_{\text{e}}\|}$ if $-\vec{E}\cdot\vec{\Gamma}_{\text{e}} > 0$ and $\widetilde{E} = 0$ otherwise

$$S_{\text{recombination}} = -rn_{\text{e}}n_{\text{i}} \tag{26}$$

- Poisson's Equation

$$\vec{\nabla}\cdot(\epsilon\vec{E}) = \rho \tag{27}$$

In this approach the electron energy equation is replaced by the so-called Local Field Approximation (LFA). This supposes that the charged particle energy gain due to the local electric field is locally balanced by the losses due to collisions. The consequences of the LFA are that transport and reaction coefficients such as mobilities, ionization coefficients, etc at a given location and at a given time depend only on the electric field at that position and that time. The dependence of these coefficients on the electric field is therefore the same as under uniform field conditions, and is obtained by solving the steady state and homogeneous (no spatial gradients) Boltzmann equation.

4.2. Using the asynchronous scheme for gas discharge modeling

4.2.1. Coupling transport with Poisson's equation

Special care has to be taken in the treatment of the coupling between transport and Poisson's equation for the asynchronous scheme to be efficient in terms of computation time.

Numerical evaluation of the interface flux requires the knowledge of the local electric field to estimate the drift of the charged species at the precise time of the flux refresh. the most straight forward strategy for an explicit scheme would be to solve Poisson's equation at each refresh time tag. This means that if the mesh counts N cells, Poisson's Equation shall be solved N times more often than for the synchronous scheme. Because Poisson's equation is fundamentally non local, this would make the asynchronous method totally inefficient. The alternative is to solve Poisson's equation asynchronously with a refresh rate deduced from the total current conservation equation:

$$\vec{\nabla}\cdot\left(\epsilon_0\frac{\partial\vec{E}}{\partial t} + (n_{\text{e}}\mu_{\text{e}} + n_{\text{i}}\mu_{\text{i}})e\vec{E}\right) = 0 \tag{28}$$

The characteristic time appearing in this equation is the Maxwell time or dielectric relaxation time: $\tau_M = \frac{\epsilon_0}{(n_{\text{e}}\mu_{\text{e}}+n_{\text{i}}\mu_{\text{i}})e}$ this time scale is characteristic for the dynamic of both potential and current in the system. It is also a stability constraint on the time step for the explicit synchronous scheme.

Between two updates of the potential, the electric field values lag behind. Nevertheless this is sufficient to obtain a numerically stable solution of the problem. However the solution of a stationary problem may become "noisy" because the aging of the electric field makes the flux react with delay leading to oscillations. Ionization rate may also be over-estimated if the source term is based on an old electric field value, this may lead to large current errors. A simple numeric filtering technique based on physical considerations has been developed to estimate the local electric field variation rate $\frac{\partial\vec{E}}{\partial t}$. This estimate is then used to propagate the electric field.

The idea is to linearize the total current equation locally on a short time scale (Δt) at an equilibrium state around which the current can be considered constant in time. It makes sense because solving Poisson's equation reveals any global or "long term" variation of the total current. This time scale has obviously to be lower than the dielectric relaxation time.

$$\vec{J}^{i,j}_{\text{total}}(t) = \vec{J}^{i,j}_{\text{total}}(t-\Delta t) \tag{29}$$

$$\vec{J}^{i,j}_{\text{conduction}}(t) + \vec{J}^{i,j}_{\text{displacement}}(t) = \vec{J}^{i,j}_{\text{conduction}}(t-\Delta t) + \vec{J}^{i,j}_{\text{displacement}}(t-\Delta t) \tag{30}$$

A single flux refresh will lead to a small conduction current adjustment $\delta J_{\text{conduction}}$. If the total current is conserved, this small conduction current variation will be balanced by a small numerical displacement current $\delta J_{\text{displacement}}$ in the opposite direction. The local electric field variation rate is deduced from this small numerical displacement current.

$$\frac{\partial \vec{E}^{i,j}}{\partial t}(t) = \frac{\left[(n_{\text{e}}^{i,j}\mu_{\text{e}}^{i,j} + n_{\text{ion}}^{i,j}\mu_{\text{ion}}^{i,j})e\vec{E}^{i,j}\right]_{t-\Delta t} - \left[(n_{\text{e}}^{i,j}\mu_{\text{e}}^{i,j} + n_{\text{ion}}^{i,j}\mu_{\text{ion}}^{i,j})e\vec{E}^{i,j}\right]_{t}}{\epsilon_0} \tag{31}$$

Note: in 1D the total current equation can be analytically solved see Appendix.

4.2.2. Local CFL condition for charged particles flux

For simplicity the local CFL condition is described on a regular mesh, but it can be easily generalized to any mesh. In fact the local CFL condition has three contributions, the drift velocity $|\mu(E)E|$, the "diffusion velocity" $\frac{2D(E)}{\Delta x}$, and the potential variation velocity linked to the local Maxwell time $\frac{(n_{\text{e}}\mu_{\text{e}}+n_{\text{i}}\mu_{\text{i}})e}{\epsilon_0}\Delta x$. For a purely 1D problem, one obtains the following CFL condition:

$$\Delta t = k_{\text{CFL}} \frac{\Delta x}{|\mu(E)E| + \frac{2D(E)}{\Delta x} + \frac{(n_{\text{e}}\mu_{\text{e}}+n_{\text{i}}\mu_{\text{i}})e}{\epsilon_0}\Delta x} \tag{32}$$

with k_{CFL} a stability margin ($\text{k}_{\text{CFL}} = 0.99$ for first-order scheme or $k_{\text{CFL}} = 0.49$ for MUSCL scheme).

This expression is numerically quite different for the electrons and the ions (electron mobilities are usually at least 100 times larger than the corresponding ion mobilities). This means that a lot of computation time can be saved by using different time steps for the various species whereas in synchronous methods the ions flux refresh rate are defined by the electrons dynamics. In low charge density areas, time steps of the various species are decoupled, whereas in high charge density areas (plasma) these time steps tend to become independent of the species. This is a consequence of the coupling of the different species through the electric field within the plasma so the time steps must adapt to properly integrate the fast variations of the electric field within this region even for the slow species.

In 2D the flux refresh time step has to insure that the CFL condition is not violated, which means that information within the mesh propagates as fast as the fastest physical wave in the system in any direction. This is done by defining: $v_x = |\mu(E)E_x| + \frac{2D(E)}{\Delta x} + \frac{(n_{\text{e}}\mu_{\text{e}}+n_{\text{i}}\mu_{\text{i}})e}{\epsilon_0}\Delta x$ and $v_y = |\mu(E)E_y| + \frac{2D(E)}{\Delta x} + \frac{(n_{\text{e}}\mu_{\text{e}}+n_{\text{i}}\mu_{\text{i}})e}{\epsilon_0}\Delta y$ the information velocities in x and y directions. In particular the CFL condition has to be respected diagonally to the mesh, which means that the refresh rate of a flux has to take into account the most demanding value of the four surrounding velocities in the orthogonal direction.

For instance in the x direction

$$\Delta t_x = \frac{\Delta x \Delta y}{v_x \Delta y + v_\perp \Delta x} \tag{33}$$

The orthogonal velocity may be chosen as the maximum of the four surrounding values:

$$v_\perp = \max(v_y^{x+,y+}, v_y^{x+,y-}, v_y^{x-,y+}, v_y^{x-,y-}) \tag{34}$$

However it is slightly more effective to take into account the direction of the updated flux. Hence only the maximum orthogonal velocity leaving the upwind cell as to be considered.

4.2.3. Source term time stepping

The refresh rate of each reaction has to be chosen in order to follow this reaction dynamic.

- Ionization: $S_{\text{ionization}} = \alpha(\widetilde{E})\|\vec{\Gamma}_{\text{e}}\|$
  - it depends on the electron flux, so it has to react at least as fast as the electron flux.$\Delta t = \frac{\Delta l}{|\mu(E)E| + \frac{2D(E)}{\Delta l} + \frac{(n_{\text{e}}\mu_{\text{e}}+n_{\text{i}}\mu_{\text{i}})e}{\epsilon_0}\Delta l}$ with $\Delta l = \sqrt{\Delta x^2 + \Delta y^2}$
  - it leads to an exponential growth of density with a characteristic time $\Delta t = \frac{1}{\alpha\|\mu E\|}$. For this exponential to be integrated not too coarsely, time step should be a fraction of this time. for instance: $\Delta t = \frac{0.1}{\alpha\|\mu E\|}$

  The ionization time step shall be the minimum of these two conditions

- Recombination: $S_{\text{recombination}} = -rn_en_i$
  - For cell k, the recombination refresh shall be less than $\Delta t_k = \frac{0.1}{\max(n_e^k, n_i^k)}$ for precision with a maximum value by default in the very low density areas.
  - The recombination source term shall "see" the density dynamic due to ionization This can be done by recomputing the recombination source term when ionization source term is refreshed.

4.2.4. Time steps limitation on local density variation

An additional constraint has been added to the time steps of both fluxes and source terms. For precision purpose the local variation of any density should not exceed 10% during any time step.

4.2.5. Asynchronous algorithm for gas discharge

4.2.5.1. Initialization.

(1) Initialize all the density and fluxes.
(2) Initialize all the refresh time tags to the initial time.
(3) Compute the initial values of potential and electric field.
(4) Interpolate the transport coefficients at the initial time.

4.2.5.2. Proceed until the simulation time is completed.

(1) Find the most urgent flux or source term to be refreshed.
(2) $t_{\text{simulation}}$ becomes this most urgent refresh time tag.
(3) For a flux
    (a) update neighboring density of all species and the E field;
    (b) interpolate the transport coefficient;
    (c) store the previous conduction current;
    (d) compute the new value of the flux (MUSCL for drift plus diffusion);
    (e) compute the new conduction current;
    (f) update the E field variation rate with the conduction current variation;
    (g) update the variation rate of density;
    (h) compute the local time step;
    (i) replace the next refresh time tag into the tree or scheduler;
    (j) update Maxwell time if required;
    (k) update Poisson's equation if required;
  For source term
    (a) update the local density of all species and the E field;
    (b) interpolate the reaction coefficient;
    (c) compute the new value of the source term;
    (d) compute the local time step;
    (e) replace the next refresh time tag into the tree or scheduler;
(4) If the next output time tag is lower than the next most urgent refresh time tag, build the solution at this output time tag and store it.

4.3. Application to surface dielectric barrier discharge (DBD) at atmospheric pressure

4.3.1. Model description

Fig. 5 shows a typical DBD configuration that has been suggested for aerodynamic flow control. In a surface DBD plasma actuator, a sinusoidal voltage is applied between the electrodes. Transient discharges develop above the dielectric surface and momentum transfer between charged particles and neutral molecules can generate a flow or modify the boundary layer of a flow along an airfoil (see [7–10]).

In this paper we consider a simple numerical experiment (as in [9]) in the 2D Cartesian geometry of Fig. 5 with a constant applied voltage between the electrodes. The electrode above the dielectric surface is the anode. This numerical experiment is used to compare the asynchronous method with classical methods in the same

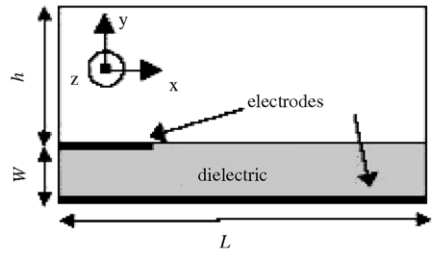

Fig. 5. Simulation domain for surface dielectric barrier discharge for flow control.

conditions. The gas is pure Nitrogen at atmospheric pressure, and the electron and ion transport coefficients are the same as in [9]. We assume that secondary electrons are generated by the dielectric surface under ion impact, and that the charged particles charge the dielectric surface or recombine instantaneously, i.e. the surface charge σ on the dielectric is given by:

$$\sigma = \int_0^t e(\Gamma_i^\perp - \Gamma_e^\perp)\mathrm{d}t$$

The secondary electron coefficient, i.e. the number of electrons emitted by the dielectric surface for each ion hitting the surface is supposed to be equal to 0.05 in the simulations. Secondary electron emission from the surface plays an important role in this model and is responsible (as well as electron impact ionization of the nitrogen molecule above the surface) for the maintenance and propagation of the discharge along the surface. The conditions of the simulation are indicated below.

Simulation parameters:

- Gas: Nitrogen.
- Grid: 200×100, Cartesian geometry.
- Size: $h = 150$ μm $w = 50$ μm $L = 400$ μm.
- Dielectric relative permittivity: $\epsilon_r = 10$.
- Electron temperature: $T_e = 10^4$ K, ion temperature: $T_i = 350$ K.
- Thermal velocity: $v_{th}^e = \sqrt{\frac{8k_BT_e}{\Pi m_e}}$ for electron and $v_{th}^i = \sqrt{\frac{8k_BT_i}{\Pi m_i}}$.
- Flux scheme: Muscl (minmod limiter) for drift plus diffusion.
- Simulated time: 150 ns.

Boundary conditions:

- E field boundary left, right up: $E_\perp = 0$.
- E field boundary dielectric: $\epsilon_0 E_\perp - \epsilon_r \epsilon_0 E_\perp^{\text{dielectric}} = \sigma$.
- Transport boundary left, right: $\Gamma_\perp = 0$.
- Transport boundary up: $\max(0, \Gamma_\perp)$.
- Transport boundary dielectric electrons: $\Gamma_\perp^e = \min(0, \mu_e E_\perp - \frac{1}{4} v_{th}^e) + \gamma max(0, -\Gamma_i)$.
- Secondary emission coefficient: $\gamma = 0.05$.
- Transport boundary dielectric ions: $\Gamma_\perp^i = \min(0, \mu_i E_\perp - \frac{1}{4} v_{th}^i)$.
- Upper Electrode potential: $V_{\text{anode}} = 1200$ V.
- Lower Electrode potential: $V_{\text{cathode}} = 0$ V.

Low current or corona discharge phase:

For better current computation during transient phase such as sheath formation, Poisson's equation has to be solved according to an additional time constraint. This is required to correctly describe the steep current rise phase. During these phases plasma has not formed yet so the electron density is low, only the ions density is high. This means that Maxwell time is large, however there is already a distortion of the potential due to the space charge or dielectric charging. A proper refresh constraint is to say that the variation of the space charge created electric field remains low with respect to the geometrical electric field. However this constraint is unnecessary to insure the simulation stability. Consequently Poisson's equation is solved periodically according to Eq. (35) with V, the electrode voltage, L, a typical length of the system and k, a constant adjusted so that Maxwell time is more restrictive in the plasma phase. An iterative conjugate gradient solver is used to compute the potential.

$$\Delta t_{\text{Poisson}} = \min\left(\tau_M, k\frac{V\epsilon_0}{L^2\frac{\partial\rho}{\partial t}}\right) \tag{35}$$

4.3.2. Simulation results

The results of the simulation are displayed in Figs. 6–9.

The synchronous and asynchronous methods give very close results. We see in Figs. 6–8 that, because the applied voltage is larger than the breakdown voltage, a plasma forms at the tip of the upper electrode, where the electric field is larger. The electric field lines push the ions along the dielectric surface and toward the surface. The field in the plasma drops and the field at the right end of the plasma is enhanced. At the right end of the plasma an ion sheath forms, where ions are accelerated along and toward the surface, and generate secondary electrons when they impact on the surface. This mechanism makes the plasma propagate along the surface, until the potential drop along the plasma become so large that the potential drop across the sheath

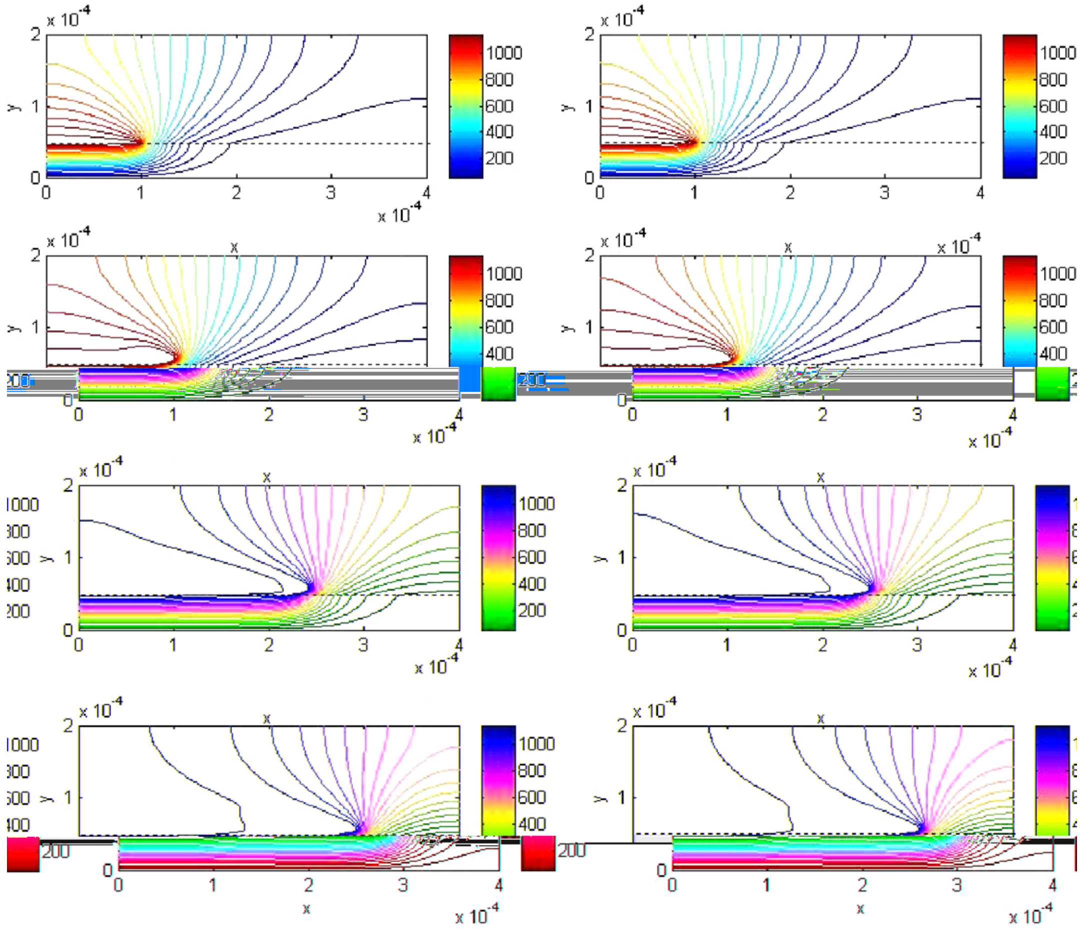

Fig. 6. Comparison of equipotential lines for the asynchronous scheme (left) and the standard synchronous scheme (right) at 1.5, 15, 45 and 60 ns.

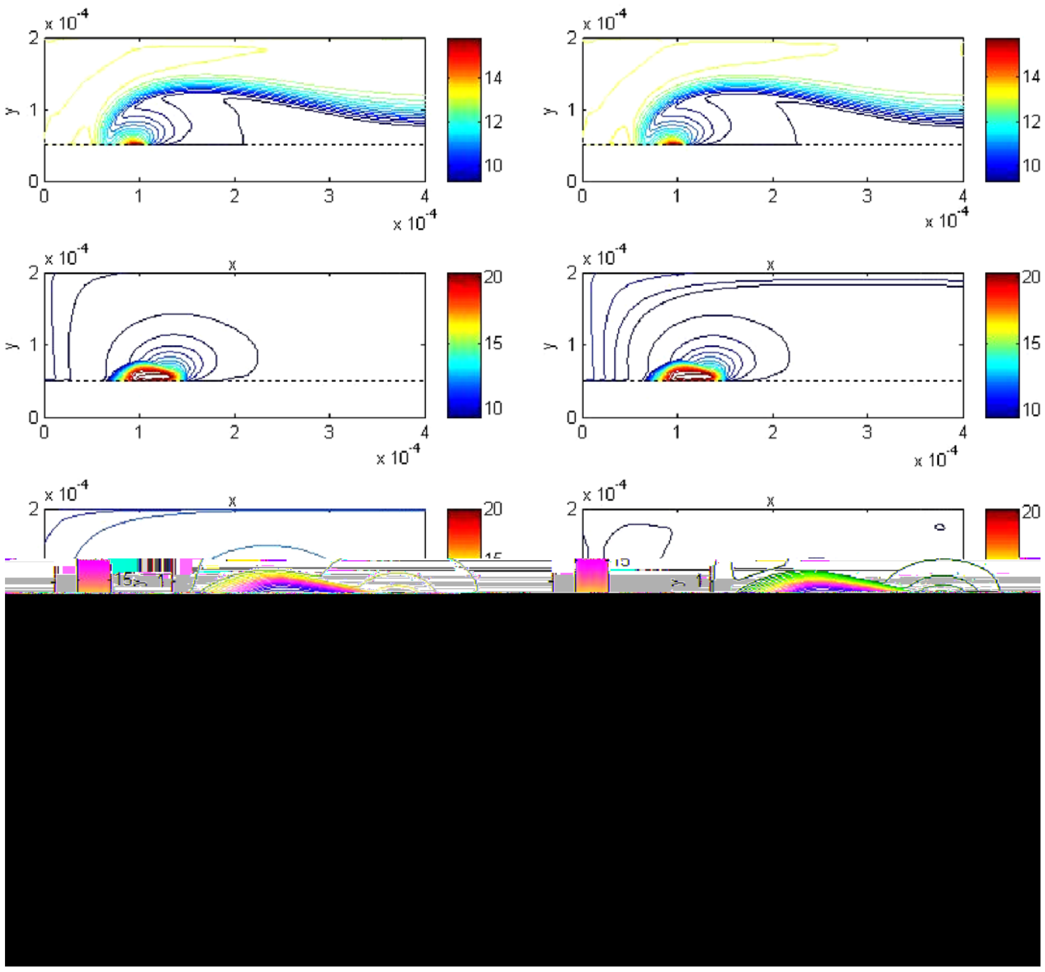

Fig. 7. Comparison of electron density ($\log(m^{-3})$) for the asynchronous scheme (left) and the standard synchronous scheme (right) at 1.5, 15, 45 and 60 ns.

is no longer sufficient to sustain the sheath. The velocity of the propagation along the sheath depends on several parameters such as: applied voltage, ionization coefficient, secondary electron emission coefficient and ion mobility.

Fig. 6 shows the sheath region propagation (the zone where the potential drop is concentrated). The sheath propagation is slightly slower for the asynchronous scheme than for the synchronous scheme. Excepted for the small propagation velocity difference, the density levels for both electrons and ions (see Figs. 7 and 8) agree. Moreover Fig. 9 shows that both discharges fit "macroscopically" because the total current collected at the lower electrode differs less than 5% between both methods. This is indeed a very good fit knowing that ionization is an exponential phenomenon. In comparison the current computed with the Scharfetter and Gummel semi-implicit scheme also shown on Fig. 9 is quite different. Tree-based and discrete time scheduling asynchronous results where not plotted separately on Fig. 9 because the difference lays within current noise.

As already mentioned, the results obtained with the asynchronous scheme are well in line with those obtained with the synchronous method. Still the slight difference might be explained by a larger numerical diffusion for the synchronous method. In fact larger numerical diffusion for ions would lead to a faster ion transport to the surface, thus creating more electrons by secondary emission. This leads to a faster growth of the plasma and a faster discharge development which also means a bigger current.

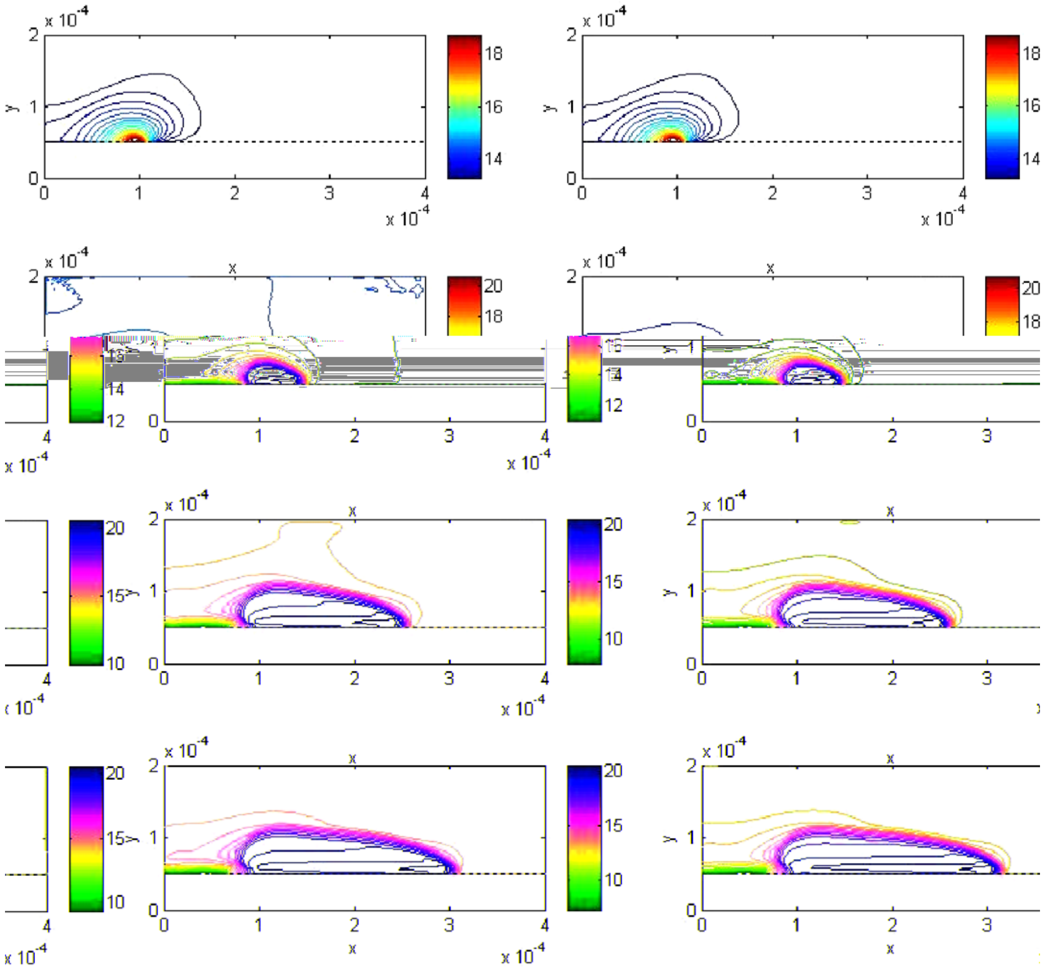

Fig. 8. Comparison of ion density (log(m^{-3})) for the asynchronous scheme (left) and the standard synchronous scheme (right) at 1.5, 15, 45 and 60 ns.

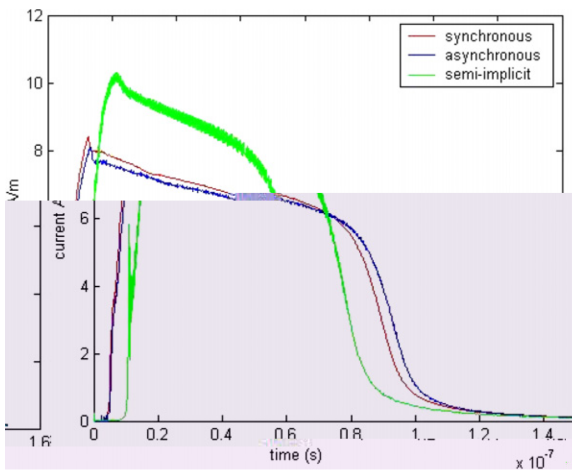

Fig. 9. Discharge current vs simulated time.

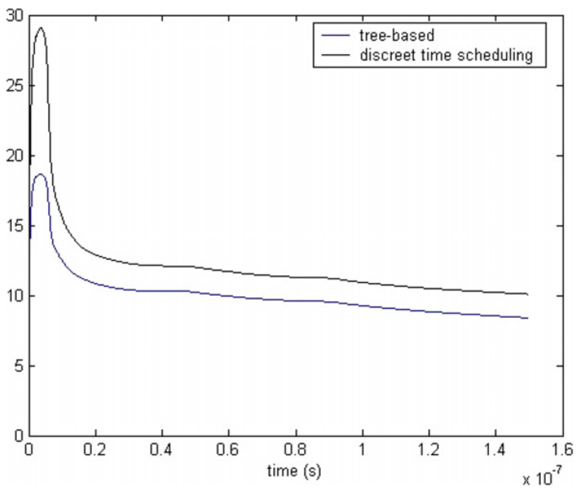

Fig. 10. R_{cputime} vs simulated time for tree-based sorting and discrete time scheduling with the asynchronous scheme.

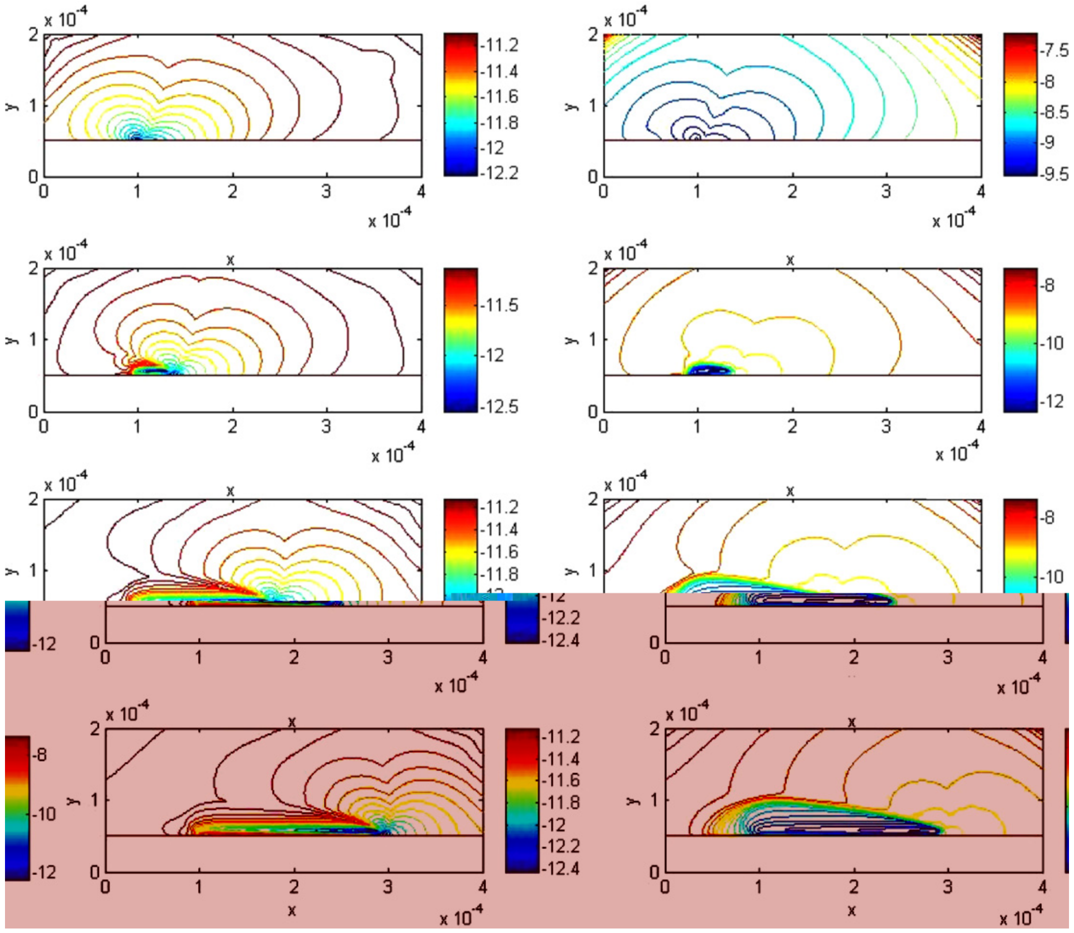

Fig. 11. Time steps (in log(s)) due to CFL condition for electrons (left) and ions (right).

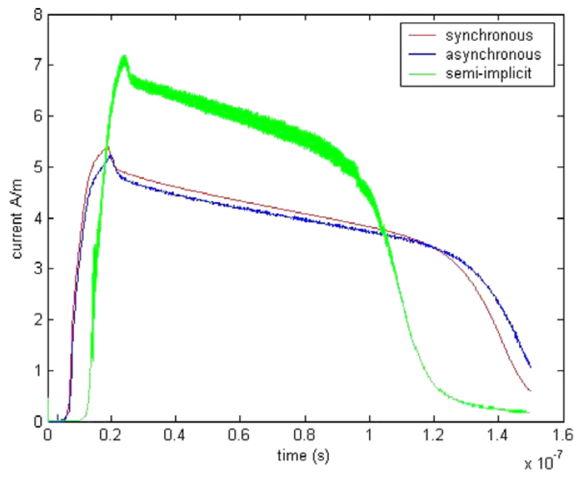

Fig. 12. Discharge current vs simulated time (grid 400 × 200).

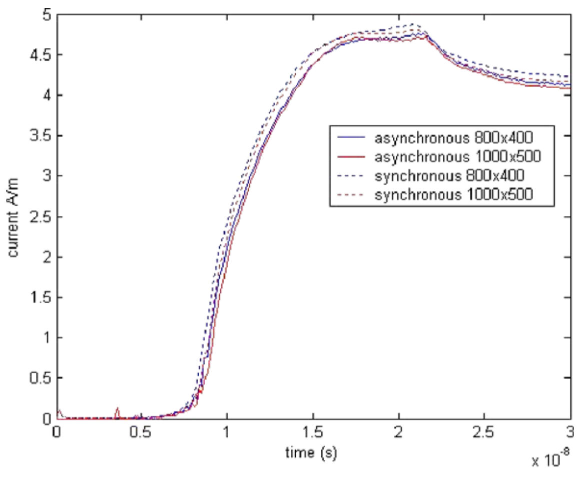

Fig. 13. Discharge current vs simulated time near mesh convergence.

The asynchronous method reduced computation time by more than 10, which is an impressive speed up. Fig. 11 shows that this performance is largely due to the bigger time steps used for ion transport. One can expect that the speed up should be even larger for air plasma simulation where the number of slow species (ions, metastable) is higher. Fig. 10 shows that speed up is maximal in the very first instants because the Maxwell time is still low so that Poisson's equation is not resolved very often. Most of the CPU time is then used for solving the transport equations. When plasma forms Poisson's equation solving takes important CPU time, this takes as much CPU time for synchronous and asynchronous methods. That is why R_{cputime} drops after the very beginning then reaches a plateau phase during plasma propagation. After 100 ns the discharge has reached the right end of the simulation domain. At that time the dielectric surface has been totally charged and there is no more potential difference in the fluid domain. Then the plasma evolution is dominated by recombination and ambipolar diffusion. During this phase the minimum of the CFL condition is less sharp and localized than during the discharge that is why R_{cputime} decreases again after 100 ns. Only R_{cputime} order of magnitude is relevant. Its value is sensitive to various parameters such as sharpness of the CFL minimum, Poisson's solver convergence criterion, code optimization, CPU vs memory access performance, etc...

grid	200x100	400x200	800x400	1000x500
Δ_x(m)	2×10^{-6}	1×10^{-6}	5.0×10^{-7}	2.5×10^{-7}
λ_D(m)	2.5×10^{-7}	3.3×10^{-7}	4.0×10^{-7}	3.7×10^{-7}

Fig. 14. Grid spacing vs Debye's length for various grids.

4.3.3. Mesh convergence

When refining the mesh, the discharge current tends to decrease until convergence is reached around 4.5 A/m peak current as shown on Figs. 12 and 13. The asynchronous scheme performs slightly better than the synchronous scheme because it seems to reach convergence a little faster. This is probably due to a lower numerical diffusion in ion transport in particular towards the dielectric surface which means fewer secondary electron to sustain the plasma. The key parameter of this problem is electron multiplication within the cathode sheath. Usually the cathode sheath has the size of about 10 Debye's length. Fig. 14 shows that convergence is reached when the grid spacing is about Debye's length, this means that at least 10 cells are needed to correctly describe the electron multiplication within the cathode sheath in such a discharge.

5. Conclusion

The present asynchronous scheme has proven to hold good numerical properties especially in terms of numerical diffusion, theoretically with first order space discretization in Section 2.7 and numerically with second order space discretization in Section 3. The scheme has been tested under various meshes (homogeneous or inhomogeneous refinements) and is more accurate in these conditions than the classical synchronous scheme. In terms of computation time, the asynchronous scheme is most time saving for locally sharp minimum of the CFL condition. This technique is particularly adapted to multi-scale multi-species transport problems and leads to tremendous speed-up in calculation for inhomogeneous media in terms of CFL condition. In the case of DBD for aerodynamic flow control an impressive computation time gain of about 10 has been achieved. For many applications of atmospheric plasmas this method is especially effective for three reasons. First the electric field exhibits steep gradient. Second lot of CPU time is saved by using specific time stepping for the heavy particles. Third in the case of surface DBD the plasma is quite localized near the dielectric. This technique is promising for various transport problems provided there are inhomogeneous in terms of CFL condition. This applies to steep localized gradient but can also applied to locally refined meshes. In the field of gas discharge modeling this new tool is well adapted for dielectric barrier discharge and corona discharges. The method could also be used for 3D simulations.

Acknowledgments

This work was partially supported by the EADS foundation (One of the authors Thomas Unfer is supported by a fellowship of the EADS foundation).

Appendix. Poisson/asynchronous transport coupling in 1D

In 1D the total current density is constant in the Ox direction over the domain and the current equation can be analytically integrated and discretized (see [6]):

$$\epsilon_0 \frac{\partial[V_{\text{cathode}} - V_{\text{anode}}]}{\partial t} + \sum (n_e\mu_e + n_i\mu_i)e\vec{E} \cdot \Delta x\vec{u_x} = J_{\text{total}}L \tag{36}$$

The total current density is then the mean value over the mesh of the conduction current plus the injected displacement current in the electrodes. This mean value can be updated "locally" because only one flux has changed from the previous value, consequently it is a O(1) operation. Thus, the local variation rate of the electric field for cell k can be deduced from the total current density at a low computational cost:

$$\frac{\partial E_k}{\partial t} = \frac{J_{\text{total}} - [(n_e\mu_e + n_i\mu_i)eE]_k}{\epsilon_0} \tag{37}$$

After solving Poisson's equation initially to obtain the electric field, it can be propagated locally by direct integration. However this process does not insure that the computed electric field remains rotational free and is also integrating round-off errors. Consequently after a while the computed electric field diverges slowly from the actual solution of Poisson's equation. Practically it seems sufficient to solve again Poisson's equation every 100 Maxwell time to filter off this error.

References

[1] M. Berger, J. Oliger, Adaptive mesh refinement for hyperbolic partial differential equations, J. Comput. Phys. 53 (1984) 484512.

[2] X. Lu, R. Schmid, Symplectic discretisation for Maxwell's Equations, J. Math Comput. 25 (2001).

[3] Y.A. Omelchenko, H. Karimabadi, Self-adaptive time integration of flux-conservative equations with sources, J. Comput. Phys. 216 (1) (2006) 179–194.

[4] C. Dawson, R. Kirby, High resolution schemes for conservation laws with locally varying time steps, Siam J. Sci. Comput. 22 (6) (2001) 2256–2281.

[5] H. Karimabadi, J. Driscoll, Y.A. Omelchenko, N. Omidi, A new asynchronous methodology for modeling of physical systems: breaking the curse of courant condition, J. Comput. Phys. 205 (2005) 755775.

[6] A.A. Kulikovsky, The structure of streamers in $N_2$1: fast method of space-charge dominated plasma simulation, J. Phys. D: Appl. Phys. 27 (1994) 2556–2563.

[7] J.R. Roth, Aerodynamic flow acceleration using paraelectric and peristaltic electrohydrodynamic effects of a One Atmosphere Uniform Glow Discharge Plasma, Phys. Plasmas 10 (2003) 2117–2126.

[8] J. Pons, E. Moreau, G. Touchard, Asymmetric surface dielectric barrier discharge in air at atmospheric pressure: electrical properties and induced airflow characteristics, J. Phys. D: Appl. Phys. 38 (2005) 3635–3642.

[9] J.P. Boeuf, L.C. Pitchford, Electrohydrodynamic force and aerodynamic flow acceleration in surface dielectric barrier discharge, J. Appl. Phys. 97 (103307) (2005) 1–9.

[10] J.P. Boeuf, Y. Lagmich, Th. Unfer, Th. Callegari, L.C. Pitchford, Electrohydrodynamic force in dielectric barrier discharge plasma actuators, J. Phys. D: Appl. Phys. 40 (2007) 652–662.

[11] G.J.M. Hagelaar, Modeling of microdischarges for display technology PhD Thesis Technische Universiteit Eindhoven, The Netherlands, 2000.