Introducing Merlino2D, an open-source fluid code for gas discharge simulations

F. Ragazzi[1], A. Popoli[1], G. Mongaretto[1], L.L. Alves[2], A. Cristofolini[1]

[1] *Department of Electrical, Electronic and Information Engineering "Guglielmo Marconi", University of Bologna, Bologna, Italy*

[2] *Instituto de Plasmas e Fusão Nuclear, Instituto Superior Tecnico, Av Rovisco Pais, 1049-001, Lisbon, Lisboa, Portugal*

Introduction

Numerical simulations of low-temperature plasma discharges are essential for the design and optimization of plasma-based devices and for understanding the physical mechanisms governing discharge behavior. However, these simulations are numerically challenging due to the strong nonlinear coupling between plasma chemistry and electrostatic interactions, as well as the pronounced stiffness arising from the wide range of characteristic time scales involved [1].

We introduce Merlino2D, an open-source MATLAB code for the simulation of low-temperature gas discharges and plasma-based devices. The code operates on unstructured triangular meshes and supports robust time-dependent simulations in two-dimensional domains. It also provides a native interface to the open-source LisbOn KInetics Boltzmann solver (LoKI-B) [2, 3] for the computation of electron energy distribution functions and swarm parameters.

Physical and Numerical Model

The code implements a fluid model. For each s species the continuity equation

$$\frac{\partial n_s}{\partial t} + \nabla \cdot \Gamma_s = \Omega_s \tag{1}$$

is solved. Photoionization source term can also be considered for discharges in air, following the model described in [4]. The fluxes are computed considering the drift diffusion approximation: $\Gamma_s = -D\nabla n_s + \text{sign}(q_s)\,\mu_s n_s E$. Eq. (1) is coupled with the Poisson equation

$$\nabla \cdot (\varepsilon_r E) = \frac{\rho}{\varepsilon_0}. \tag{2}$$

The contribution of surface charge density σ to the electric field can also be considered when simulating dielectric barrier discharges. The continuity equation is discretized according to the finite volume method. For the estimation of the fluxes, the number density gradient is approximated with first order accuracy, and the drift contribution is computed with a first-order upwind scheme. The Poisson equation is discretized according to the finite element method with linear shape functions.

Time Discretization – Comparison Between ODE and DAE Formulations

Historically, plasma simulation codes have relied on fully explicit or semi-explicit time integration schemes. Although these approaches are generally robust, they impose severe restrictions on the allowable time-step size due to stability constraints, such as the Courant–Friedrichs–Lewy (CFL) condition and the Maxwell relaxation time. As a consequence, simulations often require a very large number of time steps, leading to substantial computational costs, particularly when fine spatial discretizations are employed. Fully implicit schemes, in contrast, are not subject to these stability limitations and therefore permit significantly larger time steps. However, their usage has traditionally been limited by unfavorable computational scaling with the number of degrees of freedom and by their high memory requirements.

These limitations can be understood by considering a formulation of the problem as a system of ordinary differential equations (ODEs), similarly to what was done in [5]:

$$\frac{\mathrm{d}\{s\}}{\mathrm{d}t} = f\left(t, \{s\}\right), \tag{3}$$

where $\{s\}$ denotes the state vector containing all problem degrees of freedom and f is a nonlinear function describing the underlying physical processes. In the present case, the state vector consists of the number densities of all species in each computational cell. The application of a fully implicit time integration scheme requires the solution of a nonlinear system at each time step, typically through a Newton method involving the Jacobian matrix $\partial f/\partial s$.

Within the ODE formulation, the electric field (that appears as part of the nonlinear function f) depends implicitly on the number densities of all charged species throughout the computational domain through the solution of the Poisson equation. Consequently, the Jacobian matrix contains large dense blocks, resulting in substantial memory consumption and significantly increasing the computational cost of matrix factorizations.

To alleviate these limitations, Merlino2D formulates the problem as a system of differential-algebraic equations (DAEs) [6],

$$\begin{cases} \dfrac{\mathrm{d}\{n\}}{\mathrm{d}t} = f\left(t, \{n\}, \{\varphi\}\right), \\ \qquad 0 = g\left(t, \{n\}, \{\varphi\}\right), \end{cases} \tag{4}$$

where the differential equations describe the temporal evolution of the species number densities and the algebraic constraint is given by the Poisson equation. This system can be written in the compact form

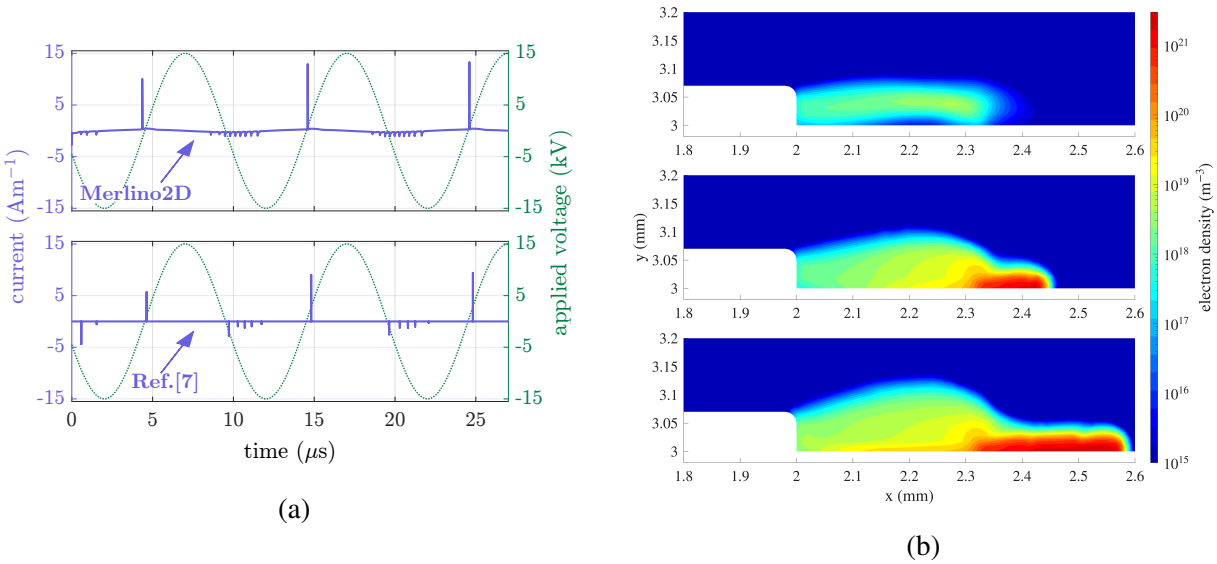
$$[\mathrm{M}]\,\frac{\mathrm{d}\{s\}}{\mathrm{d}t} = f\left(t, \{s\}\right), \tag{5}$$

where [M] is a singular mass matrix and the state vector contains both the species number densities in each domain cell and the electric potential values at mesh nodes.

In the DAE formulation, the electric potential is treated as an independent component of the state vector rather than being implicitly embedded in the nonlinear function. As a result, the electric field depends only on a subset of the state variables, yielding a substantially sparser Jacobian matrix than in the ODE formulation. This sparsity significantly reduces memory requirements and accelerates the solution of linear systems arising during Newton iterations.

By combining the DAE formulation with a suitable numerical library, Merlino2D integrates Eq. (5) with a fully implicit scheme with adaptive time-step control, enabling for the efficient treatment of the strong numerical stiffness characteristic of plasma discharge simulations while maintaining favorable computational performance as the mesh resolution increases.

Results



(a) (b)

Figure 1: sDBD results.

The dielectric barrier discharge simulation presented in [7] is used as a reference case to illustrate the capabilities of Merlino2D. The considered geometry consists of a 3 mm thick dielectric layer ($\varepsilon_r = 3.2$) with an HV electrode (70 µm x 2 mm) on top and a grounded electrode with length of 5 mm on the bottom. A sinusoidal voltage with amplitude of 15 kV and frequency of 100 kHz is applied to the HV electrode. A constant secondary electron emission coefficient $\gamma_{see} = 5 \times 10^{-2}$ is considered at the HV electrode, while at the dielectric interface the coefficient is $\gamma_{see_{diel}} = 1 \times 10^{-2}$. Fig. 1a presents a comparison between the current computed with Merlino2D and the one obtained in [7], showing a good qualitative agreement. In both cases, a positive current peak is observed shortly before the applied voltage changes polarity from

negative to positive. Conversely, the negative-voltage phase is characterized by a higher pulse frequency and lower pulse amplitudes. Fig. 1b shows the electron number density during the inception and evolution of a positive streamer that originates shortly before the change in voltage polarity. It is possible to see the electron tail attached to the HV electrode that reaches the dielectric interface and then propagates with a speed of approximately $2 \times 10^5\,\mathrm{m\,s^{-1}}$.

Conclusion

This work presented Merlino2D, an open-source MATLAB code for low-temperature plasma simulations integrated with LoKI-B and available at the link `https://github.com/PTL-Unibo/Merlino2D`. The code solves the drift–diffusion–Poisson system on unstructured 2D meshes using a fully implicit DAE formulation with adaptive time stepping. The resulting approach enables efficient simulation of DC and AC discharges while maintaining favorable scaling with mesh size.

Acknowledgements

Funded by the European Union

Funded by the European Union Grant Agreement No 101098900. Views and opinions expressed are however those of the author(s) only and do not necessarily reflect those of the European Union or EISMEA. Neither the European Union nor the granting authority can be held responsible for them.

IPFN activities were supported by FCT - Fundação para a Ciência e Tecnologia, I.P. by project reference UID/50010/2025, by project reference UID/PRR/50010/2025, by project reference UID/PRR2/50010/2025 and by project reference LA/P/0061/2020.

References

[1] A. Popoli et al., Plasma, **6**, 393-407, (2023)

[2] A. Tejero-del-Caz et al., Plasma Sources Sci. Technol. **28** 043001 (2019)

[3] A. Tejero-del-Caz et al., Plasma Sources Sci. Technol. **30** 065008 (2021)

[4] A. Bourdon et al., Plasma Sources Sci. Technol. **16** 656 (2007)

[5] F. Ragazzi et al., IEEE Access, **12**, 12545–61, (2024)

[6] L. F. Shampine et al., SIAM Review **41** 538-552 (1999)

[7] K. Kourtzanidis et al., J. Phys. D: Appl. Phys. **54** 045203 (2021)