Development of a Parallelised, Adaptive Mesh Drift-Diffusion Solver in FEniCS for the Modelling of Streamer Discharges in Gas-Solid Topologies

T. Wong*, I. Timoshkin, S. MacGregor, M. Wilson and M. Given
Electronic and Electrical Engineering
The University of Strathclyde, Glasgow
*timothy.wong@strath.ac.uk

IEEE Pulsed Power
Conference 2021

University of Strathclyde Glasgow

Research
Summary

Streamer discharges are fast ionisation fronts that are of crucial importance to many academic and industrial fields, including:

Understanding of geophysical processes
Gaseous breakdown
Surface treatment technology
Bio-decontamination
Air Cleaning
Pulsed power system design

The growing capabilities of modern computers allow streamers to be simulated on desktop computers. However, it is far from a trivial task. In this work, we demonstrate a parallelised, adaptive-mesh enabled code developed using the open-source finite element platform FeniCS in Python. We show that the platform is capable of simulating streamer propagation in pure gas and across gas-solid interfaces, with the inclusion of complex photoionisation processes, while simultaneously being user-friendly yet comparably fast to previously published codes.

We further demonstrate that FEniCS is a highly suitable alternative, by showing excellent agreement between our simulated results, results from published custom codes, and results from popular commercial applications.

Drift-Diffusion Model for
Streamer Discharges

Drift-Diffusion Equations

$$\frac{\partial n_i}{\partial t} - \nabla\cdot\left(\operatorname{sgn}(q_i) n_i \mu_i \nabla\varphi\right) - \nabla\cdot\left(D_i \nabla n_i\right) = S_i$$

Poisson Equation

$$-\nabla\cdot\left(\varepsilon\nabla\varphi\right) = \sigma_s + \sum_i q_i n_i$$

Helmholtz Approximation for Photoionisation

$$\nabla^2 S_{ph,j} - \left(p_{O_2}\lambda_j\right)^2 S_{ph,j} = -\left(A_j p_{O_2}^2 \frac{p_q}{p+p_q}\xi\frac{\nu_u}{\nu_i}\right) S_{im}$$

$$S_{ph} = \sum_j S_{ph,j}$$

Jump Condition at Dielectric Boundaries

$$\left(\vec{\mathbf{D}}_2 - \vec{\mathbf{D}}_1\right)\cdot\hat{\mathbf{n}} = \sigma_s$$

Source Terms

$$S_e = (\alpha - \eta)\,\mu_e|\vec{\mathbf{E}}|n_e + S_{ph}$$

$$S_+ = \alpha\mu_e|\vec{\mathbf{E}}|n_e + S_{ph}$$

$$S_- = \eta\mu_e|\vec{\mathbf{E}}|n_e$$

Implemented on the Open-Source
FEniCS Platform

FEniCS PROJECT [1] + python™ †

The FEniCS project [1] is a collection of open-source (LGPLv3) software tools for the purposes of solving differential equations using the finite-element (FE) method. The framework features a high degree of automation, allowing FE problems to be defined in a way which closely matches that of the mathematical formulation, and in minimal lines of code. Software components are natively built to support distributed memory parallelism through the message-passing interface (MPI). As such, the framework has been shown to be excellent for the rapid prototyping of FE problems, and can be scaled up to high performance computing (HPC) platforms with minimal changes to desktop-written code.

FEniCS can be interfaced through the popular programming language Python, known to be highly readable, easy to learn and require far less computational expertise to use. FEniCS backend components, however, are based in C++, thereby sacrificing little in terms of speed and computational efficiency. Dedicated automation tools enable all associated FE aspects to be easily changed, such as the spatial discretisation order, the dimensions of the problem, time integration scheme, linear algebra backend, and many more.

FEniCS has been demonstrated for streamer simulation and dielectric barrier discharges in the past [2]. However, the present work features several important improvements, which are highlighted in bold in the list of capabilities below:

- ✓ Support for unstructured triangular and tetrahedral meshes. (For 2D and 3D problems).
- ✓ Distributed memory MPI parallelisation.
- ✓ Support for solid dielectric surfaces.
- ✓ **A custom built, highly flexible adaptive mesh refinement (AMR) routine written in Python (see Fig. 1), which is user configurable.**
- ✓ **Dynamic timestepping using an implicit Euler scheme.**
- ✓ **Support for photoionisation using the Zheleznyak model [3] and the Helmholtz approximation.**

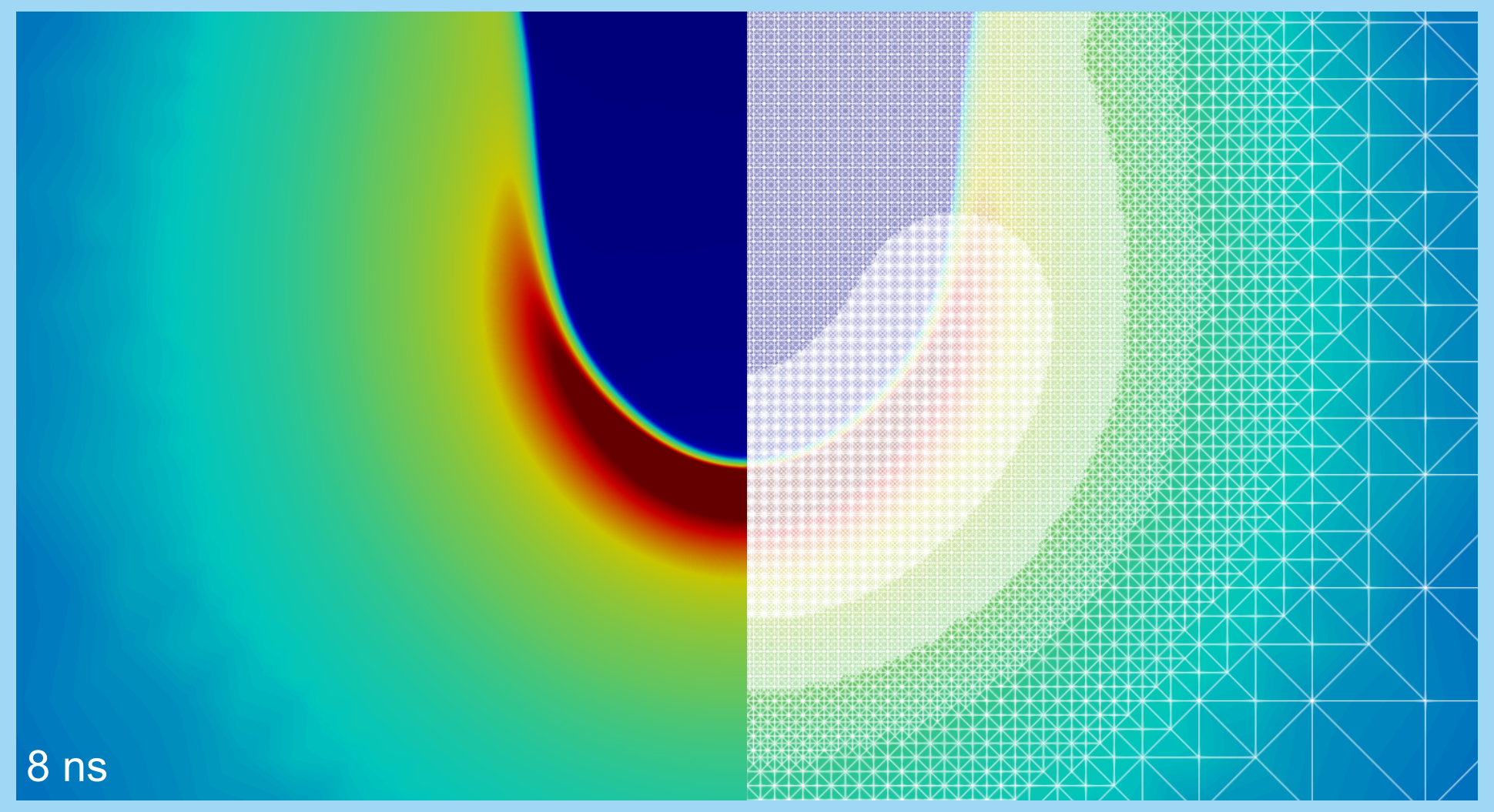

Fig. 1. Image of adaptive mesh overlaid on top of streamer head (of Fig. 2.)

Code Verification
Comparison Studies

Simulation results presented here were generated by a desktop computer with a 16-core AMD Ryzen 9 5950x at 3.4 Ghz and 64 GB memory. Linear Lagrange elements were used, with implicit time integration performed via a backward Euler scheme. The GMRES+AMG method is used in the PETSc backend.

▶ Axisymmetric Positive Streamer in Air from [4] ◀

Following a 2018 comparison study by Bagheri et al [4], we successfully simulate single positive streamers in air with photoionisation, in an axisymmetric domain (Using Bourdon's [5] parameters for photoionisation). Fig. 2 shows the time evolution of the electric field and electron density over 15 ns, while Fig. 3 compares the streamer length and maximum electric field to the prediction of other groups. Excellent agreement is attained between the present work and both custom and commercial codes. Our results are very similar to that of the DE group, who used the commercial platform COMSOL Multiphysics. Differences in settings between the groups are shown in Table 1.

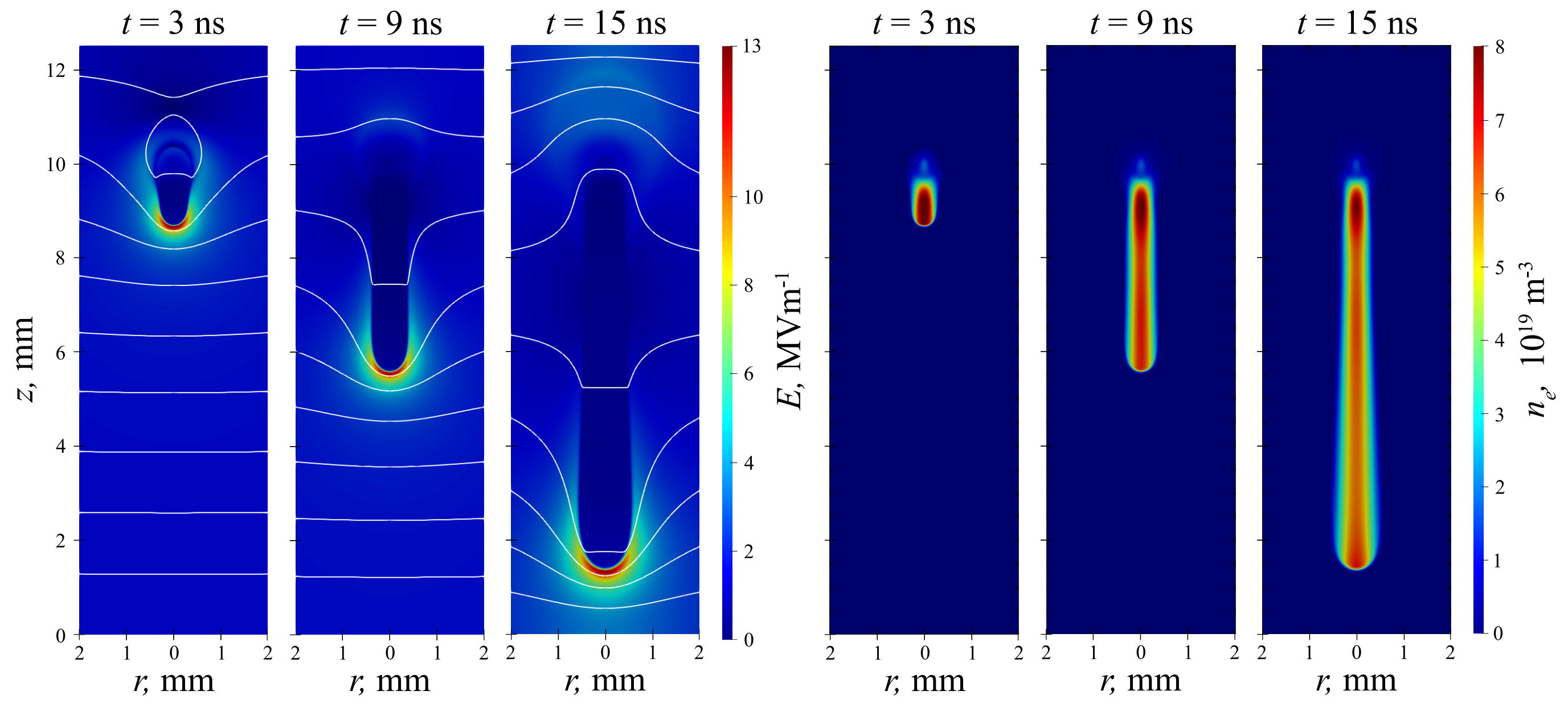

Fig. 2. Time evolution of the axisymmetric positive streamer in air using the benchmark parameters in [4] for Case (3). Panels 1-3 show the electric field, panels 4-6 show the corresponding electron density. Equipotential lines are spaced by 2 kV.

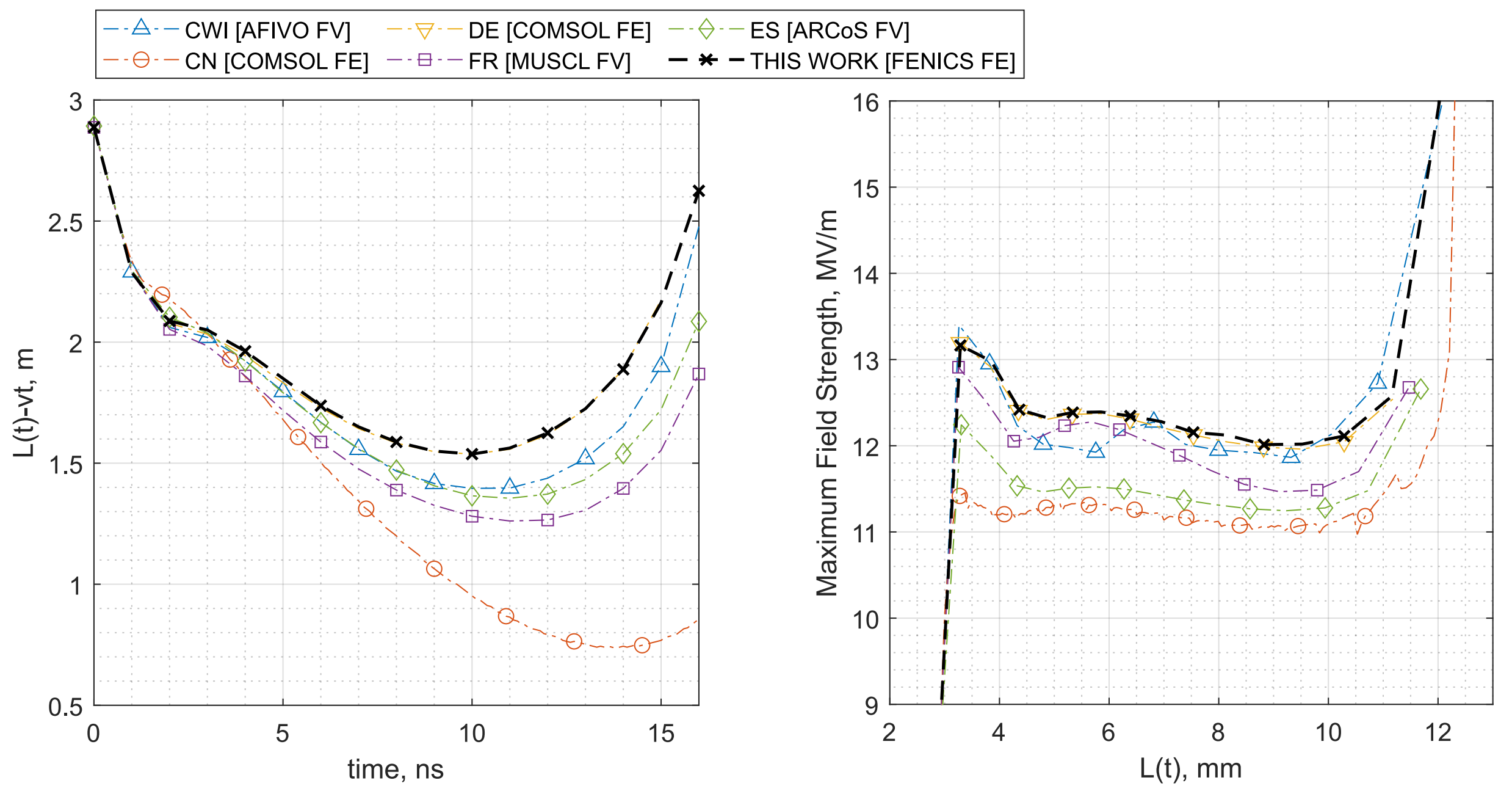

Fig. 3. (*left*) Streamer length over time (*right*) Maximum field strength of streamer against streamer length, compared with other groups who participated in the comparison study [4].

Table 1: Comparison of the present work to other groups who took part in the study [4]

	CWI	ES	FR	CN	DE	THIS WORK
Finite volume/element	FV	FV	FV	FE*	FE*	FE
Unstructured grid				✓	✓	✓
Spatial discr. order	2	2	2	1	1	1
Time discr. order	2	2	2	1 to 2	1 to 5	1
Mesh refinement	✓	✓	✓	✓	✓	✓
Adaptive refinement	✓	✓		✓		✓
Parallel	✓		✓	✓	✓	✓
Tabulated Coefficients	✓	✓				
Case 3 Simulation Settings						
Min grid size	3.0 µm	3.9 µm	6.0 µm	2.0 µm	4.1 µm	2.2 µm
Max grid size				8.0 µm	5.0 µm	
Max N_{cells}	7.6×10^4	2.0×10^6	2.9×10^5	7.2×10^5	7.6×10^5	7.8×10^5
Time step	dyn.	1.0 ps	dyn.	dyn.	dyn.	dyn.
CPU cores	4	1	1	4	6	16
Run time	<5 min	72 h	34 min	26 h	42 h	6 h

*These groups used COMSOL Multiphysics

▶ 2D Positive Surface Streamer from [6] ◀

In addition, we were able to demonstrate the inclusion of solid dielectric surfaces into the simulation domain following [6]. The results once again exhibit excellent qualitative and quantitative agreement with the original study. Photoemission and secondary emission from the surface has currently not been implemented, but are being considered.

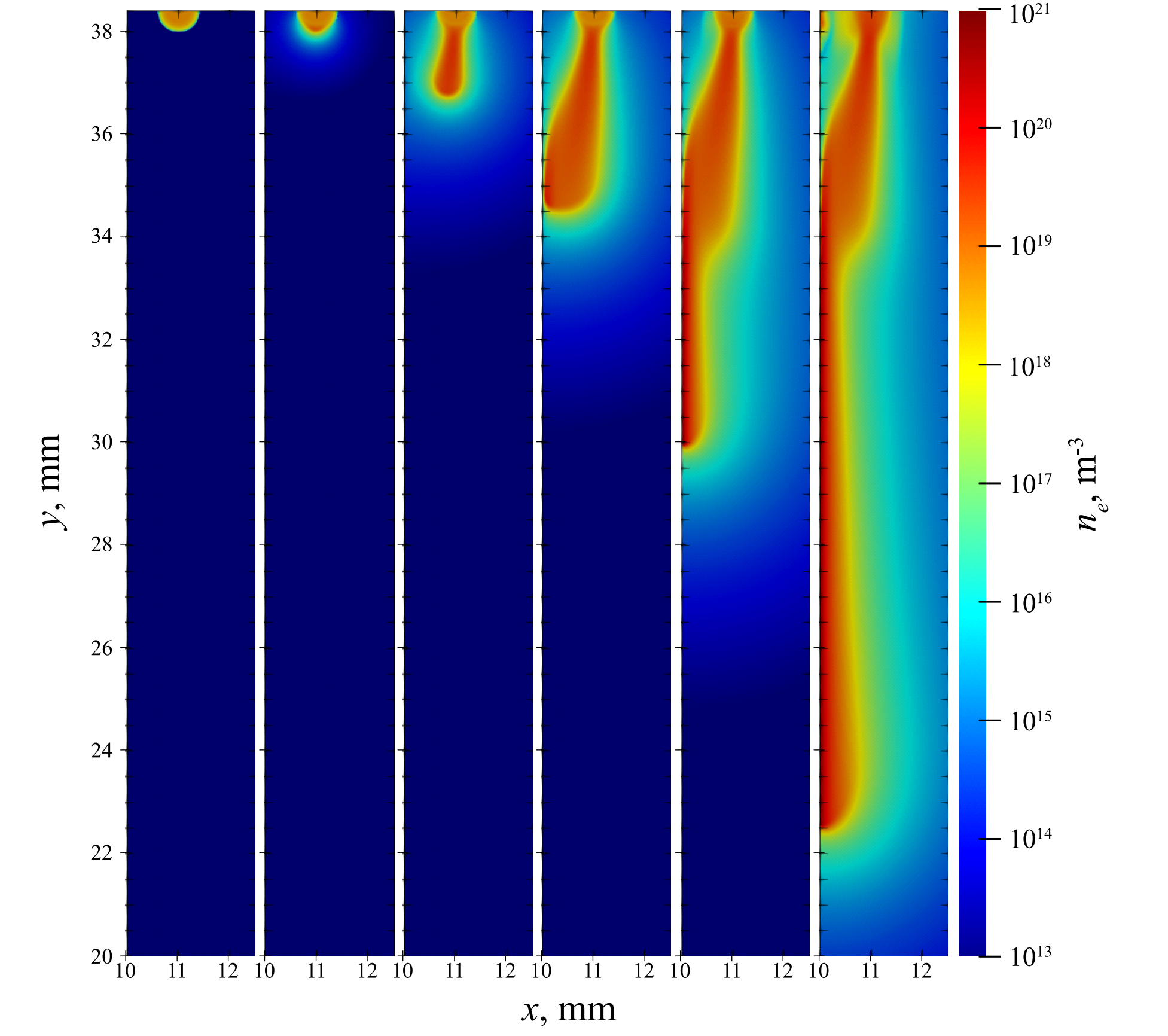

Fig. 4. Initiation, attachment and propagation of a positive streamer down a dielectric surface of relative permittivity 2, using the configuration in [6]. The surface is located on the left side of each panel (at 10 mm) in a 40x40 mm domain. Applied voltage is 100 kV. Colour plot shows the electron density, in excellent agreement with the original study.

Please attend/view Oral Session **"Modelling of Positive Streamers Using the Open-Source PDE Platform FEniCS: Initiation and Development Along a Dielectric Surface"** for further details regarding positive streamers and gas-solid interfaces

Conclusions

- In this work, we successfully demonstrate that FEniCS is a suitable platform for performing challenging streamer discharge simulations in pure gas and gas-solid topologies.
- The Python interface and modularity of FEniCS renders it highly user-friendly and excellent for rapid prototyping of problems, and scaling to final runs.
- The code was benchmarked against several other published codes and commercial applications, showing excellent agreement between results and comparable runtimes.
- The code is currently being used to study surface streamers under a variety of conditions, including different gases, pressures, and voltage signals.

References

[1] M. Alnaes *et al*, "The FeniCS Project Version 1.5," *Archive of Numerical Software 3*, 2015.

[2] A. Jovanovic, D. Loffhagen and M. Becker, "Modeling of electrical discharges using FEniCS," *72nd Annual Gaseous Electronics Conference*, November 2019, TX, USA

[3] M. Zheleznyak, A. Mnatsakanyan and S. Sizykh, "Photo-ionisation of nitrogen and oxygen mixtures by radiation from a gas-discharge," *High. Temp.*, vol. 20, pp. 357-362, 1982.

[4] B. Bagheri *et al*, "Comparison of six simulation codes for positive streamers in air," *Plasma Sources Sci. Technol.*, vol. 27, 2018.

[5] A Bourdon *et al, "*Efficieny models for photoionisation produced by non-thermal gas discharges in air based on radiative transfer and the helmholtz equations," *Plasma Sources Sci. Technol., vol. 16, pp. 656-678, 2007.*

[6] X. Li, A. Sun, G. Zhang and J. Teunissen "A computational study of positive streamers interacting with dielectrics," *Plasma Sources Sci. Technol.*, vol. 29, 2020.