IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received 28 February 2023, accepted 12 March 2023, date of publication 15 March 2023, date of current version 21 March 2023.

Digital Object Identifier 10.1109/ACCESS.2023.3257724

METHODS

The Design of a Python Library for the Automatic Definition and Simulation of Transient Ionization Fronts

TIMOTHY WONG, (Graduate Student Member, IEEE), IGOR TIMOSHKIN, (Senior Member, IEEE), SCOTT MACGREGOR, (Senior Member, IEEE), MARK WILSON, (Member, IEEE), AND MARTIN GIVEN, (Senior Member, IEEE)

Department of Electronic and Electrical Engineering, High Voltage Technologies Research Group, University of Strathclyde, G1 1XW Glasgow, U.K.

Corresponding author: Timothy Wong (timothy.wong@strath.ac.uk)

The work of Timothy Wong was supported in part by the Engineering and Physical Science Research Council (EPSRC) under Grant EP/T517938/1.

ABSTRACT In recent years, the interest in nonthermal plasma dynamics has grown significantly, within both industry and research. This has been driven by the development of several novel cold plasma technologies across a wide range of different fields, for example, for plasma medicine, chemical processing, pollution control, and surface treatment. The optimization of these technologies relies heavily upon the understanding of gas discharge plasmas: their generation, electrical characteristics, and interaction with their surroundings. Moreover, the manifestation of nonthermal plasmas in the form of streamers is of high relevance and critical importance to high voltage insulation technology, and has further significance to geophysical research concerning atmospheric discharges. The present work describes the development of the StrAFE (**Str**eamers on **A**daptive **F**inite **E**lements) package, a dedicated Python library built atop the popular open-source FEniCS finite element software, designed with the objective to simplify and to automate the solution of ionization front models. The library features support for mesh adaptivity, distributed memory parallelism, and an intuitive programming interface, while providing an exceptionally high level of user configurability. This article presents the software implementation, describes its features, and presents several code verification studies performed within simple and complex domains. It is concluded that the numerical results gained from this open-source framework are comparable to other well-established software in terms of accuracy. Therefore, it further demonstrates the great potential for open-source software to make significant contributions to technologies involving nonthermal plasmas, ionization fronts, and gas discharges.

INDEX TERMS Nonthermal plasma, gas discharge plasma, finite element analysis, open-source software, streamer discharge, high voltage phenomena, transient ionization fronts.

NOMENCLATURE

A. FINITE ELEMENT METHOD

u	Arbitrary unknown function.
u_h	Approximation of the arbitrary function u.
U_i	Coefficient to finite-element basis functions.
ϕ_i	Finite-element basis function
L	Arbitrary differential operator.
f	Arbitrary known coefficient.
v	Finite-element test functions.
Ω	Finite-element domain.
$\mathbf{A}$	Finite-element system matrix.
$\mathbf{U}$	Vector of unknown coefficients.
$\mathbf{b}$	Vector of known coefficients.

B. IONIZATION FRONT MODEL

N	Set of chemical species in a model.
n_i	Concentration of species i.
t	Time.

The associate editor coordinating the review of this manuscript and approving it for publication was Agustin Leobardo Herrera-May.

$\vec{\Gamma}_i$	Total flux of species i.
S_i	Total sources and sinks of species i.
q_i	Electric charge of species i.
μ_i	Charge mobility of species i.
D_i	Diffusion coefficient of species i.
μ_ε	Electron energy mobility.
D_ε	Electron energy diffusion coefficient.
ε	Dielectric permittivity.
φ	Scalar electric potential.
E	Electric field magnitude.
ρ	Net charge density.
$\bar{\alpha}$	Effective ionization coefficient.
α	Ionization coefficient.
η	Attachment coefficient.
h_j	Sign for loss or gain source for j-th reaction.
k_j	Reaction rate for the j-th reaction.
R	Set of chemical reactions in a model.
S_{ph}	Photoelectron source term.
S_{ion}	Total ionizing reaction source term.
p	Gas pressure.
p_q	Collisional quenching pressure.
p_{O_2}	Partial pressure of oxygen.
ν_u	Impact ionization frequency for level u.
ν_i	Ionization frequency.
ξ	Photoionization efficiency.
$A_j,\ \lambda_j$	Helmholtz fitting parameters
n_ε	Electron energy density.
$\bar{e}$	Elementary charge.
γ	Secondary emission (SE) coefficient.
$\hat{n}$	Unit normal.
k_b	Boltzmann constant.
$\bar{\varepsilon}_\gamma$	Mean energy of SE electrons.
$\nu_{th,i}$	Thermal velocity of species i.
r_i	Reflection coefficient of species i.
m_i	Mass of species i.
T_i	Temperature of species i.
σ_s	Surface charge density.

C. WEAK FORMULATION

$\mathbf{u}$	Vector of unknown functions.
$\mathbf{v}$	Vector of test functions.
u_j	j-th component of $\mathbf{u}$.
v_j	j-th component of $\mathbf{v}$.
$d\Omega$	External boundary of Ω.
ω_j	j-th boundary segment of $d\Omega$.

D. COMPUTATIONAL ASPECTS

Δt	Time integration step size.
$\mathcal{M}_b$	The base computational mesh.
$\mathcal{M}_0$	The initial computational mesh.
$\mathcal{M}_t$	A temporary computational mesh.
$\mathcal{F}$	Set of refinement functions.
$\mathcal{R}$	Set of refinement growth radii.
$\mathcal{K}$	Set of refinement tolerances.
θ	Theta time-integration scheme parameter.

E. OTHER

x, y	Cartesian geometric dimensions.
r, z	Cylindrical geometric dimensions.
U_0	Applied voltage.
N_0	Peak density of initial charge distribution.
E_{max}	Maximum electric field strength in the domain.
ε_r	Relative dielectric constant.
s	Deviation parameter for gaussian seed.

I. INTRODUCTION

Nonthermal plasmas belong to a class of non-equilibrium (non-Maxwellian) gas discharge plasma, where the electron temperature is far higher than that of the ion temperature. Such plasmas may be produced under the application of sufficiently intense electric fields to gaseous media. Also known as cold- or low-temperature plasmas, their properties have led to their application to various novel technologies, found across many different areas of engineering and science. For example, for surface processing [1], [2], air cleaning [3], [4], [5], plasma medicine [6], or chemical processing [7]. However, many aspects of nonthermal plasmas remains to be fully understood, and a deeper understanding of the related processes is crucial for the successful advancement and optimization of these developing technologies. Equally, the study of fast ionization fronts is also critical to the prevention of electrical failure in high-voltage equipment, as the initial development of streamers in high-field regions may result in partial (or complete) discharges across insulation, compromising system integrity. Streamer discharges are similarly important for the understanding of geophysical phenomena occurring in the upper atmosphere [8], and for certain plasma flow and propulsion applications [9]. These transient, filamentary-type discharges are multiscale in nature [10], rendering them highly difficult to characterize solely through experimental means.

In recent times, advances in computational power, and the growing accessibility to high performance computing (HPC), has facilitated complex multiphysics modelling, allowing it to become more widely available. For example, time-dependent simulations of streamer discharges have been successfully demonstrated on standard consumer grade hardware, using commercially available software [11], [12]. Specialized schemes using custom codes, employing techniques which are particularly effective for multiscale modelling have also been developed, and demonstrate major advances in computational efficiency and speed [13], [14]. The popularity of these methods has grown rapidly, with models that incorporate aspects such as complex plasma chemistry [15], nonuniform electric fields [16], and solid dielectric boundaries [17], [18], becoming increasingly prevalent. Currently, the software developed must typically strike a balance between issues such as computational speed, resource usage, accessibility, adaptability, accuracy, and model fidelity. Further difficulty comes with the sheer number of necessary parameters involved within plasma simulations, much of which must be

manually entered into simulation software, a process which is highly susceptible to human error [19]. Generally, custom modifications to complex, often low-level, code are required when setting up new problems, which can be time consuming, and, in many cases, require some degree of computational expertise.

In this work, the development and design of StrAFE (**Str**eamers on **A**daptive **F**inite **E**lements), a convenient and user-friendly Python library, for the automation and solution of transient ionization front problems, is described. The StrAFE system was built atop the open-source FEniCS finite element (FE) framework [20], [21], [22], which provides an already convenient yet highly flexible set of software tools for general-purpose finite element modelling. The design philosophy behind StrAFE follows directly in the footsteps of FEniCS: the rapid translation of physical models to efficient FE code; reduction in the setup time necessary to generate new models, by reducing the need for computational expertise; and to allow maximum user control and configurability.

StrAFE was developed using the now legacy (2019.1.0) version of FEniCS. Development is now focused on a next-generation version known as FEniCSx, and documentation for both versions of this platform may be found at docs.fenicsproject.org. In this article, the technical implementation of StrAFE using FEniCS is described, while also discussing the many benefits which can be gained by employing open-source code. Results from several code verification and comparison studies are then presented. It should be noted that the StrAFE library itself has not been made openly available. Rather, the focus of this article is to demonstrate how software like StrAFE can be achieved using FEniCS, which itself is open-source and fully available. Consequently, this article acts as a guide for interested parties to develop similar software, and encourages the use of open-source frameworks such as FEniCS for simulating transient ionization fronts developed in gases.

II. PLATFORM AND METHOD

A. THE FINITE ELEMENT METHOD

When using the hydrodynamic description of plasmas, solving the set of hyperbolic conservation laws which arise (see section III-A) generally benefits from numerical methods that guarantee exact local conservation, e.g., finite volume methods (FVM) as used in [13], [14], and [23]. However, this work is instead based on the continuous Galerkin finite element method (CG-FEM), to take advantage of some FEM-specific features. Despite only maintaining global conservation, FEM offers greater flexibility in terms of the order of spatial discretization, the type of finite element, and can handle complex boundaries with relative ease, without the need for any specialized numerical schemes. CG-FEM is mature, versatile, and has been frequently applied in many areas of science and engineering. Thus, a detailed review of FEM will not be conducted here, for which the reader is referred to [21], [24], and [25], and references therein. Instead, only a brief outline of necessary aspects is provided, to act as support for later discussion within this article.

To begin, let u denote the exact solution of an unknown function, and u_h be its approximation. FEM is defined when u_h is sought as a linear combination of basis functions, ϕ_i, such that:

$$u \approx u_h = \sum_i U_i \phi_i, \tag{1}$$

where U_i are coefficients to ϕ_i which approximate u. Suppose that u satisfies the differential equation:

$$L(u) = f, \tag{2}$$

in a bounded domain Ω, where L is a differential operator; then the weighted residual reads:

$$r = \int_\Omega vL(u) - \int_\Omega fv, \tag{3}$$

for weight function (or test function) v. In the standard Galerkin method, weight functions v are chosen to belong to the same function space as the approximating basis functions ϕ_i, then the Galerkin FEM problem is solved by requiring that the weighted residual be driven to zero, i.e.,

$$r = \int_\Omega vL\left(\sum_i U_i \phi_i\right) - \int_\Omega fv = 0, \tag{4}$$

where the integration over Ω is over each finite element which spatially discretizes the domain. Application of (4) over each element results in the linear system:

$$\mathbf{AU} = \mathbf{b} \tag{5}$$

where $\mathbf{A}$ is a matrix, $\mathbf{b}$ is a vector, and $\mathbf{U}$ is the vector of unknown coefficients to be obtained. The system (5) can then be solved using any preferred linear algebra solution method. In physical problems, the differential operator L often has order two, which due to (4) imposes the condition that ϕ_i must be at least twice-differentiable (i.e., elements with a linear basis cannot be used, which are preferred for their lower computational cost). To lift this constraint, Green's identity can be applied to the integral (4), which reduces second-order derivative terms to two derivatives of order one, yielding the *weak* or *variational formulation* of (4). This is illustrated further in section III-G for the specific case of coupled drift-diffusion-reaction-Poisson equations.

B. THE FEniCS PROJECT

The FEniCS Project [20], [21], [22] is an open-source collection of software components, designed with the aim to automate and simplify general FE problem generation and solution. Licensed under LPGL-v3, the original platform was developed using C++, and features several core components under the main user interface named DOLFIN. Fig. 1 outlines the corresponding system architecture; using corresponding labels:

(i) *FEniCS Form Compiler (FFC)* [21], [26], [27] – Handles just-in-time (JIT) compilation and generation of

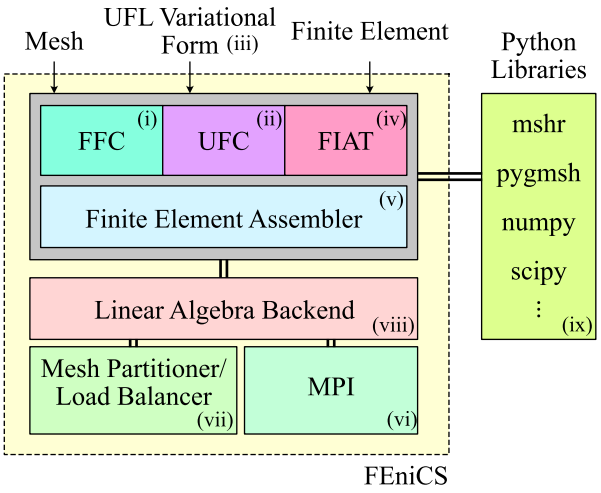

FIGURE 1. Block diagram depicting the architecture of FEniCS, adapted from [48].

finite element variational forms in high-speed C++ code.

(ii) *Unified Form Assembly Code (UFC)* [28] – An interface for finite element assembly from provided variational forms.

(iii) *Unified Form Language (UFL)* [20], [28] – A language developed for the discretization of partial-differential equations (PDE), in a form that closely resembles the original mathematical expressions.

(iv) *Finite Element Automatic Tabulator (FIAT)* [29] – Handles the generation of finite element basis functions and elements of arbitrary order.

The components listed above work alongside the system assembler (v) to form the system matrix, to then be passed to the solver. FEniCS can additionally be programmed using Python, which significantly lowers the entry barrier to performing efficient FEM simulations. Furthermore, there is native support for distributed memory parallelization through MPI (vi), with built-in mesh partitioning and load balancing, performed using the PT-SCOTCH [30] or ParMETIS [31] libraries (vii). FEniCS can also be compiled with a selection of popular open-source linear algebra backends (viii), which provides efficient computation, e.g., PETSc [32], [33], uBLAS [34], or Epetra [35]. Direct programming through Python also facilitates the automated processing of data outputs, since scripts for data analysis can be constructed using any available third-party Python library (ix), and can directly communicate with FEniCS. The platform has proved highly capable in numerous studies across a range of disciplines [36], [37], [38], and has shown to adapt well to HPC settings with over 24,000 processes [22], [39]. StrAFE is primarily an automation tool built on top of FEniCS through its Python interface, but additionally implements crucial features such as adaptive mesh refinement for efficient multiscale modelling.

III. MATHEMATICAL MODELLING

A. HYDRODYNAMIC DESCRIPTION OF NONTHERMAL PLASMAS

The hydrodynamic (or fluid) approximation of plasma arises from the zero-order moment of the Boltzmann equation [40], and becomes valid when charge concentrations are sufficiently high, or if the Knudsen number is within the bounds of validity [40]. Consider a plasma comprised of a set, N, of different charged species, interacting under the presence of an electric field. According to the hydrodynamic approximation, the spatial and temporal evolution of their concentrations can be modelled following a set of advection-diffusion-reaction equations, given by:

$$\frac{\partial n_i}{\partial t} + \nabla \cdot \vec{\Gamma}_i = S_i, \tag{6}$$

for $i \in N$, where the term S_i describes the sources and sinks of species i, and the flux $\vec{\Gamma}_i$ is given by the sum of advective and diffusive components,

$$\vec{\Gamma}_i = -\text{sgn}\,(q_i)\, n_i \mu_i \nabla\varphi - D_i \nabla n_i, \tag{7}$$

where q_i, n_i, μ_i, and D_i are the electric charge, concentration, (positive) mobility, and diffusion coefficient of species i, respectively. φ is the potential, coupled to the charge concentrations through the Poisson equation,

$$-\nabla \cdot (\varepsilon \nabla\varphi) = \rho = \sum_{j \in N} q_j n_j, \tag{8}$$

where ε is the permittivity. The self-consistent solution of (6) and (8) with appropriate boundary conditions, initial conditions, and charge sources, forms the basic hydrodynamic description of plasmas. The types of initial and boundary conditions which are supported in StrAFE are described in the following sections, as are several additional functions, which have been developed to improve modelling fidelity. Sections III-A to III-G firstly focuses on the mathematical formulation, while details regarding the usage and syntactical implementation of the software can be found in section IV.

B. USING TOWNSEND COEFFICIENTS

A basic description of plasma can be gained using a minimum of two species – electrons, and generic positive ions. Using a minimal model, the Townsend ionisation coefficient can be used to define an ionisation source term of the form:

$$S_i = \bar{\alpha} n_e \mu_e |\nabla\varphi|, \tag{9}$$

where $\bar{\alpha}$ is the effective ionisation coefficient, and the subscript e refers to electrons. Alternatively, $\bar{\alpha}$ can be split into $\bar{\alpha} = \alpha - \eta$, where α, η are the ionization and attachment coefficients, respectively, and the set N includes negative ions, with the source term:

$$S_- = \eta n_e \mu_e |\nabla\varphi|. \tag{10}$$

For advanced models using individual reaction rate coefficients, and considers specific plasma chemistry, section III-C applies.

C. PLASMA CHEMISTRY

For detailed studies which concern plasma composition, and which involve numerous active ionic species, source terms are computed following:

$$S_i = \sum_{j=1}^{n} \left(h_j k_j \prod_{m \in R} n_m \right), \tag{11}$$

where k_j is the reaction rate for reaction j, n is the total number of reactions which involve species i, and R is the set of reactants involved in reaction j. Symbol h_j is either $+1$ or -1 for sources and sinks, respectively. Examples which employ the plasma chemistry module of StrAFE have been demonstrated in section V.

D. PHOTOIONIZATION

In some gases (e.g., atmospheric air), photoionization induced by emitted photons from radiative de-excitation processes, are thought to be a major contributor of free electrons to sustained discharges such as positive streamers [41], [42], [43]. To account for this, Zheleznyak's model [41] has been implemented with a photoelectron source term given by:

$$S_{ph} = \frac{p_q}{p + p_q} \xi \frac{\nu_u}{\nu_i} \sum_{j=1} p_{O_2}^2 A_j \iiint_{V'} \frac{S_{ion} e^{-\lambda p_{O_2} |r - r'|}}{4\pi |r - r'|} \, dV'. \tag{12}$$

The photoelectron source can be efficiently obtained by using the Helmholtz approximation as described by Bourdon et al. [44]:

$$\nabla^2 S_{ph,j} - \left(p_{O_2} \lambda_j\right)^2 S_{ph,j} = -\left(A_j p_{O_2}^2 \frac{p_q}{p + p_q} \xi \frac{\nu_u}{\nu_i} \right) S_{ion},$$
$$S_{ph} = \sum_j S_{ph,j}, \tag{13}$$

for $j = 1, 2, 3$. Symbols p, p_q, and p_{O_2} are the total gas pressure, collisional quenching pressure of nitrogen (accounting for non-radiative de-excitation), and the partial pressure of oxygen, respectively. ν_u is the impact ionisation frequency for level u, ν_i is the ionisation frequency, ξ is the photoionization efficiency [44], and S_{ion} is the summed source term over all contributing ionizing reactions. Lastly, A_j and λ_j are fitting parameters for the j-th equation, and the values which are used throughout the examples within section V follow those computed in [44].

E. LOCAL FIELD AND LOCAL MEAN ENERGY APPROXIMATIONS

In many previous studies, the local field approximation (LFA) has been used for simplicity, which assumes that transport and rate coefficients are dependent only on the local magnitude of the electric field, which becomes immediately available from the solution of (8), and then applying $\vec{E} = -\nabla\varphi$. The range of validity of the LFA has been shown to be limited, and may become inaccurate for high- or spatially-inhomogeneous field conditions [45], [46]. In StrAFE, the LFA is the default option, however, there also exist the option to use the local mean energy approximation (LMEA), which attempts to include (to some extent) non-local electron transport processes, thereby extending the range of validity to beyond that of the LFA. Under the LMEA, transport and rate coefficients are dependent on the local value of the electron energy, which is dynamically computed by appending an additional balance equation to the model, given by:

$$\frac{\partial n_\varepsilon}{\partial t} + \nabla \cdot \vec{\Gamma}_\varepsilon = \bar{e} \vec{\Gamma}_e \cdot \nabla\varphi - \sum_{j=1}^{n} \left(\Delta E_j k_j \prod_{m \in R} n_m \right), \tag{14}$$

where the electron density flux $\vec{\Gamma}_\varepsilon$ is

$$\vec{\Gamma}_\varepsilon = n_\varepsilon \mu_\varepsilon \nabla\varphi - D_\varepsilon \nabla n_\varepsilon. \tag{15}$$

Here, n_ε is the electron energy density, $\bar{e}$ is the elementary charge, $\vec{\Gamma}_e$ is the electron flux, and ΔE_j is the energy loss or gain during reaction j. μ_ε and D_ε are the energy mobility and energy diffusion coefficient, respectively; these can either be user defined, or optionally, be approximated from the electron transport parameters [40] as:

$$\mu_\varepsilon = \frac{5}{3}\mu_e, \; D_\varepsilon = \frac{5}{3} D_e. \tag{16}$$

F. BOUNDARY CONDITIONS

For electrode boundaries, Dirichlet boundary conditions are prescribed for the electrostatic potential, while Neumann-zero conditions are applied at axes of symmetry. For studies involving subdomains (e.g., solid barriers, electrodes, walls), the accurate reflection of physical behavior requires appropriate boundary conditions for the normal charge fluxes to be prescribed at interfaces. In StrAFE, the option exists to have either a Neumann-zero condition, or wall conditions following Hagelaar et al. [47]. Using similar notation to Jovanovic et al. [19], these are given by:

$$\vec{\Gamma}_e \cdot \hat{n} = \frac{1 - r_e}{1 + r_e} \left(|n_e \mu_e \nabla\varphi \cdot \hat{n}| + \frac{1}{2} n_e v_{th,e} \right)$$
$$- \frac{2}{1 + r_e} \gamma \sum_{\substack{j \in N \\ j \neq e}} \max\left(\vec{\Gamma}_j \cdot \hat{n}, 0 \right) \tag{17}$$

for electrons,

$$\vec{\Gamma}_i \cdot \hat{n} = \frac{1 - r_i}{1 + r_i} \left(|n_i \mu_i \nabla\varphi \cdot \hat{n}| + \frac{1}{2} n_i v_{th,i} \right) \tag{18}$$

for heavy species, and

$$\vec{\Gamma}_\varepsilon \cdot \hat{n} = \frac{1 - r_e}{1 + r_e} \left(|n_\varepsilon \mu_\varepsilon \nabla\varphi \cdot \hat{n}| + \frac{1}{2} n_\varepsilon v_{th,\varepsilon} \right)$$
$$- \frac{2}{1 + r_e} \bar{\varepsilon}_\gamma \gamma \sum_{\substack{j \in N \\ j \neq e}} \max\left(\vec{\Gamma}_j \cdot \hat{n}, 0 \right) \tag{19}$$

for the electron energy density. The parameter r is the reflection coefficient of the species at the boundary, γ is the secondary emission coefficient, and $\bar{\varepsilon}_\gamma$ is the mean energy of

secondary electrons. v_{th} is the thermal velocity of the species identified by the subscript, given by:

$$v_{th,i} = \sqrt{\frac{8k_bT_i}{\pi m_i}}, \tag{20}$$

where k_B is the Boltzmann constant, T_i is the species temperature, and m_i is the species mass. For the electron temperature, T_e is calculated from the energy following:

$$T_e = \frac{2n_\varepsilon}{3k_bn_e}, \tag{21}$$

and for the electron energy velocity, equation (22) holds:

$$v_{th,\varepsilon} = 2k_bT_ev_{th,e}. \tag{22}$$

Across interfacial boundaries with differing permittivity, a discontinuity in the electric displacement is induced by the accumulation of surface charge, σ_s, from incoming fluxes (volume conduction is currently not considered). To incorporate this, internal subdomain boundaries require that:

$$\varepsilon\nabla\varphi \cdot \hat{n} = \sigma_s, \tag{23}$$

where the surface charge is computed based on the sum of boundary-directed fluxes, following:

$$\frac{\partial\sigma_s}{\partial t} = \bar{e}\sum_{j\in N}\text{sgn}\,(q_i)\,\vec{\Gamma}_j\cdot\hat{n} + \bar{e}\gamma\sum_{\substack{j\in N\\ j\neq e}}\vec{\Gamma}_j\cdot\hat{n}. \tag{24}$$

G. WEAK FORMULATION

As described in section II-A, the application of the Galerkin method requires the reformulation of equations (6)–(24) to hold weakly with respect to an appropriate set of test functions. The weak formulation of the governing equations is fundamental to the implementation of StrAFE, which uses the UFL syntax to directly input math-like form expressions, which are passed to the system assembler. See [20] and [21] for a detailed description of UFL and of the FEniCS form compiler. In the interests of repeatability, the remainder of this section focuses on providing the mathematical definition of the variational problem. To maintain consistency with indexing in Python, $\mathbf{u}$ is hereby defined as a vector function with components $\mathbf{u} = [u_0, u_1, u_2, \ldots, u_p]$, with index starting at zero, each of which represents a scalar field in a domain Ω, and where p varies depending on the problem type (i.e., the number of species considered, whether LFA or LMEA is used, etc.). The rules for the construction of this vector are as follows:

- Index 0 is always the potential field, φ.
- Index 1 is always the electron density, n_e.
- Index 2 to N stores the densities of all other species under consideration, e.g., $(n_+, n_-, \ldots)$.
- Index $N+1$ to $N+3$ stores the three components of the photoionization source term, $S_{ph,1}$, $S_{ph,2}$, and $S_{ph,3}$, if enabled.
- Index $N+4$ stores the surface charge σ_s, defined only on element facets, and only if there are subdomains.
- The last index of the vector (in Python, index -1) stores the electron energy density, n_ε, if LMEA is enabled.

In summary, the mapping of vector $\mathbf{u}$ can be expressed as:

$$\mathbf{u} = \begin{bmatrix} u_0\\ u_1\\ u_2\\ u_3\\ \vdots\\ u_N\\ u_{N+1}\\ u_{N+2}\\ u_{N+3}\\ u_{N+4}\\ u_{-1}\end{bmatrix} \equiv \begin{bmatrix} \varphi\\ n_e\\ n_2\\ n_3\\ \vdots\\ n_N\\ S_{ph,1}\\ S_{ph,2}\\ S_{ph,3}\\ \sigma_s\\ n_\varepsilon\end{bmatrix} \tag{25}$$

Correspondingly, a vector of test functions, $\mathbf{v} = [v_0, v_1, v_2, \ldots, v_p]$, is stored for the construction of variational forms, with each function v_j acting as the test function for the corresponding index in $\mathbf{u}$.

Consider firstly the Poisson equation (8), from which one may construct the weak form following section II-A to be:

$$\int_\Omega v_0\nabla\varepsilon\cdot\nabla u_0\,d\Omega + \int_\Omega v_0\sum_{j\in N}q_jn_j\,d\Omega - \int_\Omega\nabla\left(v_0\varepsilon\right)\cdot\nabla u_0\,d\Omega + \int_{\partial\Omega}v_0\varepsilon\left(\nabla u_0\cdot\hat{n}\right)\,dS = 0 \tag{26}$$

where $\partial\Omega$ denotes the external boundary of domain Ω, arising from Green's identity. The last integral includes the term $\varepsilon\left(\nabla u_0\cdot\hat{n}\right)$ which may be chosen to enforce Neumann boundary conditions such as (23), or, if a Neumann-zero condition is present, this integral vanishes. Different Neumann conditions can also be prescribed to disjoint sections of $\partial\Omega$, simply by splitting this integral into boundary segments such that:

$$\bigcup_j\omega_j = \partial\Omega, \tag{27}$$

where ω is a boundary segment, and separate integrations are performed over each segment j, respectively. For advection-diffusion-reaction equations (6), the weak form becomes:

$$\begin{aligned}\int_\Omega\frac{\partial u_i}{\partial t}v_i\,d\Omega &- \text{sgn}(q_i)\left[\int_\Omega v_i\nabla\left(u_i\mu_i\right)\cdot\nabla\varphi\,d\Omega - \int_\Omega\nabla\left(v_iu_i\mu_i\right)\cdot\nabla\varphi\,d\Omega\right]\\ &+\int_\Omega\nabla\left(v_iD_i\right)\cdot\nabla u_i\,d\Omega - \int_\Omega v_i\nabla D_i\cdot\nabla u_i\,d\Omega\\ &-\int_\Omega S_iv_i\,d\Omega + \int_{\partial\Omega}v_i\left(\vec{\Gamma}_i\cdot\hat{n}\right)\,dS = 0\end{aligned} \tag{28}$$

for $i \in N$, and similarly for the energy balance equation (14), but with the right-hand-side replacing S_i in (28). The

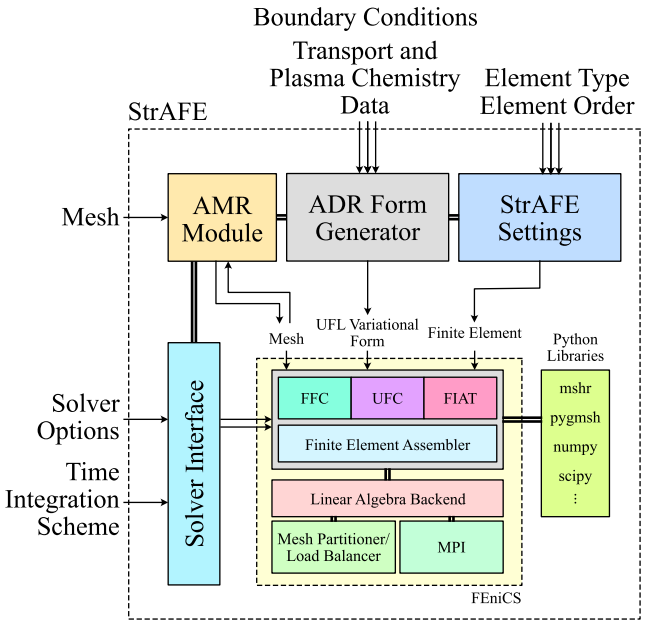

FIGURE 2. Block diagram of the architecture of the StrAFE library, and its connections to the core FEniCS library as shown in Fig. 1.

Helmholtz weak form for photoionization (13) is obtained in the same way, yielding:

$$\int_\Omega \nabla v_i \cdot \nabla u_i \, d\Omega + \int_\Omega v_i \left(p_{O_2}\lambda_j\right)^2 u_i \, d\Omega - \int_\Omega v_i \left(A_j p_{O_2}^2 \frac{p_q}{p+p_q} \xi \frac{\nu_u}{\nu_i}\right) S_{ion} \, d\Omega - \int_{\partial\Omega} v_i \left(\nabla u_i \cdot \hat{n}\right) \, dS = 0, \tag{29}$$

for $i = N+1, N+2, N+3$.

IV. StrAFE

Fig. 2 shows the architecture of the StrAFE library, and describes its interconnections to the core FEniCS system as laid out in [48] and in Fig. 1. StrAFE was developed with a strong emphasis on usability and flexibility, in-line with the original design philosophy for FEniCS. Classes provided through StrAFE have been designed to be intuitive, and to require only basic knowledge of the Python programming language. Sections IV-A to IV-E gives a broad overview of the components of StrAFE, providing details regarding their implementation and usage.

A. BASIC USAGE AND CLASSES

In this section, classes which are central to StrAFE's functionality are described, to highlight how the combination of FEniCS and Python has been used to significantly simplify the process of configuring simulation models.

At the core of any simulation is the computational mesh. The inheritance of StrAFE from FEniCS therefore permits that any mesh format supported by FEniCS as listed in [21], is also supported by StrAFE. This includes internally generated meshes using native FEniCS functions, and external meshes from any other software. Both 2D and 3D unstructured meshes are supported, and are compatible with mesh routines as described in section IV-B. A simulation can then be configured using a number of dedicated StrAFE classes, the most important of which are described in brief below:

- `DriftDiffusionProblem()` - The main problem class, where all simulation settings can be accessed and changed. Uses intuitive `.set` syntax, such as `set_base_mesh(mesh)` to provide a meshed domain object `mesh` to the solver. All settings pertaining to the problem can be set similarly, including the choice of LFA or LMEA, the species under consideration, AMR schemes, time-stepping, etc.
- `ChargedSpecies()` - Defines a single instance of a charged species to be tracked within the simulation. This class stores the species properties, such as its name, mass, charge, and transport parameters (can be provided as Python functions or tabulated data, such as those computed using BOLSIG+ [49]). Initial conditions are also stored in this class, where several popular initial charge distributions can be chosen from, e.g., a gaussian [50], capsule [17], or uniform distribution.
- `Electrode()` - Defines nodes on the mesh which are to be marked as electrodes. Doing so allows the application of potential boundary conditions. This is also a child class of the more general `DirichletCondition()`, which can be used for the prescription of arbitrary Dirichlet conditions, and is not limited to the potential field.
- `Wall()` - Defines nodes on the mesh which are to be marked as a wall. Doing so allows the application of the wall conditions according to (17)–(19). This class stores the attributes of the wall, such as reflection and secondary emission properties. This is also a child class of the more general `NeumannCondition()`, which can be used for the prescription of arbitrary Neumann conditions, and is not limited to the charge fluxes.

A subsequent call to the `.solve()` method launches the system assembler, which automatically constructs the UFL forms of equations (26)–(29), based on the provided settings. This is subsequently passed to the solver environment, and the time-stepping loop is invoked. In this way, simulation problems can be configured using minimal lines of highly-readable code. This facilitates the rapid development and refinement of computational models, and has strong support for integration with external, Python-developed code. This is particularly useful for performing batch parametric studies, or to conduct automatic post-processing tasks on the simulation outputs using other Python-based programs.

B. ADAPTIVE MESH REFINEMENT (AMR)

A part of the motivation to develop StrAFE, was the want of a flexible and transparent framework to study fast-transient ionisation waves, such as streamer discharges, with emphasis

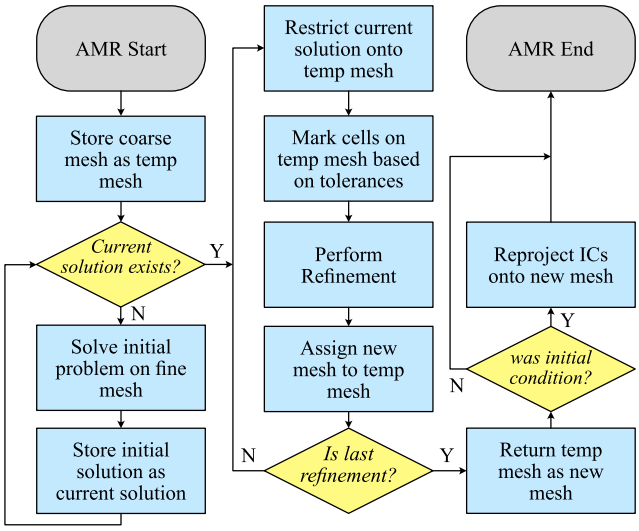

FIGURE 3. Flowchart of the adaptive mesh refinement algorithm implemented in StrAFE.

on divergent field and time-varying field conditions. As mentioned, streamer discharges are multiscale phenomena [10], and often exhibit sharp features and steep spatial gradients, particularly within their electric field and ionic density distributions. This imparts considerable computational difficulty for mesh-based methods like FEM, where the accuracy is highly dependent on mesh resolution, but where computational resources are often limited. Therefore, adaptive meshing techniques have essentially become a necessity for simulating streamers in complex domains, or for longer time periods. These methods automatically adapt the mesh elements in time and space to suit the evolving dynamics of the simulation. Most commonly, this requires refinement of mesh cells in regions of fast-moving or high-gradient behavior, and subsequent de-refinement of cells where the solution is once again slow-varying (also known as *h*-type adaptivity, as opposed to *p*-type adaptivity which locally adapts the order of the approximating basis functions). The criterion for refinement is typically based upon an error estimate, or alternatively, can simply be based upon some combination of cell or nodal values of the most recently computed solution.

At the time of development of StrAFE, FEniCS did not include native support for time-dependent adaptive meshing schemes that allowed for per-cell de-refinement, and was limited to refinement only. To include this feature, StrAFE implements several refinement routines for the purpose of achieving dynamic meshes in time and in space. This is done through periodic domain remeshing at a user-defined frequency. Fig. 3 shows a flowchart of the AMR algorithm which has been employed, and which is also described in further detail by the following algorithm:

1) At the beginning of the simulation, store a base coarse mesh $\mathcal{M}_b$, and a fine mesh $\mathcal{M}_0$ to solve for initial conditions.
2) If it is the first iteration, project initial conditions onto $\mathcal{M}_0$, solve once for u_0, and store a temporary copy $u_r \leftarrow u_0$. If not, assign the current solution $u_r \leftarrow u_k$. In either case, set a temporary mesh $\mathcal{M}_t \leftarrow \mathcal{M}_b$.
3) Construct a set of functions $\mathcal{F}$ (or combination of functions) from those stored in u_r, which are chosen by the user, to be evaluated against the refinement criteria.
4) For all functions $f \in \mathcal{F}$, mark cells of $\mathcal{M}_t$ for refinement if $f \geq \kappa_\ell$, where κ_ℓ is a tolerance defined by a set of tolerances in the simulation settings, $\kappa_\ell \in \mathcal{K}$.
5) (Optional) grow the region of refinement markers by marking all cells within a distance $\mathcal{R}$ from all marked cells in $\mathcal{M}_t$.
6) Refine $\mathcal{M}_t$ based on these markers g_ℓ times, then reassign $\mathcal{M}_t \leftarrow \mathcal{M}_t$.
7) For the number of total required refinement levels, ℓ, increment ℓ and repeat from step 4.
8) Return the fully refined mesh $\mathcal{M}_t$.
9) If it is the first iteration, re-project initial conditions onto $\mathcal{M}_t$ and re-solve before continuing.
10) Project the current solution onto $\mathcal{M}_t$ and re-assemble the variational problem.

The refinement functions $\mathcal{F}$, tolerances $\mathcal{K}$, refinement multipliers g_ℓ, refinement levels ℓ, and radial growth distance $\mathcal{R}$, can all be user-defined as Python functions and passed to the problem environment, using similar `.set` syntax as described in section IV-A. The AMR routine also supports time as an input argument, which allows the refinement criteria to be adjusted automatically over the course of the simulation. Moreover, marker functions can be provided, which allows the refinement region to be limited to a user-specified sub-domain of the mesh. Additional routines which permit the static refinement of the base mesh $\mathcal{M}_b$ around a point, a line, or a user provided region have also been developed. These are particularly useful for simulations involving static features, like dielectric boundaries or particles, which may require a denser mesh near its interfacial region, to ensure good numerical convergence. AMR would then refine on top of the already refined base mesh. Using this scheme, granular control of the mesh is provided to the user, as there exists many different methods to refine the mesh precisely where required over the course of the simulation. It is remarked that the AMR algorithm operates in parallel with MPI, and has been designed to minimize inter-process communication where possible. An example of an adaptively refined mesh using this scheme is shown in Appendix A.

C. TIME INTEGRATION

For time-stepping, StrAFE currently employs the θ-scheme, such that the time derivative of the function $\mathbf{u}$ as featured in (25)–(29) is discretized as:

$$\frac{\partial \mathbf{u}}{\partial t} \approx \frac{\mathbf{u}_{k+1} - \mathbf{u}_k}{\Delta t} = \theta f_{k+1}\left[t, \mathbf{u}_{k+1}\right] + (1-\theta) f_k\left[t, \mathbf{u}_k\right]. \tag{30}$$

Setting the parameter $\theta = 0$ or $\theta = 1$ results in the first-order explicit and implicit Euler schemes, respectively, while the default $\theta = 1/2$ recovers the Crank-Nicolson scheme, giving

second order accuracy in time. Currently, StrAFE also has some initial support for adaptive time stepping, which when enabled, returns an estimate of the local truncation error, from which the user may implement custom timestep controllers, to update Δt for the next step based on their own specified tolerances. In future, the implementation of higher order schemes would be highly beneficial, to increase solution accuracy and numerical convergence.

D. SUBDOMAIN SUPPORT

In practice, many engineering systems are composite in nature, such that interfaces between materials (e.g., between gas and solid dielectrics) may exist inside the domain of interest, and may not necessarily be positioned at the external boundaries. In such cases, equations (23), (17)–(19) must be applied for internal nodes of the computational mesh. StrAFE currently supports subdomains for modelling solid dielectric material with different relative permittivity values.

To create subdomains in StrAFE, the user must first ensure that the attached mesh has nodes that align with the internal boundaries, which can be achieved with ease using available mesh generation software, e.g., gmsh [51]. Boundary and subdomain markers should also be generated from external software, unless the boundary interface conforms to simple shapes which can be described analytically, in which case, they can be marked using native FEniCS tools [21]. A subdomain is defined by passing the cell marker function, the boundary marker function, and the value of permittivity, reflection, secondary emission, etc., to the `.set_solid_subdomain()` method. If this subdomain boundary function is used, conditions given by (17)–(19) are automatically applied, and do not need to be manually provided by setting Neumann conditions. If not, the boundary defaults to a Neumann-zero condition.

An arbitrary number of solid subdomains can be defined, which are internally stored as a list of markers. These are automatically regenerated on each new mesh if AMR is enabled. The markers instruct the assembler to split the meshed domain Ω into two sets, $\Omega = \Omega_s \cup \Omega_g$, for solid and gas, respectively, where the drift-diffusion equations (6) are solved in Ω_g only, while Poisson's equation (8) is defined over the full domain Ω. It is remarked that, although StrAFE currently does not provide a convenient interface to define additional physics within subdomains, this could theoretically be done in future, by simply appending the corresponding variational forms before passing to the solver. This may be useful for studies involving simultaneous charge transport in solid media, or subdomains which contain liquid dielectrics and fluid flow.

E. PROBLEM GENERATION AND PLASMA CHEMISTRY INPUT

In section IV-A, the `ChargedSpecies()` class was described, which is used to define and include a single charged species. However, in many nonthermal plasma applications, e.g., [7] and [15], knowledge of the plasma composition, and of the exact combination of charged species developed during a discharge, is of critical importance. Manual definition of each individual species using `ChargedSpecies()` would be tedious and error-prone. For this, StrAFE features automated generation of the species list from a plaintext file, containing a list of ionic species, neutral species, table of chemical reactions, rate coefficients, and energy loss coefficients. Using the `TabulatedSpecies()` class, this user provided file can be read, and the list of species is automatically generated and attached to the problem environment. Parameters can be provided within the text file as constants, plaintext functions, or a file path pointing toward tabulated data. The chemistry data is independent of the mesh data, therefore allowing studies in different gases to be achieved with ease, simply by switching to a different chemistry file.

F. SOLVER AND OTHER MISCELLANEOUS OPTIONS

On default settings, StrAFE solves the linear system using the solvers provided by the PETSc [33] backend through FEniCS. Newton's method is used for the outer iteration, whilst the generalized minimum residual method (GMRES), preconditioned using the Hypre [52] algebraic multigrid (AMG) method is used as the inner linear solver. However, the solver interface is fully exposed, and can be configured to meet the user's requirements. Other options include biconjugate gradient methods, various relaxation methods, and popular direct solvers such as MUMPS [53]. For a full list of supported linear algebra solvers and settings, the reader is referred to [21].

The versatility which StrAFE has inherited from FEniCS further enables other options to be readily changed, for example, the use of higher-order spatial discretization, or different element types. Moreover, it is possible to override or modify the variational form, which is assembled only when `.solve()` is called. This allows the implementation of additional physics, stabilization schemes, or other, to suit the exact application.

V. SIMULATION EXAMPLES

Since StrAFE relies upon FEniCS for system assembly and for its solver routines, it is unnecessary for this article to concern itself with the detailed benchmarking and validation of the numerical assembly and solver components. Instead, the objective is to demonstrate the accuracy and capabilities of the StrAFE interface, specifically for streamer and plasma modelling, which has been subsequently developed. At the time of writing, there exists no standardized 'benchmark' problems for nonthermal plasma simulations, though this may change as the interest in such numerical codes increase, with studies like [50]. Instead, the present section demonstrates several examples of streamer discharge simulations, in multiple configurations which have been reported in past literature, and were conducted by multiple other research groups. For each case, the simulation domain, boundary conditions, and relevant model settings have been

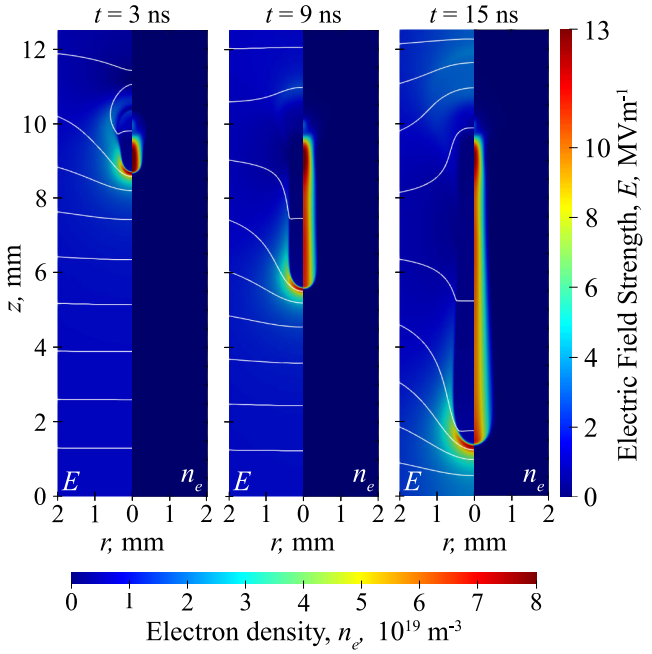

FIGURE 4. Time evolution of an axisymmetric positive streamers' electric field and electron density, at times $t = 3$, 9, and 15 ns, generated using StrAFE. Original study from [50]. Equipotential lines are spaced by 2 kV. This simulation completed in approximately 3-4 hours.

briefly provided before the results are presented. Where possible, direct comparisons to original studies have been made, though this was not always possible, as it was dependent on data availability. Unless otherwise specified, all simulations were performed using triangular linear Lagrange elements with StrAFE running inside a Docker container [54], on either 16-core (AMD Ryzen 9 5950X) or 18-core (Intel Xeon W-2295) workstation computers with 64GB memory. It is remarked that while all the necessary components for 3D simulations have been implemented, only 2D (or 2D-axisymmetric) simulations have thus far been conducted. This is primarily due to the need for code verification and comparison, for which there is a far greater number of documented 2D studies. Since full 3D simulations have only become possible recently [13], [14], it would be difficult to evaluate the accuracy of the implementation based on limited data. However, it would be of high interest to conduct 3D simulations in the future, especially to evaluate StrAFE under HPC settings.

A. AXISYMMETRIC POSITIVE STREAMER

For the hydrodynamic approximation used in StrAFE, [50] provides the most recent and comprehensive comparison of different codes used by different groups for the simulation of streamer discharges. At the time of writing, it is essentially the only study which has been conducted purely for the purposes of verification and comparison. As such, evaluating results attained using StrAFE against the multiple available datasets from [50] was prioritized.

Focus was placed on the 'case 3' simulation of [50], to allow for the implementation of the Helmholtz photoionization model to also be evaluated. The domain consisted of a 2D square box with dimensions $(r, z) = [1.25, 1.25]$ cm, and was rotationally symmetric around $r = 0$. Dirichlet conditions of $U_0 = 18.75$ kV and 0 kV were prescribed at $z = 1.25$ cm and $z = 0$ cm, respectively, and Neumann-zero conditions were present on all four sides for all charge fluxes and photoionization source terms. Only electrons and generic positive ions were considered, and transport parameters were supplied as empirically fitted expressions given in [50], using the local field approximation. Initial conditions consisted of a gaussian-distributed positive charge density placed at $(r_0, z_0) = (0, 1.0)$ cm following the expression:

$$n_+(t = 0) = N_0 \exp\left[-\frac{(r - r_0)^2 + (z - z_0)^2}{s^2}\right], \quad (31)$$

where $N_0 = 5 \times 10^{18}$ m^{-3} and $s = 0.4$ mm. A uniform background density for both electrons and positive ions was set to 10^9 m^{-3}, and photoionization was enabled with the three-term exponential fitting parameters as provided in [50]. An adaptive timestep between 1 and 3 ps was used, with AMR enabled to perform remeshing every 30 iterations based on the value of the electric field normalized by the maximum field, and of the magnitude of the net space charge. The use of 30 iterations was determined from initial trial-and-error testing, and was found to provide a good balance between solution accuracy and total required computational time (frequent remeshing can be become detrimental to computational speed, but may not necessarily increase the solution accuracy). There currently exist no automated method to determine the optimal number of iterations between AMR, though this would be of interest to develop in the future.

The panels of Fig. 4 are split down the axis of symmetry, where the left and right color plots correspond to the electric field magnitude, and electron density, respectively. Three timesteps of the positive streamer evolution have been shown, at 3, 9, and 15 ns. The streamer length over time, and the maximum field strength over the length of the streamer, has been additionally compared in Fig. 5 to the data from five other groups who participated in [50]. Results from StrAFE compare well to all other codes, and is well within the expected margin of error considering the many potential differences in implementation [50]. The results from StrAFE were found to resemble most closely those from group DE, whose computation was performed using the commercially available COMSOL Multiphysics [19] software. With a run-time of around three to four hours for the present code, and considering imperfect parallel scaling (see Appendix B), StrAFE compares well with this widely-used commercial option.

B. DOUBLE-HEADED COUNTERPROPAGATING STREAMERS

Demonstrated in several studies, e.g., [55], [56], and [57], is the initiation of streamers from an initially charged seed,

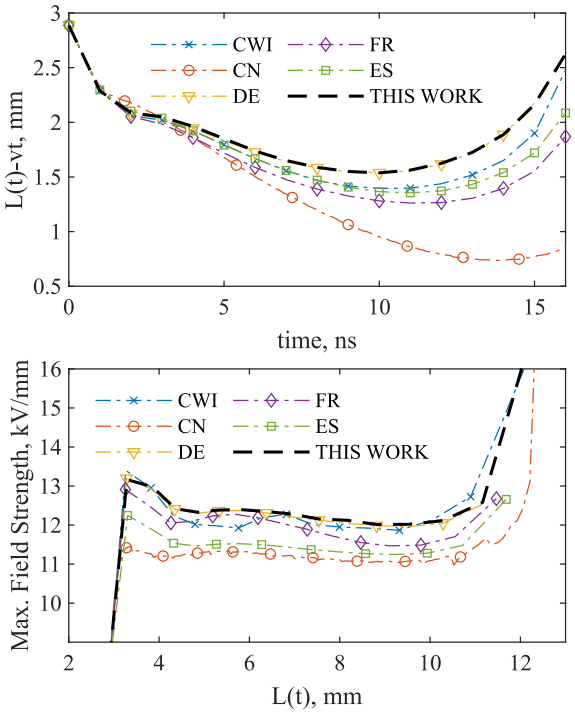

FIGURE 5. Comparison of (top) streamer length over time, where $v = 0.06$ cm ns^{-1}, (bottom) maximum field strength over streamer length with other group data from [50]. The identifying labels for each group correspond to the country or institution of the original authors: CWI = Centrum Wiskunde Informatica, FR = France, CN = China, ES = Spain, DE = Germany. Full details are provided in [50].

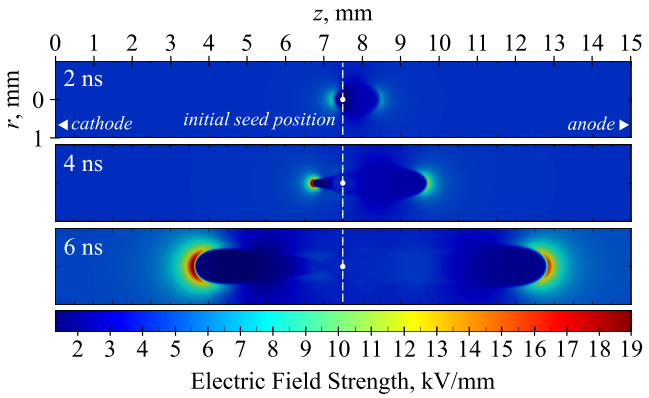

FIGURE 6. Electric field magnitude at $t = 2$, 4, and 6 ns of a double-headed streamer developed in a homogeneous background electric field.

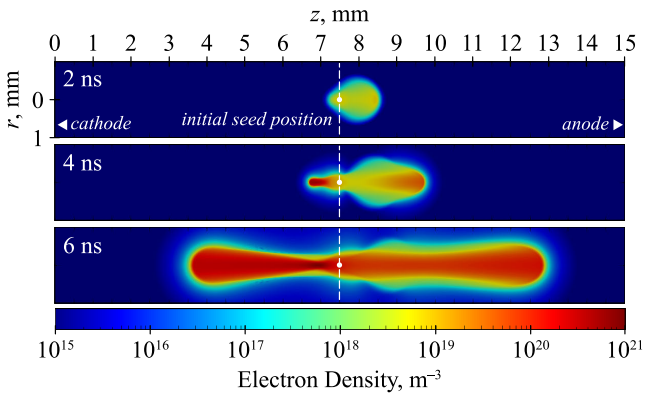

FIGURE 7. Electron density at $t = 2$, 4, and 6 ns of a double-headed streamer developed in a homogeneous background electric field.

which evolves into one positive and one negative streamer propagating in opposite directions, away from the location of the source. The morphology of negative streamers is known to differ significantly from their positive counterparts, as the differences in their sources of electrons have been suggested to result in vastly different characteristics, such as the maximum electric field, streamer radius, and propagation velocity [55]. When both are simultaneously present in a single simulation domain, it can be challenging to find a single set of AMR criteria which is equally effective for both streamers, because in general, the refinement criteria and tolerances require to be adjusted on a case-by-case basis. In these simulations, StrAFE is used with a partitioned-AMR scheme such that two different refinement criteria are used for the top half ($z \geq 7.5$ mm), and bottom half ($z < 7.5$ mm), of the domain, ensuring that sufficiently fine meshes are provided for both streamers. This demonstrates a useful feature which can significantly aid convergence for some simulations, where there exists some prior knowledge of how the discharge will evolve in space. To avoid over-refinement in the early stages of the simulation, the AMR scheme was also configured to be time-dependent. The usage of E/E_{max} is effective as a refinement threshold, but not during the initial stages where $E/E_{max} \approx 1$ everywhere in the domain, due to the presence of a neutral seed. Therefore, this criterion is only introduced once $t \geq 4$ ns.

The domain in this case was once again axisymmetric, forming a cylinder of dimensions $(r, z) = [4], [15]$ mm, with a constant voltage of 65 kV applied at $z = 15$ mm, and 0 kV at $z = 0$ mm. In this study, a simplified set of plasma chemical reactions for air was used, reaction rates following Pancheshnyi and Starikovskii [58], while electron transport parameters were obtained using BOLSIG+ [49] with Phelps' cross-sectional data [59], [60], [61]. These can be found tabulated together in Appendix C. Equal densities of $N_0 = 5\times 10^{18}$ m^{-3} electrons and N_2^+ ions were placed at the center of the domain, again following the gaussian form of (31), but with $s = 0.2$ mm. A background ionisation level of 10^9 m^{-3} was again used, as was photoionization. Fig. 6 shows the electric field distribution at 2, 4, and 6 ns, while Fig. 7 shows the corresponding electron density. The double-headed streamer was successfully resolved, and the propagating characteristics of both positive and negative streamers are in agreement with past studies [55].

C. COUNTERPROPAGATING STREAMERS TOWARD EACH OTHER FROM NEEDLE ELECTRODES UNDER FAST-RISING RAMP VOLTAGE

In practical electrical systems, such as HV power and pulsed power equipment, electrical pre-breakdown and breakdown

phenomena are seldom initiated in regions of homogeneous electric fields. In most cases, highly nonuniform and divergent field conditions, induced by high aspect ratio geometry, field redistribution at triple-junctions, or accumulated space-charge, typically form the most pressing problems in HV insulating system design.

Here, StrAFE's handling of curved geometries in the form of sharp needle electrodes is demonstrated, alongside a time-varying Dirichlet condition representing a fast-rising applied voltage. The simulation in question was initially performed in [62], between two needle electrodes with a curvature radius of 25 μm, and with an interelectrode gap distance of 1.2 mm. Once again, the domain is axisymmetric around $r = 0$. In the original study, a waveform generated using the CST studio suite [63] from their experimental configuration was also used for simulation, but was not made openly available. However, since the simulation was performed only over the initial stages of the discharge, and during the rising edge, an alternative waveform was used in this study, which closely approximates the rising-edge of the original signal, using a linearly increasing ramp voltage of the form:

$$U_0(t) = \frac{dU}{dt}t, \tag{32}$$

where the rate of voltage rise dU/dt was chosen to be 75 kV/ns, to roughly align with the slope of the original waveform from [62]. The plasma chemistry model of Appendix C and photoionization was once again used, with initial background densities of 10^9 m^{-3} for electrons and N_2^+ ions. No initial seed was necessary in this simulation, as the streamers would initiate directly from the electrode tips, where the electric field was maximally enhanced. The LMEA was additionally used in this simulation, with energy-dependent electron transport data computed using BOLSIG+ [49] with Phelps' cross-sectional data [59], [60], [61]. Results are presented in the same format as in [62] to facilitate direct comparison. These include Fig. 8(a) and 8(b), which show streak plots of the electric field magnitude and electron density along the axis of symmetry, respectively; and Fig. 9, which plots the electric field strength along the same axis, at timesteps between $t = 160$ and 260 ps.

Overall, excellent agreement with the results from [62] was found, including the time of streamer inception, the delayed propagation of the negative streamer, the electron density distribution, and propagation velocities. Minor differences are observed in the maximum electric field strength at the negative streamer head and in the position where the streamers collide. As indicated by the difference between the black markers of Fig. 9 and the obtained data, the positive streamer field agrees closely with [62], but there exists a larger discrepancy for peak field values with the negative streamer. However, the difference is within expectation considering the many possible differences in numerical implementation. The discrepencies are believed to be attributed to the differences in the waveform, the numerical implementation, or possible differences in the boundary conditions applied at the electrodes, as these were not specified in [62].

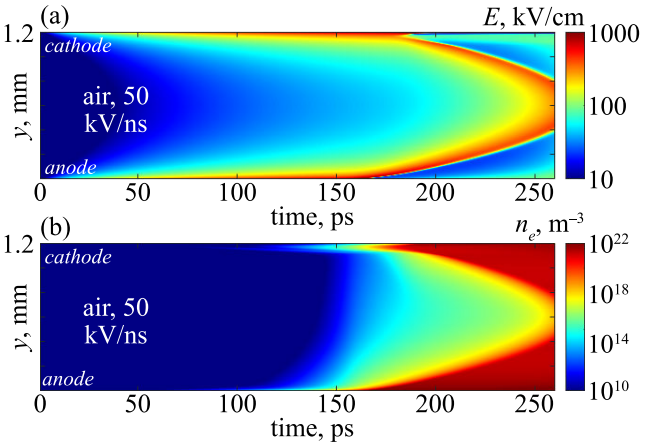

FIGURE 8. Streak images of (a) electric field magnitude, (b) electron density, of counterpropagating streamers developed between needle-needle electrodes of 25 μm radius, generated using StrAFE. This figure corresponds to Fig. 8(a) and 8(b) of [62].

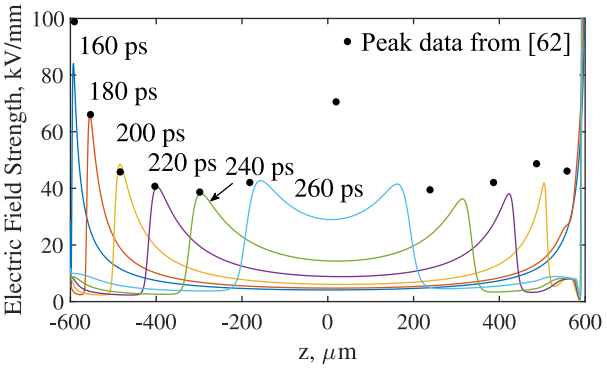

FIGURE 9. Electric field magnitude down the axis of symmetry for various timesteps, during the development of counterpropagating streamers, generated using StrAFE. This figure corresponds to Fig. 9(a) of [62]. Solid black markers indicate the position of the peaks from the simulations conducted in [62].

D. POSITIVE STREAMER WITH SOLID DIELECTRIC

Equally prevalent in, and important to, practical HV systems, are the interactions between nonthermal plasma discharges and dielectric surfaces. Knowledge of these processes would be hugely beneficial to the development of novel plasma technologies, e.g., surface treatment technologies [2], or any other apparatus involving dielectric barrier discharges [7].

Using the described subdomain support in section IV-D, simulations from [17] have been recreated, showing the electrostatic attraction of streamers toward solid dielectric surfaces, and their subsequent propagation across these surfaces. Briefly, the domain consisted of a square geometry with $(x, y) = [40, 40]$ mm. A solid dielectric subdomain was defined where $0 \leq x \leq 10$ mm, and with relative permittivity $\varepsilon_r = 2$, while a neutral seed composed of electrons and positive ions initiated the discharge (plasma chemistry

was not used in this simulation, following the original. Only electrons, generic positive ions, and generic negative ions were included). The seed followed the capsule definition as in [17], placed with its center 0.5 mm away from the solid surface, and near the anode that was set to 100 kV. Townsend coefficients were computed using BOLSIG+ [49] as before. The base mesh was refined along the gas-solid interface, while AMR was enabled to perform a remesh every 30 iterations - this value was once again determined from manual testing. No significant difference in the solution was observed when only 20 iterations per AMR was used, but signs of numerical nonconvergence near the fast-moving streamer head at 40 iterations per AMR were observed. 30 was therefore chosen to reduce computational time and to provide good convergence, yet have minimal effect on the computed solution.

Fig. 10 shows the time evolution of the streamer electron density at $t = 10$, 12.5, 15, and 20 ns. The streamer initiates from the charged seed due to the field enhancement at the bottom of the seed, as electrons drift upward. The streamer then begins to propagate before 10 ns, and is drawn toward the solid dielectric surface, before evolving into a surface streamer. Here, the streamer is observed to gain velocity, and accelerates down the surface, as in [17] and [18]. StrAFE was also shown capable of resolving the thin sheath region between the positive streamer and the surface, as described in [17]. The maximum electric field strength at the streamer head was found to be very similar to that of [17], as was the propagation velocity during the gas-stage and surface streamer stage. The electron density was similarly comparable, with the channel density stabilizing at around 10^{21} m^{-3}. Additional studies on the effects of increasing the solid permittivity were also conducted, results which recreated characteristics as observed in [17], such as the thinner surface streamer with increased ε_r, and a far more rapid surface attachment speed.

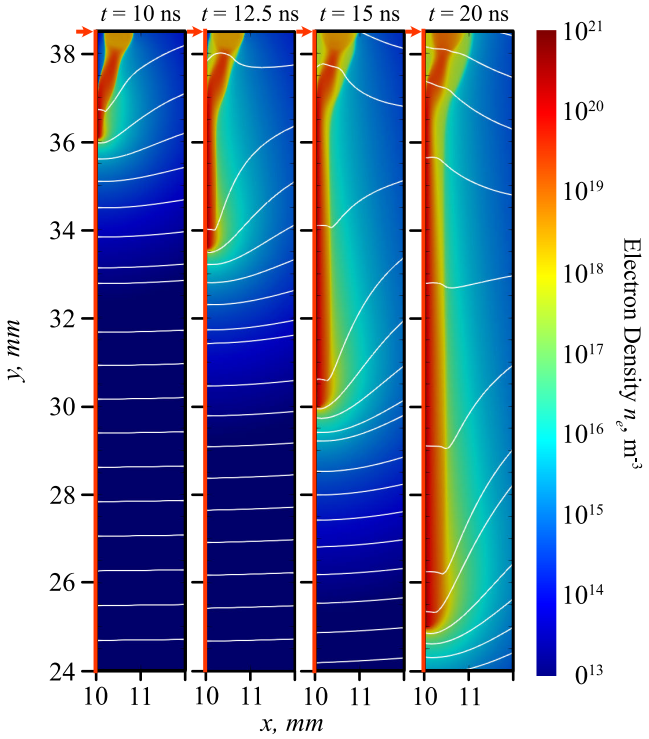

FIGURE 10. Positive streamer attaching to a solid dielectric surface (solid red line), and forming a surface streamer at timesteps $t = 10$, 12.5, 15, and 20 ns, generated using StrAFE. Equipotential lines are spaced by 5 kV. Originally studied in [17].

VI. CONCLUSION

In summary, the present article has introduced and described the StrAFE (**Str**eamers on **A**daptive **F**inite **E**lements) Python library, a powerful automation tool for the study of transient ionization fronts, using the hydrodynamic approximation. Based on the open-source FEniCS framework, this article has presented the design and development of the library, showing how FEniCS and Python can be used to develop an effective platform for the simulation of nonthermal gas discharge plasmas, with emphasis on configurability, ease-of-use, and a high degree of flexibility.

StrAFE was developed in Python, known for its accessibility, due to its English-like syntax. The implementation of a number of dedicated classes allows the substantial simplification of generating transient ionization-FE problems, while also streamlining the process of passing the problem to FEniCS for efficient solving, using its high-speed C++ backend. StrAFE employs the hydrodynamic approach of coupled drift-diffusion-reaction-Poisson equations for nonthermal plasmas, and further includes crucial features necessary for multiscale modelling. These include: user-configurable adaptive mesh refinement and adaptive timestep controllers, support for problem generation from a single plasma chemistry file; support for both local field approximations and local mean energy approximations, and support for subdomains, e.g., solid dielectric materials. Since StrAFE inherits from FEniCS, it additionally offers full support for distributed memory parallelism through the message passing interface (MPI), and the same desktop-developed prototype code can be executed in HPC settings with little to no change.

This article has importantly showed how it can be used to significantly simplify the process of simulating streamer discharges. Several simulation examples have been presented in various domains, and were of varying complexity, which demonstrated the capabilities of StrAFE through comparison with existing work. It was found that the use of FEniCS compares very well to a range of existing and well-established codes, both commercial and custom-made. Yet, its open-source nature offers far greater flexibility and code transparency, and can be used to develop user-friendly software with a very low entry barrier. Such innovations are exceptionally well-suited for interested parties who may be new, or unfamiliar, with the field of gas discharge modelling, and thus possesses immense educational value. From this study, it can be concluded that open-source frameworks such

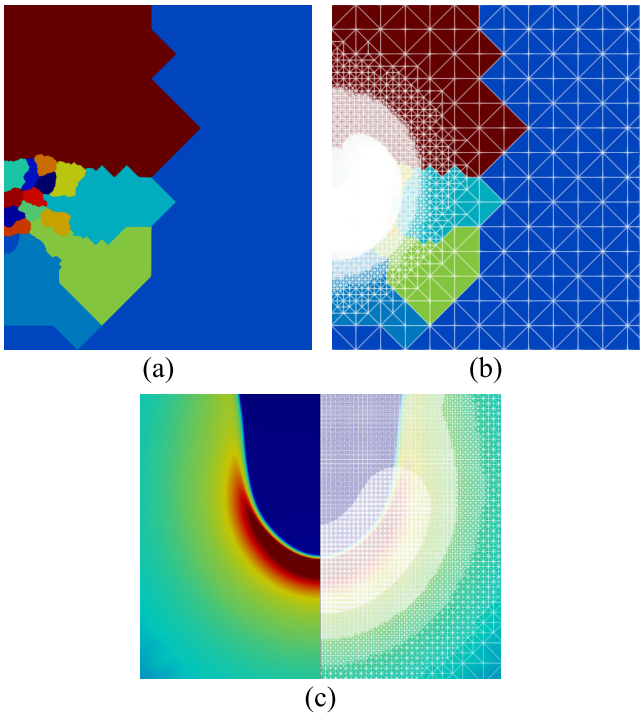

FIGURE 11. (a) Load balanced mesh, each color demarcates nodes which are owned by a unique computational process, (b) adaptive mesh around an initially gaussianly-distributed charge seed, (c) example of the dynamic AMR scheme tracking a streamer head, overlaid on top of the electric field color map.

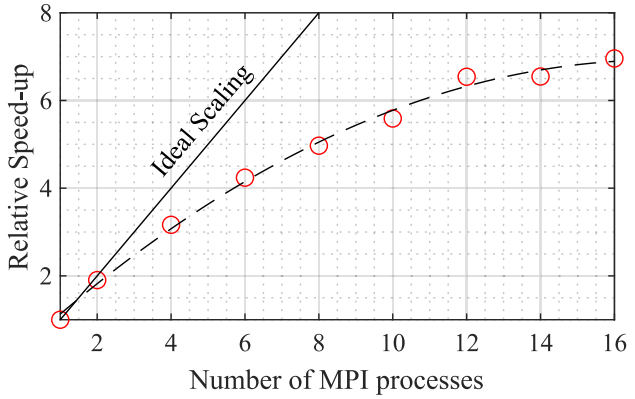

FIGURE 12. Data from a basic parallel scaling test. Red circles are measured data, dashed black line is a best fit curve, and the solid black line represents ideal scaling.

as FEniCS, have great potential to make significant contributions to the study of nonthermal plasmas, ionization fronts, and gas discharges; and can help to further increase the accessibility of computational modelling of complex phenomena in engineering and science.

In future, StrAFE will be used to explore nonthermal plasma dynamics in practical topologies of interest. The flexibility offered by the library further facilitates experimentation with different numerical schemes, the exploration of higher-order discretization, the possibility to use discontinuous Galerkin methods (DG-FEM), additional physics, and more. It is also of high interest to conduct three-dimensional simulations in the future, and to further evaluate StrAFE in HPC settings – 3D features have theoretically been implemented, but not yet thoroughly tested.

TABLE 1. Plasma chemical reactions used in example simulations for 80/20 air. Notation $f(\bar{\varepsilon})$ means that the reaction rate was a function of the local electron energy. Chemical symbol M represents O_2 or N_2.

R	Reaction	Rate	Unit	Ref.
R_1	$e + N_2 \rightarrow N_2^+ + e + e : 15.6\,\text{eV}$	BOLSIG+	m^3s^{-1}	[49]
R_2	$e + N_2 \rightarrow N_2^+ + e + e : 18.8\,\text{eV}$	BOLSIG+	m^3s^{-1}	[49]
R_3	$e + O_2 \rightarrow O_2^+ + e + e$	BOLSIG+	m^3s^{-1}	[49]
R_4	$e + O_2 + O_2 \rightarrow O_2^- + O_2$	$f(\bar{\varepsilon})$	m^6s^{-1}	[58]
R_5	$N_2^+ + N_2 + M \rightarrow N_4^+ + M$	5×10^{-41}	m^6s^{-1}	[58]
R_6	$N_4^+ + O_2 \rightarrow O_2^+ + N_2 + N_2$	2.5×10^{-16}	m^3s^{-1}	[58]
R_7	$N_2^+ + O_2 \rightarrow O_2^+ + N_2$	6×10^{-17}	m^3s^{-1}	[58]
R_8	$O_2^+ + N_2 + N_2 \rightarrow O_2^+ N_2 + N_2$	9×10^{-43}	m^6s^{-1}	[58]
R_9	$O_2^+ N_2 + N_2 \rightarrow O_2^+ + N_2 + N_2$	4.3×10^{-16}	m^3s^{-1}	[58]
R_{10}	$O_2^+ N_2 + O_2 \rightarrow O_4^+ N_2$	1×10^{-15}	m^3s^{-1}	[58]
R_{11}	$O_2^+ + O_2 + M \rightarrow O_4^+ + M$	2.4×10^{-42}	m^6s^{-1}	[58]
R_{12}	$e + O_4^+ \rightarrow O_2 + O_2$	$f(\bar{\varepsilon})$	m^3s^{-1}	[58]
R_{13}	$e + O_2^+ \rightarrow O + O$	$f(\bar{\varepsilon})$	m^3s^{-1}	[58]
R_{14}	$O_2^- + O_4^+ \rightarrow 3O_2$	1×10^{-13}	m^3s^{-1}	[58]
R_{15}	$O_2^- + O_4^+ + M \rightarrow 3O_2 + M$	2×10^{-37}	m^6s^{-1}	[58]
R_{16}	$O_2^- + O_2^+ + M \rightarrow O_2 + O_2 + M$	2×10^{-37}	m^6s^{-1}	[58]
R_{17}	$e + N_2 \rightarrow e + N_2 + \gamma$	Zheleznyak.	–	[44]
R_{18}	$\gamma + O_2 \rightarrow e + O_2^+$	Zheleznyak.	–	[44]

APPENDIX A
EXAMPLE OF AMR AND LOAD BALANCING

Fig. 11(a) shows a visualization of the load-balanced mesh generated by FEniCS, around the initial seed of the example from section V-A. Fig. 11(b) shows the corresponding mesh around the initial seed region. Mesh cells belonging to the same color group are owned by the same MPI process. Fig. 11(c) additionally shows the adaptive mesh overlaid on half of the domain at $t = 8$ ns, showing the fine mesh tracking the streamer head.

APPENDIX B
PARALLEL SCALING

Fig. 12 shows the results of a basic parallel scaling test, up to 16 MPI processes, conducted by varying the number of processes the program ran with, when simulating example V-A. Note that only physical cores were used here, the use of virtual cores was disabled, since it did not appear to offer any additional computational speedup.

APPENDIX C
PLASMA CHEMISTRY FOR AIR

Table 1 encloses the plasma chemical reactions for air, which were included in several of the simulations presented in section V.

REFERENCES

[1] L. Bárdos and H. Baránková, "Cold atmospheric plasma: Sources, processes, and applications," *Thin Solid Films*, vol. 518, no. 23, pp. 6705–6713, Sep. 2010.

[2] Z. Fang, X. Qiu, Y. Qiu, and E. Kuffel, "Dielectric barrier discharge in atmospheric air for glass-surface treatment to enhance hydrophobicity," *IEEE Trans. Plasma Sci.*, vol. 34, no. 4, pp. 1216–1222, Aug. 2006.

[3] H.-H. Kim, "Nonthermal plasma processing for air-pollution control: A historical review, current issues, and future prospects," *Plasma Process. Polym.*, vol. 1, no. 2, pp. 91–110, 2004.

[4] K. Urashima and J.-S. Chang, "Removal of volatile organic compounds from air streams and industrial flue gases by non-thermal plasma technology," *IEEE Trans. Dielectr. Electr. Insul.*, vol. 7, no. 5, pp. 602–614, Oct. 2000.

[5] M. Rong, J. Liu, X. Wang, and X. Yuan, "Research on air purification efficiency by nonthermal plasma along with the application of magnetic field," *IEEE Trans. Plasma Sci.*, vol. 34, no. 4, pp. 1345–1350, Aug. 2006.

[6] M. Laroussi, S. Bekeschus, M. Keidar, A. Bogaerts, A. Fridman, X. Lu, K. Ostrikov, M. Hori, K. Stapelmann, V. Miller, S. Reuter, C. Laux, A. Mesbah, J. Walsh, C. Jiang, S. M. Thagard, H. Tanaka, D. Liu, D. Yan, and M. Yusupov, "Low-temperature plasma for biology, hygiene, and medicine: Perspective and roadmap," *IEEE Trans. Radiat. Plasma Med. Sci.*, vol. 6, no. 2, pp. 127–157, Feb. 2022.

[7] R. Aerts, W. Somers, and A. Bogaerts, "Carbon dioxide splitting in a dielectric barrier discharge plasma: A combined experimental and computational study," *ChemSusChem*, vol. 8, no. 4, pp. 702–716, 2015.

[8] Y. P. Raizer, G. M. Milikh, M. N. Shneider, and S. V. Novakovski, "Long streamers in the upper atmosphere above thundercloud," *J. Phys. D, Appl. Phys.*, vol. 31, no. 22, pp. 3255–3264, Nov. 1998.

[9] S. B. Leonov, I. V. Adamovich, and V. R. Soloviev, "Dynamics of near-surface electric discharges and mechanisms of their interaction with the airflow," *Plasma Sources Sci. Technol.*, vol. 25, no. 6, Nov. 2016, Art. no. 063001.

[10] U. Ebert, "The multiscale nature of streamers," *Plasma Sources Sci. Technol.*, vol. 15, no. 2, pp. 118–129, Apr. 2006.

[11] F. Boakye-Mensah, N. Bonifaci, R. Hanna, I. Niyonzima, and I. Timoshkin, "Modelling of positive streamers in SF_6 gas under non-uniform electric field conditions: Effect of electronegativity on streamer discharges," *J*, vol. 5, no. 2, pp. 255–276, May 2022.

[12] Z. Zhao, X. Wei, S. Song, L. Cui, Z. Zhang, and K. Yang, "A two-dimensional air discharge modified model under unipolar square pulse voltage at low temperature and sub-atmospheric pressure," *IEEE Access*, vol. 9, pp. 51896–51909, 2021.

[13] J. Teunissen and U. Ebert, "Simulating streamer discharges in 3D with the parallel adaptive Afivo framework," *J. Phys. D, Appl. Phys.*, vol. 50, no. 47, Oct. 2017, Art. no. 474001.

[14] R. Marskar, "Adaptive multiscale methods for 3D streamer discharges in air," *Plasma Res. Exp.*, vol. 1, no. 1, Jan. 2019, Art. no. 015011.

[15] C. Xu, N. Huret, M. Garnung, and S. Celestin, "A new detailed plasma-chemistry model for the potential impact of blue jet streamers on atmospheric chemistry," *J. Geophys. Res., Atmos.*, vol. 125, no. 6, Mar. 2020, Art. no. e2019JD031789.

[16] L. Zhao, Y. Zang, W. Liu, Y. Qian, and X. Zhou, "Two-dimensional simulation of transition from primary to secondary streamer discharge in air," *AIP Adv.*, vol. 9, no. 9, Sep. 2019, Art. no. 095028.

[17] X. Li, A. Sun, G. Zhang, and J. Teunissen, "A computational study of positive streamers interacting with dielectrics," *Plasma Sources Sci. Technol.*, vol. 29, no. 6, Jun. 2020, Art. no. 065004.

[18] X. Li, A. Sun, and J. Teunissen, "A computational study of negative surface discharges: Characteristics of surface streamers and surface charges," *IEEE Trans. Dielectr. Electr. Insul.*, vol. 27, no. 4, pp. 1178–1186, Aug. 2020.

[19] A. P. Jovanovic, M. N. Stankov, D. Loffhagen, and M. M. Becker, "Automated fluid model generation and numerical analysis of dielectric barrier discharges using comsol," *IEEE Trans. Plasma Sci.*, vol. 49, no. 11, pp. 3710–3718, Nov. 2021.

[20] M. Alnaes, "The FEniCS project version 1.5," *Arch. Numer. Softw.*, vol. 3, no. 100, pp. 1–15, Dec. 2015.

[21] A. Logg, *Automated Solution of Differential Equations by the Finite Element Method*. Cham, Switzerland: Springer, 2012.

[22] *The FEniCS Project*. Accessed: Oct. 31, 2022. [Online]. Available: fenicsproject.org

[23] O. Ducasse, O. Eichwald, and M. Yousfi, "Finite vol. method, for streamer and gas dynamics modelling in air discharges at atmospheric pressure," in *Finite Volume Method Power Means of Engineering Design*. London, U.K.: IntechOpen, 2012.

[24] G. Dhatt, G. Touzot, and E. Lefrancois, *Finite Element Method*. Hoboken, NJ, USA: Wiley, 2012.

[25] S. S. Rao, *The Finite Element Method in Engineering*. Amsterdam, The Netherlands: Elsevier, 2018.

[26] K. B. Ølgaard and G. N. Wells, "Optimizations for quadrature representations of finite element tensors through automated code generation," *ACM Trans. Math. Softw.*, vol. 37, no. 1, pp. 1–23, Jan. 2010.

[27] R. C. Kirby and A. Logg, "A compiler for variational forms," *ACM Trans. Math. Softw.*, vol. 32, no. 3, pp. 417–444, Sep. 2006.

[28] M. S. Alnaes, A. Logg, K. B. Ølgaard, M. E. Rognes, and G. N. Wells, "Unified form language: A domain-specific language for weak formulations of partial differential equations," *ACM Trans. Math. Softw.*, vol. 40, no. 2, pp. 1–37, Feb. 2014.

[29] R. C. Kirby, "Algorithm 839: Fiat, a new paradigm for computing finite element basis functions," *ACM Trans. Math. Softw.*, vol. 30, no. 4, pp. 502–516, 2004.

[30] C. Chevalier and F. Pellegrini, "PT-scotch: A tool for efficient parallel graph ordering," *Parallel Comput.*, vol. 34, nos. 6–8, pp. 318–331, Jul. 2008.

[31] G. Karypis and K. Schoegel, "ParMETIS: Parallel graph partitioning and sparse matrix ordering library, version 4.0," Dept. Comput. Sci. Eng., Univ. Minnesota, Minneapolis, MN, USA, Mar. 2013.

[32] S. Balay, W. D. Gropp, L. C. McInnes, and B. F. Smith, "Efficient management of parallelism in object oriented numerical software libraries," in *Modern Software Tools in Scientific Computing*. Boston, MA, USA: Birkhauser, 1997.

[33] S. Balay. (2022). *PETSc Web Page*. Accessed: Oct. 31, 2022. [Online]. Available: https://petsc.org

[34] J. Walter, M. Koch, G. Winkler and D. Bellot. (2000). *uBLAS Webpage*. Accessed: Oct. 31, 2022. [Online]. Available: https://www.boost.org/doc/libs/1_80_0/libs/numeric/ublas/doc/index.html

[35] (2022). *Epetra Web Page*. Accessed: Oct. 31, 2022. [Online]. Available: https://trilinos.github.io/epetra.html

[36] L. Vynnytska, M. E. Rognes, and S. R. Clark, "Benchmarking FEniCS for mantle convection simulations," *Comput. Geosci.*, vol. 50, pp. 95–105, Jan. 2013.

[37] M. A. Rodriguez, C. M. Augustin, and S. C. Shadden, "FEniCS mechanics: A package for continuum mechanics simulations," *SoftwareX*, vol. 9, pp. 107–111, Jan. 2019.

[38] S. Natarajan and R. K. Annabattula, "A FEniCS implementation of the phase field method for quasi-static brittle fracture," *Frontiers Struct. Civil Eng.*, vol. 13, no. 2, pp. 380–396, Apr. 2019.

[39] J. Hoffman, J. Jansson, and N. Jansson, "FEniCS-HPC: Automated predictive high-performance finite element computing with applications in aerodynamics," in *Parallel Processing and Applied Mathematics*. Cham, Switzerland: Springer, 2016.

[40] I. Rafatov and A. Kudryavtsev, *Introduction to Simulation Methods for Gas Discharge Plasmas*. Bristol, U.K.: IOP Publishing, 2020.

[41] M. Zheleznyak and S. Sizykh, "Photo-ionization of nitrogen and oxygen mixtures by radiation from a gas-discharge," *High Temp.*, vol. 20, pp. 357–362, Nov. 1982.

[42] A. A. Kulikovsky, "The role of photoionization in positive streamer dynamics," *J. Phys. D, Appl. Phys.*, vol. 33, no. 12, p. 1514, 2000.

[43] G. Wormeester, S. Pancheshnyi, A. Luque, S. Nijdam, and U. Ebert, "Probing photo-ionization: Simulations of positive streamers in varying n2: O2-mixtures," *J. Phys. D, Appl. Phys.*, vol. 43, no. 50, Dec. 2010, Art. no. 505201.

[44] A. Bourdon, "Efficient models for photoionization produced by non-thermal gas discharges in air based on radiative transfer and the Helmholtz equations," *Plasma Sources Sci. Technol.*, vol. 16, p. 656, Aug. 2007.

[45] G. K. Grubert, M. M. Becker, and D. Loffhagen, "Reasons for the usage of the local-mean-energy approximation instead of the local-field-approximation," in *Proc. 19th Intl. Symp. Plasma Chem.*, Bochum, Germany, Jul. 2009, pp. 1–4.

[46] G. K. Grubert, M. M. Becker, and D. Loffhagen, "Why the local-mean-energy approximation should be used in hydrodynamic plasma descriptions instead of the local-field approximation," *Phys. Rev. E, Stat. Phys. Plasmas Fluids Relat. Interdiscip. Top.*, vol. 80, no. 3, Sep. 2009, Art. no. 036405.

[47] G. J. M. Hagelaar, F. J. De Hoog, and G. M. W. Kroesen, "Boundary conditions in fluid models of gas discharges," *Phys. Rev. E, Stat. Phys. Plasmas Fluids Relat. Interdiscip. Top.*, vol. 62, no. 1, p. 1452, 2000.

[48] A. Logg and G. N. Wells, "DOLFIN: Automated finite element computing," *ACM Trans. Math. Softw.*, vol. 37, no. 2, pp. 1–28, Apr. 2010.

[49] G. J. M. Hagelaar and L. C. Pitchford, "Solving the Boltzmann equation to obtain electron transport coefficients and rate coefficients for fluid models," *Plasma Sources Sci. Technol.*, vol. 14, no. 4, p. 722, Oct. 2005.

[50] 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 Sci. Technol.*, vol. 27, no. 9, Sep. 2018, Art. no. 095002.
[51] C. Geuzaine and J.-F. Remacle, "Gmsh: A 3-D finite element mesh generator with built-in pre- and post-processing facilities," *Int. J. Numer. Methods Eng.*, vol. 79, no. 11, pp. 1309–1331, Sep. 2009.
[52] R. D. Falgout and U. M. Yang, "Hypre: A library of high performance preconditioners," in *Proc. Int. Conf. Comput. Sci.*, 2002, pp. 632–641.
[53] P. R. Amestoy, I. S. Duff, J.-Y. L'Excellent, and J. Koster, "A fully asynchronous multifrontal solver using distributed dynamic scheduling," *SIAM J. Matrix Anal. Appl.*, vol. 23, no. 1, pp. 15–41, 2001.
[54] D. Merkel, "Docker: Lightweight Linux containers for consistent development and deployment," *Linux J.*, vol. 239, p. 2, Mar. 2014.
[55] A. Luque, V. Ratushnaya, and U. Ebert, "Positive and negative streamers in ambient air: Modelling evolution and velocities," *J. Phys. D, Appl. Phys.*, vol. 41, no. 23, Nov. 2008, Art. no. 234005.
[56] J. Qin, S. Celestin, and V. P. Pasko, "Formation of single and double-headed streamers in sprite-halo events," *Geophys. Res. Lett.*, vol. 39, no. 5, Mar. 2012, Art. no. L05810.
[57] M. B. Garnung, S. Celestin, and T. Farges, "HF-VHF electromagnetic emissions from collisions of sprite streamers," *J. Geophys. Res., Space Phys.*, vol. 126, no. 6, Jun. 2021, Art. no. e2020JA028824.
[58] S. V. Pancheshnyi and A. Y. Starikovskii, "Two-dimensional numerical modelling of the cathode-directed streamer development in a long gap at high voltage," *J. Phys. D, Appl. Phys.*, vol. 36, no. 21, p. 2683, 2003.
[59] *Phelps Database*. Lxcat. Accessed: Oct. 31, 2022. [Online]. Available: www.lxcat.net/Phelps
[60] A. V. Phelps and L. C. Pitchford, "Anisotropic scattering of electrons by N_2 and its effect on electron transport," *Phys. Rev. A,* vol. 31, no. 5, p. 2932, May 1985.
[61] S. A. Lawton and A. V. Phelps, "Excitation of the b1Σ+G state of O_2 by low energy electrons," *J. Chem. Phys.*, vol. 69, no. 3, p. 1055, 1978.
[62] H. Höft, M. M. Becker, J. F. Kolb, and T. Huiskamp, "Double-propagation mode in short-gap spark discharges driven by HV pulses with sub-ns rise time," *Plasma Sources Sci. Technol.*, vol. 29, no. 8, Aug. 2020, Art. no. 085002.
[63] *CST Studio Suite Electromagnetic Field Simulation Software*, Simulia, Dassault Systémes, Johnston, RI, USA, 2022.

TIMOTHY WONG (Graduate Student Member, IEEE) received the M.Eng. degree in electrical and mechanical engineering with international study from the University of Strathclyde, Glasgow, U.K., in 2020, where he is currently pursuing the Ph.D. degree in electronic and electrical engineering with the High Voltage Technologies research group. His current research interests include the pulsed breakdown of solid–solid dielectric interfaces, solid–gas interfaces, computational modeling of fast ionization fronts and streamer discharges in gas and gas–solid topologies, and pulsed power insulation systems. He is currently a Graduate Member of the Dielectrics and Electrical Insulation Society (DEIS) and the Nuclear and Plasma Sciences Society (NPSS). He was a recipient of the IMechE Student Award, in 2020.

IGOR TIMOSHKIN (Senior Member, IEEE) received the degree in physics from Moscow State University, Moscow, Russia, in 1992, and the Ph.D. degree from the Imperial College of Science, Technology, and Medicine (ICSTM), London, U.K., in 2001. He was a Researcher with Moscow State Agro Engineering University, Moscow, and the Institute for High Temperatures of Russian Academy of Sciences, Moscow. In 1997, he joined ICSTM. He joined the Department of Electronic and Electrical Engineering, University of Strathclyde, Glasgow, U.K., in 2001, where he became a Reader, in 2016. His research interests include dielectric materials, pulsed power, transient spark discharges, and environmental applications of non-thermal plasma discharges. He was a Voting Member of the Pulsed Power Science and Technology Committee of the IEEE Nuclear and Plasma Science Society (2017–2021). He is currently a member of the International Advisory Committee of the IEEE Conference on Dielectric Liquids, a member of the International Scientific Committee of the Gas Discharges and Their Applications Conference, a Subject Editor of *IET Nanodielectrics*, and a member of the Editorial Board of *MDPI Energies*.

SCOTT MACGREGOR (Senior Member, IEEE) received the B.Sc. and Ph.D. degrees from the University of Strathclyde, Glasgow, U.K., in 1982 and 1986, respectively. He was a fellow of pulsed-power research, in 1986, and a Lecturer in pulsed-power technology, in 1989. In 1994, he became a Senior Lecturer, with a promotion to Reader and a Professor of high voltage engineering, in 1999 and 2001, respectively. In 2006 and 2010, he became the Head of the Department of Electronic and Electrical Engineering and the Executive Dean of the Faculty of Engineering. He has been the Vice-Principal with the University of Strathclyde, since 2014. His current research interests include high-voltage pulse generation, high-frequency diagnostics, high-power repetitive switching, high-speed switching, electronic methods for food pasteurization and sterilization, the generation of high-power ultrasound (HPU), plasma channel drilling, pulsed-plasma cleaning of pipes, and the stimulation of oil wells with HPU. He was a recipient of the 2013 IEEE Peter Haas Award. He was an Associate Editor of the IEEE TRANSACTIONS ON DIELECTRICS AND ELECTRICAL INSULATION, in 2015.

MARK WILSON (Member, IEEE) was born in Stranraer, Scotland, in 1982. He received the B.Eng. (Hons.), M.Phil., and Ph.D. degrees in electronic and electrical engineering from the University of Strathclyde, Glasgow, U.K., in 2004, 2007, and 2011, respectively. He is currently with the High Voltage Technologies research group, University of Strathclyde. His research interests include interfacial surface flashover, nanodielectrics, practical applications of high-power ultrasound, corona discharges, and pulsed electric fields. He is a member of the IEEE Nuclear and Plasma Science Society, IEEE Dielectrics and Electrical Insulation Society, and the IET. He received the Graduate Scholarship Award from the IEEE Nuclear and Plasma Science Society, in 2011.

MARTIN GIVEN (Senior Member, IEEE) received the B.Sc. degree in physics from the University of Sussex, Brighton, U.K., in 1981, and the Ph.D. degree in electronic and electrical engineering from the University of Strathclyde, Glasgow, U.K., in 1996. He is currently a Senior Lecturer with the Department of Electronic and Electrical Engineering, University of Strathclyde. His current research interests include aging processes and condition monitoring in solid and liquid insulation systems, high-speed switching, and pulsed power.