A spectral element method for modelling streamer discharges in low-temperature atmospheric-pressure plasmas

I. L. Semenov, K.-D. Weltmann

Leibniz Institute for Plasma Science and Technology, Felix-Hausdorff-Str. 2, 17489 Greifswald, Germany

Abstract

Streamer discharges are ionization fronts that occur in gases at atmospheric and sub-atmospheric pressures. Numerical studies of streamers are important for practical applications but are challenging due to the multiscale nature of this discharge type. This paper introduces a spectral element method for modelling streamer discharges. The method is developed for Cartesian grids but can be extended to be used on unstructured meshes. The streamer model is based on the Poisson equation for the electric potential and the electron continuity equation. The Poisson equation is discretized via a spectral method based on the integral representation of the solution. The hierarchical Poincaré-Steklov (HPS) scheme is used to solve the resulting set of equations. The electron continuity equation is solved by means of the discontinuous Galerkin spectral element method (DGSEM). The DGSEM is extended by an alternative definition of the diffusion flux. A subcell finite volume method is used to stabilize the DGSEM scheme, if required. The entire simulation scheme is validated by solving a number of test problems and reproducing the results of previous studies. Adaptive mesh refinement is used to reduce the number of unknowns. The proposed method is found to be sufficiently fast for being used in practical applications. The flexibility of the method provides an interesting opportunity to broaden the range of problems that can be addressed in numerical studies of low-temperature plasma discharges.

Keywords: streamer discharge, low-temperature plasma, spectral element method, hierarchical Poincaré-Steklov scheme, discontinuous Galerkin method

1. Introduction

Non-equilibrium plasma discharges generated at atmospheric and sub-atmospheric pressures have been a topic of intense research over the last decades. Progress in the development of novel plasma sources and methods for diagnostic of non-equilibrium plasmas has stimulated work on new applications such as plasma assisted combustion [1, 2], flow control by non-thermal plasma actuators [3, 4], plasma medicine [5], plasma treatment of liquids [6], etc. Experimental studies of plasma discharges are usually complemented with the results of numerical simulations. However, numerical studies of non-equilibrium atmospheric pressure plasmas are still highly challenging. This is mainly due to the multiscale nature of this discharge type.

One example is the problem of simulating streamer discharges. Streamers are fast ionization fronts that can occur in a gas with low conductivity exposed to a high electric field [7]. Formation of such fronts can be viewed as an elementary process inherent to various types of atmospheric pressure discharges. The streamers are typically simulated using fluid models based on the transport equations for plasma components, the Poisson equation for the electric...
potential and, if considered, the Helmholtz equation for the photoionization source term. The characteristic discretiza-
tion length required to resolve the streamer front is of the order of several micrometers. This length scale is much
smaller than the characteristic length of streamer propagation, which varies from several millimeters to centimeters.
As a consequence, the computational time for streamer simulations based on the conventional finite element or finite
volume methods can reach tens of hours [8] or several days [9, 10]. Note that this is partially explained by the use
of implicit time integration methods in some simulations. This choice is not always optimal. In many cases the time
step required to resolve the streamer propagation satisfies the stability conditions of explicit time integration meth-
ods. When explicit time-stepping is used, the solution of the Poisson and Helmholtz equations becomes the most
time-consuming part of the simulation procedure.
To date, the most efficient numerical codes for simulating streamer discharges combine explicit finite volume
schemes for solving the transport equations and iterative methods for solving the Poisson and Helmholtz equations
[11, 12, 13, 14, 15]. In addition, adaptive mesh refinement [11, 12, 14] and massive parallelization [13, 14, 15] are
used to reduce the required computational time. However, these codes are mainly designed for large-scale simulations
on Cartesian grids. This makes it difficult to use them for simulations of plasma discharges in complex geometries and,
consequently, limits their use in engineering applications. The commercially available simulation packages overcome
this disadvantage, e.g., by using the conventional finite element method (typically, combined with the implicit time
integration methods). But the computational efficiency of these packages is rather low, when they are applied to the
considered type of problems. Thus, further efforts are needed to overcome the limitations of the existing simulation
frameworks.
Recent progress in the development of fast and flexible spectral element methods for second-order partial differ-
ential equations [16, 17, 18, 19, 20] provides an interesting opportunity to extend the capabilities of low-temperature
plasma simulations. Indeed, the method based on the hierarchical Poincaré - Steklov (HPS) scheme, first proposed
by Martinsson in [16], has a number of attractive features. Its complexity is comparable to that of the existing direct
solvers for sparse linear systems [16, 17], it is able to treat complex geometries [16, 20], enables the use of adaptive
mesh refinement [19] and is easy to implement and parallelize. In view of its application to modelling of low temper-
ature plasmas, this method can be used for computing the electric potential, photoionization source terms and solving
the transport equations for plasma components. Moreover, the HPS scheme naturally supports domain decomposition
 technique. This property may be useful for handling thin boundary layers on electrode surfaces. Also, using the
combination of the HPS scheme and one of the explicit high-resolution schemes for advection-diffusion problems,
it could be possible to construct an accurate and flexible numerical method for high-fidelity simulations of transient
non-equilibrium plasma discharges.
The present work is the first step in the construction of such a method. The primary goal of this work is to assess
the potential of up-to-date spectral element methods for being applied to simulations of low-temperature plasmas. As
an example, we introduce a spectral element method for modelling streamer discharges in unbounded domains (i.e.,
without considering a specific electrode configuration). For simplicity, a two-dimensional (axisymmetric) problem is
considered and the method is described for the case of a Cartesian grid. This framework is useful for introducing the
key components of the method and performing the proof-of-concept experiments.
Also, a simple fluid model is used to describe the streamer. The model is based on the electron continuity equation
and the Poisson equation for the electric potential. The transport of ions is neglected. The electron flux is defined
using the drift-diffusion approximation. The local field approximation [7] is used to determine the electron transport
coefficients. The photoionization process is not considered.
The Poisson equation is solved by means of a spectral element method similar to that described in [16, 17, 19].
We propose an alternative procedure to construct the solution operator and the Dirichlet-to-Neumann (DtN) operator
for a single finite element. Our approach is based on the integral representation of the potential and can be viewed
as an extension of the ideas presented in [21, 22]. In contrast to the spectral collocation method used in [16, 17, 19],
our approach provides a more universal way to separate the internal and boundary degrees of freedom. This simplifies
the implementation of the HPS scheme (namely, the handling of finite element corners) and makes the method more
flexible. The accuracy of our method is shown to be comparable with the accuracy of the spectral collocation method.
The other steps of the simulation scheme are the same as described in [16]. The proposed approach is validated by
performing a number of test computations.

The electron continuity equation is solved by means of an explicit discontinuous Galerkin spectral element method (DGSEM). A subcell finite volume scheme is employed to stabilize the DGSEM scheme, if required. The approach we use is similar to that presented in [23]. The main difference is that we introduce an alternative procedure to approximate the diffusion flux. In our approach, the approximate solutions in the neighbouring elements are projected onto the space of functions continuous up to the first derivative on the interelement boundary. The general idea of this approach was first proposed in [20]. The consistency of the proposed formulation is confirmed by solving a number of test problems for a one-dimensional advection-diffusion equation.

The use of the DGSEM is motivated by several aspects. First, the DGSEM can be effectively used on unstructured meshes [23, 24]. This property is attractive to make the simulation scheme suitable for practical applications. Second, the high accuracy of the DGSEM enables the number of unknowns to be reduced in some cases. This, among other things, reduces the computational cost of the HPS scheme. Finally, assessing the combination of the HPS and DGSEM schemes for the considered type of problems may be of general interest.

The developed spectral element method is applied to the test problems introduced in [8, 15]. In both cases, adaptive mesh refinement is used to reduce the required computational time. The obtained results are validated against the results of previous simulations. The computational efficiency of our method is discussed and compared to that of other simulation codes.

The structure of the paper is as follows. The governing equations of the streamer model are presented in section 2. The key components of the proposed spectral element method are described in section 3. The results of numerical experiments are discussed in section 4. The conclusions are drawn in section 5.

2. Basic equations

This section introduces the governing equations of the streamer model. The model is formulated in cylindrical coordinates \((x, y)\), with \(x, y\) being the radial distance and axial coordinate, respectively. The continuity equation for electrons reads

\[
\frac{\partial n_e}{\partial t} + \frac{\partial j_e^{(x)}}{\partial x} + \frac{\partial j_e^{(y)}}{\partial y} + \frac{j_e^{(x)}}{x} = \kappa_e n_e,
\]

(1)

where \(t\) is time, \(n_e\) is the electron number density, \(\kappa_e\) is the ionization frequency and \(j_e^{(x)}, j_e^{(y)}\) are the electron flux components. Using the drift-diffusion approximation, the electron flux components are given by

\[
\begin{align*}
  j_e^{(x)} &= -\mu_e E_e^{(x)} n_e - D_e^{(x)} \frac{\partial n_e}{\partial x}, \\
  j_e^{(y)} &= -\mu_e E_e^{(y)} n_e - D_e^{(y)} \frac{\partial n_e}{\partial y},
\end{align*}
\]

(2)

where \(\mu_e\) is the electron mobility, \(E_e^{(x)}, E_e^{(y)}\) are the components of the electric field and \(D_e^{(x)}, D_e^{(y)}\) are the transverse and longitudinal electron diffusion coefficients, respectively. According to the local field approximation, the transport coefficients \(\mu_e, D_e^{(x)}, D_e^{(y)}\) and the ionization frequency \(\kappa_e\) are considered as functions of \(|\vec{E}|\), where \(\vec{E} = (E_e^{(x)}, E_e^{(y)})^T\). The corresponding dependencies are obtained by solving the simplified Boltzmann equation for electrons or using the results of kinetic Monte-Carlo simulations.

The ion number density, \(n_i\), satisfies the equation

\[
\frac{\partial n_i}{\partial t} = \kappa_e n_e.
\]

(3)

The electric field components are defined as

\[
E_e^{(x)} = -\frac{\partial \varphi}{\partial x}, \quad E_e^{(y)} = E_0 - \frac{\partial \varphi}{\partial y},
\]

(4)
where $\varphi$ is the self-consistent electric potential and $E_0$ is the background electric field. The electric potential satisfies the Poisson equation
\[
\frac{\partial^2 \varphi}{\partial x^2} + \frac{\partial^2 \varphi}{\partial y^2} + \frac{1}{x} \frac{\partial \varphi}{\partial x} = -\frac{e}{\varepsilon_0} (n_i - n_e),
\]
where $e$ is the elementary charge and $\varepsilon_0$ is the vacuum permittivity.

To formulate the problem, we consider a bounded domain $\Omega = [0, R_\Omega]^2$, $R_\Omega > 0$. The edges of $\Omega$ are denoted as
\[
\Sigma_1 = \{(x, y) \mid 0 \leq x \leq R_\Omega, y = 0\}, \quad \Sigma_2 = \{(x, y) \mid x = R_\Omega, 0 \leq y \leq R_\Omega\},
\]
\[
\Sigma_3 = \{(x, y) \mid 0 \leq x \leq R_\Omega, y = R_\Omega\}, \quad \Sigma_4 = \{(x, y) \mid x = 0, 0 \leq y \leq R_\Omega\}.
\]

The electric potential is subject to the boundary conditions
\[
\frac{\partial \varphi}{\partial x} \bigg|_{\Sigma_2 \cup \Sigma_4} = 0, \quad \varphi \big|_{\Sigma_1 \cup \Sigma_3} = 0,
\]
and the electron continuity equation (1) is subject to the boundary conditions
\[
J_e(x) \bigg|_{\Sigma_4} = 0, \quad \frac{\partial n_e}{\partial x} \bigg|_{\Sigma_2} = 0, \quad \frac{\partial n_e}{\partial y} \bigg|_{\Sigma_1 \cup \Sigma_3} = 0.
\]

The distributions of $n_e$ and $n_i$ at $t = 0$ are assumed to be known. Typically, these distributions are defined as the sum of a constant background density (preionization level) and a localized density perturbation (see section 4). The density perturbation is chosen so that it leads to the formation of moving ionization fronts. The equations of the streamer model are solved numerically to simulate the propagation of such fronts at $t > 0$.

### 3. Numerical method

#### 3.1. Computational mesh

The computational domain, $\Omega$, is partitioned into a set of square subdomains. One step of the partitioning procedure is illustrated in Fig. 1 (a). Here, a certain subdomain $\Omega_0 \subseteq \Omega$ is split horizontally into equal subdomains $\Omega_1$, $\Omega_2$, which are split vertically into equal subdomains $\Omega_3$, $\Omega_4$ and $\Omega_5$, $\Omega_6$, respectively. All subdomains are arranged in a binary tree as shown in Fig. 1 (b). The subdomains are labelled by the indices $k$, $i$, where $k$ is the refinement level and $i \in \{0, 1\}$ is the process stage. Namely, $i = 1$ labels the rectangular domains obtained after the horizontal splitting, and $i = 0$ labels the square domains. The partitioning procedure is applied sequentially to the terminal (leaf) nodes of the tree starting from the root node $\Omega$. If required, some terminal nodes can be ignored in the partitioning process depending on a certain condition. That is, the binary tree that includes all subdomains is not necessarily complete. This binary tree and its terminal nodes are referred to as the first-level grid and the block elements, respectively.

The first-level grid is subject to the following constraint: the refinement levels for any two neighbouring block elements differ by no more than one. This condition is referred to as the refinement level constraint.

Block elements are further split into subdomains using the same procedure as that described above. The resulting binary tree and its terminal nodes are referred to as the second-level grid and the finite elements, respectively. The second-level grid is always perfect, i.e., finite elements, which reside in a single block element, are of the same dimensions. The number of finite elements is chosen to be the same for all block elements.

Combined together, the first- and second-level grids represent a hierarchical Cartesian grid that is referred to as the computational mesh. If required, adaptive mesh refinement is applied to the first-level grid. That is, block elements can be added or removed during the simulation based on a certain criterion. In Fig. 1 (c) we demonstrate an example of the computational mesh constructed for a certain square domain. In this example, the first-level grid is constructed by splitting the left-most node of the tree at each step of the partitioning procedure.

A two-level structure of the computational mesh was used for the following reasons. First, this structure is attractive for the implementation of the HPS scheme. Namely, solution operators constructed for the block elements can be
reused during the simulation. That is, the solution operator has to be updated only for the first-level \( \xi \) grid. This reduces the computational cost of the HPS scheme when the adaptive mesh refinement is used. Second, the numerical scheme for the electron continuity equation can be partially vectorized within the block elements. This improves the computational efficiency of the computer code, especially when using high-level programming languages (e.g., Python). The optimal combination of the first- and second-level grids was found experimentally during the implementation of the method.

![Figure 1](image.png)

**Fig. 1.** (a) One step of the partitioning procedure. (b) Subdomains arranged in a binary tree. (c) Example of a computational mesh. Red lines show the boundaries of block elements. Black lines show the boundaries of finite elements. Different colors indicate the block elements with different refinement levels.

### 3.2. Reference element and basis functions

To approximate the governing equations, each finite element is mapped onto a reference element \( \Omega_E = [-1, 1]^2 \). The edges of \( \Omega_E \) are denoted as \( \Gamma_{1,3} = \{(x, y) \mid -1 \leq x \leq 1, y = \pm 1\} \) and \( \Gamma_{2,4} = \{(x, y) \mid -1 \leq y \leq 1, x = \pm 1\} \). The reference element is depicted in Fig. 2(a).

Let \( \xi_k \in [-1, 1] \) be the nodes of the Gauss quadrature (Gauss nodes) for \( k \in \{1, \ldots, n\} \), with \( n \in \mathbb{N} \). When considering the reference element, points \((\xi_k, \xi_m)\) for \( k, m \in \{1, \ldots, n\} \) are referred to as the internal Gauss nodes and points \((\xi_k, \pm 1), (\pm 1, \xi_k)\) are called the boundary Gauss nodes. The Gauss nodes are used to construct polynomial basis functions. The one-dimensional basis functions are defined as follows:

\[
l_k(\xi) = \prod_{m=1, m \neq k}^{n} \frac{\xi - \xi_m}{\xi_k - \xi_m}, \quad \xi \in [-1, 1], \quad k \in \{1, \ldots, n\}.
\] (8)

The two-dimensional basis functions are given by \( l_k(x)l_m(y) \) for \( k, m \in \{1, \ldots, n\} \). Note that \( l_k(\xi_m) = \delta_{km} \), where \( \delta_{km} \) is the Kronecker delta (i.e., \( \delta_{km} = 1 \) for \( k = m \) and \( \delta_{km} = 0 \) otherwise). Linear combinations of the basis functions represent interpolation polynomials. Since \( l_k(\xi_m) = \delta_{km} \), the interpolation polynomial \( f(\xi) = \sum_{k=1}^{n} f_k l_k(\xi) \) satisfies the property \( f_k = f(\xi_k) \). Similarly, the interpolation polynomial \( f(x, y) = \sum_{k=1}^{n} \sum_{m=1}^{n} f_{km} l_k(x)l_m(y) \) satisfies the property \( f_{km} = f(\xi_k, \xi_m) \).

A convenient way to compute the values of \( l_k(\xi) \) and its derivatives is to represent \( l_k(\xi) \) as a linear combination of the Legendre polynomials \( P_0(\xi), \ldots, P_{n-1}(\xi) \). Let

\[
l_k(\xi) = \sum_{m=0}^{n-1} c_m^{(k)} P_m(\xi),
\] (9)
where $c_{m}^{(k)}$ are the Legendre expansion coefficients given by [25]

$$c_{m}^{(k)} = (m + 1/2) \int_{-1}^{1} l_{k}(\xi) P_{m}(\xi) d\xi.$$  \hspace{1cm} (10)

Using the Gauss quadrature rule to calculate the integral in equation (10), we obtain

$$c_{m}^{(k)} = (m + 1/2) \gamma_k P_{m}(\xi_k),$$  \hspace{1cm} (11)

where $\gamma_k$ is the $k$-th weight of the Gauss quadrature formula. Note that relation (11) is exact, since $l_{k}(\xi)P_{m}(\xi)$ is the polynomial of degree less than $2m - 1$. Using equations (9) and (11), we can compute $l_{k}(\xi)$ and its derivatives using the well-known recurrence relations for the Legendre polynomials [25].

Given a function $f$ defined on $[-1, 1]$, it is possible to construct the vector $\vec{f} = (f(\xi_1), ..., f(\xi_n))^T$. This mapping is denoted as $\mathcal{V}_{1D} : f \rightarrow \vec{f}$.

Similarly, given a function $f$ defined on $\Omega_E$, we can construct the vector $\vec{f}$ with components $f(\xi_k, \xi_m)$. For this, the Gauss nodes $(\xi_k, \xi_m)$ are numbered either in the row-major or column-major order as illustrated in Fig. 2(b). Then, the $i$-th component of $\vec{f}$ is defined as the value of $f$ at the $i$-th Gauss node. We denote this mapping as $\mathcal{V}_{2D}^{(r)} : f \rightarrow \vec{f}$ and $\mathcal{V}_{2D}^{(c)} : f \rightarrow \vec{f}$ for the row-major and column-major numbering of the Gauss nodes, respectively. Also we introduce the permutation matrix $P_\pi$ such that $\mathcal{V}_{2D}^{(c)} [f] = P_\pi \mathcal{V}_{2D}^{(r)} [f]$. It can be shown that this matrix is orthogonal and symmetric. Consequently, $P_\pi$ satisfies the property $P_\pi^{-1} = P_\pi$.

3.3. Adaptive mesh refinement

In order to account for the multiscale nature of the streamer discharge, adaptive mesh refinement (AMR) is used during the simulation. The criterion we use to guide the AMR procedure (referred to as the AMR criterion) is based on the value of the local ionization coefficient. This approach, which was introduced in [11, 12], is not universal but is suitable for testing the proposed numerical framework. The AMR criterion is specified in section 4 separately for each test problem. As it was mentioned in section 3.1, the AMR procedure is applied to block elements.

Here, we describe how to determine the state variables (e. g., $n_e, n_i$) for new block elements created in the AMR process. For simplicity, we assume that each block element consists of a single finite element. Nevertheless, the results of this section can be easily generalized to block elements that contain multiple finite elements. For convenience, we use the same notations as in Fig. 1.

Suppose that the refinement level needs to be increased in the region occupied by a certain block element, $\Omega_0$. In this case, new block elements $\Omega_3, ..., \Omega_6$ are added to the mesh, using the partitioning procedure shown in Fig. 1(a).
If, instead, the refinement level needs to be decreased in the region occupied by \( \Omega_3, \ldots, \Omega_6 \), these elements, together with the intermediate nodes \( \Omega_1, \Omega_2 \), are removed from the mesh and the parent node, \( \Omega_0 \), becomes the terminal one. The elements \( \Omega_0, \ldots, \Omega_6 \) and the respective internal Gauss nodes are shown in Fig. 3 for the case when \( \Omega_0 \) coincides with the reference element.

When the refinement level is increased, the state variables at the Gauss nodes in \( \Omega_3, \ldots, \Omega_6 \) are obtained using the corresponding interpolation polynomials defined on \( \Omega_0 \). When the refinement level is decreased, the state variables in \( \Omega_3, \ldots, \Omega_6 \) are used to reconstruct the state variables at the Gauss nodes in \( \Omega_0 \). For this, we use the projection procedure proposed by Kopriva [26]. In the one-dimensional case, this procedure is summarized as follows.

Let \( f_1(x), f_2(x), f(x) \) be continuous functions defined on \([-1, 0], [0, 1] \), and \([-1, 1] \), respectively. Suppose that

\[
\int_{-1}^{1} f(x) l_k(x) dx = \int_{-1}^{0} f_1(x) l_k(x) dx + \int_{0}^{1} f_2(x) l_k(x) dx, \quad k \in \{1, \ldots, n\}.
\]

The integrals in (12) are approximated using the Gauss quadrature and after simplification we obtain

\[
f(\xi_k) = \sum_{m=1}^{n} \left[ R_{km}^{(1)} f_1(\eta_m^{(1)}) + R_{km}^{(2)} f_2(\eta_m^{(2)}) \right], \quad k \in \{1, \ldots, n\},
\]

where \( \eta_m^{(1)} = (\xi_m - 1)/2, \eta_m^{(2)} = (\xi_m + 1)/2 \) and

\[
R_{km}^{(\alpha)} = \frac{\gamma_m}{2\gamma_k} l_k(\eta_m^{(\alpha)}), \quad \alpha \in \{1, 2\}.
\]

Thus, given the values of \( f_1, f_2 \) at the Gauss nodes on \([-1, 0] \) and \([0, 1] \), equation (13) can be used to determine the values of \( f \) at the Gauss nodes on \([-1, 1] \).

Next, let \( f_\alpha(x, y) \) be continuous functions defined on \( \Omega_\alpha \) for \( \alpha \in \{0, \ldots, 6\} \), respectively. Applying the projection procedure described above, we introduce the relations

\[
f_1(\eta_k^{(1)}, \xi_m) = \sum_{j=1}^{n} \left[ R_{mj}^{(1)} f_3(\eta_k^{(1)}, \eta_j^{(1)}) + R_{mj}^{(2)} f_4(\eta_k^{(1)}, \eta_j^{(2)}) \right],
\]

\[
f_2(\eta_k^{(2)}, \xi_m) = \sum_{j=1}^{n} \left[ R_{mj}^{(1)} f_5(\eta_k^{(2)}, \eta_j^{(1)}) + R_{mj}^{(2)} f_6(\eta_k^{(2)}, \eta_j^{(2)}) \right],
\]

and

\[
f_0(\xi_k, \xi_m) = \sum_{j=1}^{n} \left[ R_{kj}^{(1)} f_1(\eta_j^{(1)}, \xi_m) + R_{kj}^{(2)} f_2(\eta_j^{(2)}, \xi_m) \right],
\]

for \( k, m \in \{1, \ldots, n\} \). In summary, if \( f_3, \ldots, f_6 \) represent a state variable in \( \Omega_3, \ldots, \Omega_6 \), then equations (14), (15) are used to reconstruct the values of this variable at the Gauss nodes in \( \Omega_0 \).
3.4. Method for solving the Poisson equation

In this section, we introduce the method for solving the Poisson equation. First, the construction of the local solution operator and the DtN operator on the reference element is discussed.

3.4.1. The local solution operator and the DtN operator

Let us consider the model equation of the form

$$ L [\varphi(x, y)] = \rho(x, y), \quad (x, y) \in \Omega_E, $$

where

$$ L = \partial^2 / \partial x^2 + \partial^2 / \partial y^2 + a(x) \partial / \partial x. $$

It is assumed that \( \rho(x, y) \) and \( a(x) \) are suitably smooth functions.

The solution is represented in the following form:

$$ \varphi(x, y) = \varepsilon(x, y) + \omega(x, y), $$

where

$$ \varepsilon(x, y) = \int_{-1}^{1} \int_{-1}^{1} G(x, s) G(y, r) \sigma(s, r) ds dr, $$

and \( G \) is the Green’s function for the Poisson equation on \([-1, 1]\) subject to zero boundary conditions, namely,

$$ G(\xi, \eta) = \frac{1}{2} \begin{cases} (\xi - 1)(\eta + 1), & \eta < \xi \\ (\xi + 1)(\eta - 1), & \eta > \xi \end{cases}, \quad (\xi, \eta) \in [-1, 1]^2. $$

Note that, by construction, \( \varepsilon |_{\Gamma} = 0 \), where \( \Gamma = \Gamma_1 \cup \Gamma_2 \cup \Gamma_3 \cup \Gamma_4 \).

Let \( p_{1,2}(\xi) = (1 + \xi)/2 \) for \( \xi \in [-1, 1] \) and \( \varphi_1, \varphi_2, \varphi_3, \varphi_4 \) be the values of \( \varphi \) at the corner points \((-1, -1), (1, -1), (1, 1), (-1, 1)\), respectively. Then, the second term in (17) is defined as

$$ \omega(x, y) = \sum_{i=1}^{4} N_i(x, y) \varphi_i + p_1(y) \int_{-1}^{1} G(x, s) \sigma_1(s) ds + p_2(x) \int_{-1}^{1} G(y, s) \sigma_2(s) ds $$

$$ \quad \quad \quad \quad \quad \quad + p_2(y) \int_{-1}^{1} G(x, s) \sigma_3(s) ds + p_1(x) \int_{-1}^{1} G(y, s) \sigma_4(s) ds, $$

(20)

where \( N_1(x, y) = p_1(x)p_1(y), N_2(x, y) = p_2(x)p_1(y), N_3(x, y) = p_2(x)p_2(y), N_4(x, y) = p_1(x)p_2(y) \) are the conventional linear shape functions.

The function \( \omega \) on each edge is given by the sum of the linear contribution and the residual expressed in the integral form. The same is true for \( \varphi \), since \( \varepsilon |_{\Gamma} = 0 \). For example, \( \varphi \) on \( \Gamma_1 \) is given by

$$ \varphi(x, -1) = p_1(x) \varphi_1 + p_2(x) \varphi_2 + \int_{-1}^{1} G(x, s) \sigma_1(s) ds. $$

(21)

Similar expressions can be written for \( \varphi \) on \( \Gamma_2, \Gamma_3, \Gamma_4 \). Consequently, the functions \( \sigma_1, \sigma_2, \sigma_3, \sigma_4 \) are determined by the boundary conditions on \( \Gamma \). Note that expression (17) is consistent with the values of \( \varphi \) at the corners of \( \Omega_E \), i.e.,

$$ \varepsilon(-1, -1) + \omega(-1, -1) = \varphi_1, \quad \varepsilon(1, -1) + \omega(1, -1) = \varphi_2, \quad \varepsilon(1, 1) + \omega(1, 1) = \varphi_3, \quad \varepsilon(-1, 1) + \omega(-1, 1) = \varphi_4. $$

The approach we use to express the solution is the extension of the ideas presented in [21, 22]. For example, the representation given by (17), (18), (20) is similar to that used in [22] for the solution of fourth-order boundary value problems (see equation (8) in [22]). In our case, \( G(x, s) G(y, r) \) in (18) mimics the background Green’s function.
for the problem with zero boundary values and \( \omega \) is used to satisfy the boundary conditions. The idea to write the solution in the integral form using the background Green’s function was originally proposed in [21]. Following [21], expression (17) is used to rewrite equation (16) as the integral equation for the unknown function \( \sigma \).

Further, we discuss how to approximate \( \varepsilon, \omega \) and their derivatives. From (18) we get

\[
\frac{\partial^{\alpha} \varepsilon}{\partial x^{\alpha}} = \int_{-1}^{1} \int_{-1}^{1} \frac{\partial^{\alpha} G(x,s)}{\partial x^{\alpha}} G(y,r) \sigma(s,r) \, ds \, dr, \quad \frac{\partial^{\alpha} \varepsilon}{\partial y^{\alpha}} = \int_{-1}^{1} \int_{-1}^{1} \frac{\partial^{\alpha} G(y,r)}{\partial y^{\alpha}} \sigma(s,r) \, ds \, dr,
\]

for \( \alpha \in \{1, 2\} \). The first derivatives of \( \omega \) are given by

\[
\frac{\partial \omega}{\partial x} = \frac{1}{2} p_1(y) (\varphi_2 - \varphi_1) + \frac{1}{2} p_2(y) (\varphi_3 - \varphi_4) + \frac{1}{2} \int_{-1}^{1} G(y,s) \sigma_2(s) \, ds - \frac{1}{2} \int_{-1}^{1} G(y,s) \sigma_4(s) \, ds
\]
\[+ p_1(y) \int_{-1}^{1} \frac{\partial G(x,s)}{\partial x} \sigma_1(s) \, ds + p_2(y) \int_{-1}^{1} \frac{\partial G(x,s)}{\partial x} \sigma_3(s) \, ds,
\]

\[
\frac{\partial \omega}{\partial y} = \frac{1}{2} p_1(x) (\varphi_4 - \varphi_1) + \frac{1}{2} p_2(x) (\varphi_3 - \varphi_2) + \frac{1}{2} \int_{-1}^{1} G(x,s) \sigma_3(s) \, ds - \frac{1}{2} \int_{-1}^{1} G(x,s) \sigma_1(s) \, ds
\]
\[+ p_1(x) \int_{-1}^{1} \frac{\partial G(y,s)}{\partial y} \sigma_4(s) \, ds + p_2(x) \int_{-1}^{1} \frac{\partial G(y,s)}{\partial y} \sigma_2(s) \, ds.
\]

The second derivatives of \( \omega \) are given by

\[
\frac{\partial^2 \omega}{\partial x^2} = p_1(y) \sigma_1(x) + p_2(y) \sigma_3(x), \quad \frac{\partial^2 \omega}{\partial y^2} = p_1(x) \sigma_4(y) + p_2(x) \sigma_2(y).
\]

To proceed, we need to approximate the integrals of the form

\[
I_{\alpha}(\xi) = \int_{-1}^{1} \frac{\partial^{\alpha} G(\xi, \eta)}{\partial \xi^{\alpha}} f(\eta) \, d\eta, \quad \xi \in [-1, 1], \quad \alpha \in \{0, 1, 2\},
\]

where \( f \) is a suitably smooth function on \([-1, 1]\). Let \( I_1 = \mathcal{V}_{1D}[I_a] \) and \( f = \mathcal{V}_{1D}[f] \). According to the definition of \( G(\xi, \eta) \), we obtain \( I_2 = f \). Also, it is possible to construct the matrices \( G_{\alpha} \) such that \( I_{\alpha} \approx G_\alpha f \) for \( \alpha \in \{0, 1\} \) (see Appendix A).

Let

\[
\mathcal{G}_{\alpha} = \text{diag}(G_{\alpha}, \ldots, G_{\alpha}) \in \mathbb{R}^{n^2 \times n^2}, \quad \alpha \in \{0, 1\},
\]

and let \( \vec{\varepsilon}, \vec{\varepsilon}_x, \vec{\varepsilon}_y, \vec{\varepsilon}_{xx}, \vec{\varepsilon}_{yy} \) be the vectors obtained by applying \( \mathcal{V}_{2D}^{(r)} \) to \( \varepsilon, \partial \varepsilon/\partial x, \partial \varepsilon/\partial y, \partial^2 \varepsilon/\partial x^2, \partial^2 \varepsilon/\partial y^2 \), respectively. Then, approximating the integrals in (18), (22) separately in each direction, we obtain

\[
\vec{\varepsilon} \approx Q^{(\sigma)} \vec{\sigma}, \quad \vec{\varepsilon}_x \approx U^{(\sigma)} \vec{\sigma}, \quad \vec{\varepsilon}_y \approx U^{(\sigma)} \vec{\sigma}, \quad \vec{\varepsilon}_{xx} \approx L^{(\sigma)} \vec{\sigma}, \quad \vec{\varepsilon}_{yy} \approx L^{(\sigma)} \vec{\sigma},
\]

where \( \vec{\sigma} = \mathcal{V}_{2D}^{(r)}[\sigma] \) and

\[
Q^{(\sigma)} = P_\pi \mathcal{G}_0 P_\pi, \quad U^{(\sigma)} = P_\pi \mathcal{G}_0 P_\pi \mathcal{G}_1, \quad U^{(\sigma)} = P_\pi \mathcal{G}_1 P_\pi \mathcal{G}_0, \quad L^{(\sigma)} = P_\pi \mathcal{G}_0 P_\pi, \quad L^{(\sigma)} = \mathcal{G}_0.
\]

Further, let \( \vec{g}_\alpha = \mathcal{V}_{1D}[\varphi] \), \( \vec{\sigma}_\alpha = \mathcal{V}_{1D}[\sigma] \) for \( \alpha \in \{1, 2, 3, 4\} \), and \( \vec{p}_\alpha = \mathcal{V}_{1D}[p] \) for \( \alpha \in \{1, 2\} \). Approximating \( \varphi \) on the edges of \( \Omega_E \), we get

\[
\vec{\sigma}_\alpha \approx G_0^{-1} [\vec{g}_\alpha - \vec{p}_1 \varphi_k - \vec{p}_2 \varphi_m],
\]

for \( (\alpha, k, m) \in \{(1, 1, 2), (2, 2, 3), (3, 4, 3), (4, 1, 4)\} \).
Let us introduce the notations \( S = G_1^{-1}, H = G_1 G_0^{-1} \) and

\[
\begin{align*}
\vec{S}_\alpha &= S \vec{p}_\alpha, & \vec{t}_\alpha &= H \vec{p}_\alpha, & P^{(x)}_\alpha &= \text{diag}(\vec{p}_\alpha, \ldots, \vec{p}_\alpha) \in \mathbb{R}^{n^2 \times n}, & P^{(y)}_\alpha &= P_x P^{(x)}_\alpha, & \alpha \in \{1, 2\}.
\end{align*}
\]

Also let \( \vec{c} = (1, \ldots, 1)^T \in \mathbb{R}^n, \) \( E^{(x)} = \text{diag}(\vec{c}^T, \ldots, \vec{c}^T) \in \mathbb{R}^{n^2 \times n}, \) \( E^{(y)} = P_x E^{(x)} \) and let \( \vec{\omega}_-, \vec{\omega}_x, \vec{\omega}_y, \vec{\omega}_{xx}, \vec{\omega}_{yy} \) be the vectors obtained by applying \( V_{2D}^{(r)} \) to \( \omega, \partial \omega / \partial x, \partial \omega / \partial y, \partial^2 \omega / \partial x^2, \partial^2 \omega / \partial y^2 \), respectively. Using equations (20), (23), (24), (25), (28), we obtain the following expressions

\[
\begin{align}
\vec{\omega} &\approx Q^{(c)} \vec{g} - Q^{(c)} \vec{c}, \\
\vec{\omega}_x &\approx U^{(c)}_x \vec{g} - U^{(c)}_x \vec{c}, & \vec{\omega}_y &\approx U^{(c)}_y \vec{g} - U^{(c)}_y \vec{c}, \\
\vec{\omega}_{xx} &\approx L^{(c)}_{xx} \vec{g} - L^{(c)}_{xx} \vec{c}, & \vec{\omega}_{yy} &\approx L^{(c)}_{yy} \vec{g} - L^{(c)}_{yy} \vec{c},
\end{align}
\]

where \( \vec{c} = (\varphi_1, \varphi_2, \varphi_3, \varphi_4)^T, \) \( \vec{g} = (\vec{g}_1^T, \vec{g}_2^T, \vec{g}_3^T, \vec{g}_4^T)^T \in \mathbb{R}^{4n} \) and

\[
\begin{align*}
Q^{(c)} &= \begin{bmatrix} P_{1}^{(y)} \vec{p}_1 & P_{1}^{(y)} \vec{p}_2 & P_{2}^{(y)} \vec{p}_2 & P_{2}^{(y)} \vec{p}_1 \end{bmatrix}, & Q^{(e)} &= \begin{bmatrix} P_{1}^{(y)} & P_{2}^{(y)} & P_{2}^{(y)} & P_{1}^{(y)} \end{bmatrix}, \\
U^{(c)}_x &= \begin{bmatrix} P_{1}^{(y)} \vec{h}_1 & P_{1}^{(y)} \vec{h}_2 & P_{2}^{(y)} \vec{h}_2 & P_{2}^{(y)} \vec{h}_1 \end{bmatrix}, & U^{(c)}_y &= \begin{bmatrix} P_{1}^{(y)} H & E^{(x)}/2 & P_{2}^{(y)} H & -E^{(x)}/2 \end{bmatrix}, \\
U^{(c)}_y &= \begin{bmatrix} P_{1}^{(y)} \vec{h}_1 & P_{1}^{(y)} \vec{h}_2 & P_{2}^{(y)} \vec{h}_2 & P_{2}^{(y)} \vec{h}_1 \end{bmatrix}, & U^{(c)}_y &= \begin{bmatrix} -E^{(y)}/2 & P_{2}^{(y)} H & E^{(y)}/2 & P_{1}^{(y)} H \end{bmatrix}, \\
L^{(c)}_{xx} &= \begin{bmatrix} P_{1}^{(y)} \vec{s}_1 & P_{1}^{(y)} \vec{s}_2 & P_{2}^{(y)} \vec{s}_2 & P_{2}^{(y)} \vec{s}_1 \end{bmatrix}, & L^{(c)}_{yy} &= \begin{bmatrix} P_{1}^{(y)} S & 0_{n^2 \times n} & P_{2}^{(y)} S & 0_{n^2 \times n} \end{bmatrix}, \\
L^{(c)}_{yy} &= \begin{bmatrix} P_{1}^{(y)} \vec{s}_1 & P_{2}^{(y)} \vec{s}_1 & P_{2}^{(y)} \vec{s}_2 & P_{2}^{(y)} \vec{s}_2 \end{bmatrix}, & L^{(c)}_{xx} &= \begin{bmatrix} P_{1}^{(y)} S & 0_{n^2 \times n} & P_{2}^{(y)} S & 0_{n^2 \times n} \end{bmatrix}.
\end{align*}
\]

To simplify the implementation of the HPS scheme, the vector \( \vec{c} \) is excluded from (29), (30), (31). For this, \( \vec{c} \) is approximated using the values of \( \varphi \) at the boundary Gauss nodes. For example, we use

\[
\varphi_1 = \frac{1}{2} \left[ \vec{b}_1 \vec{g}_1 + \vec{b}_1 \vec{g}_4 \right], & \varphi_2 = \frac{1}{2} \left[ \vec{b}_2 \vec{g}_1 + \vec{b}_2 \vec{g}_2 \right], & \varphi_3 = \frac{1}{2} \left[ \vec{b}_3 \vec{g}_3 + \vec{b}_3 \vec{g}_2 \right], & \varphi_4 = \frac{1}{2} \left[ \vec{b}_4 \vec{g}_3 + \vec{b}_4 \vec{g}_4 \right],
\]

where \( \vec{b}_{1,2} = (l_1(\mp 1), \ldots, l_n(\mp 1))^T \). Similarly to [16], we use a simple averaging to approximate \( \varphi \) at the corner points. The accuracy of this approach was found to be sufficient for the purpose of our work.

Using (32), it is straightforward to rewrite (29), (30), (31) as

\[
\vec{\omega} \approx Q^{(b)} \vec{g}, & \vec{\omega}_x \approx U^{(b)}_x \vec{g}, & \vec{\omega}_y \approx U^{(b)}_y \vec{g}, & \vec{\omega}_{xx} \approx L^{(b)}_{xx} \vec{g}, & \vec{\omega}_{yy} \approx L^{(b)}_{yy} \vec{g},
\]

where the matrices \( Q^{(b)}, U^{(b)}_x, U^{(b)}_y, L^{(b)}_{xx}, L^{(b)}_{yy} \in \mathbb{R}^{n^2 \times 4n} \) are constructed during the elimination of \( \vec{c} \). For brevity, we do not present the full expressions for these matrices. Note that these expressions are of little practical use because the elimination of \( \vec{c} \) from (29), (30), (31) is automated in the software implementation of the method.

Substituting (17) into equation (16) and using (27), (33), we discretize equation (16) as follows:

\[
L^{(\sigma)} \vec{\sigma} = -L^{(b)} \vec{g} + \vec{\rho},
\]

where

\[
L^{(\sigma)} = L^{(\sigma)}_{xx} + L^{(\sigma)}_{yy} + AU^{(\sigma)}_x, & L^{(b)} = L^{(b)}_{xx} + L^{(b)}_{yy} + AU^{(b)}_x, & A = \text{diag}(\vec{a}),
\]

and \( \vec{a} = V_{2D}^{(r)} [a], \vec{\rho} = V_{2D}^{(r)} [\rho] \). From equation (34) we get

\[
\vec{\sigma} = -D^{(\sigma)} L^{(b)} \vec{g} + D^{(\sigma)} \vec{\rho},
\]
where $D^{(\sigma)}$ is the inverse matrix of $L^{(\sigma)}$. Finally, we substitute (35) into (27) and, using (33), obtain

$$\tilde{\varphi} = Q^{(g)}\bar{g} + Q^{(p)}\bar{p}, \quad \tilde{\varphi}_x = U^{(g)}_x\bar{g} + U^{(p)}_x\bar{p}, \quad \tilde{\varphi}_y = U^{(g)}_y\bar{g} + U^{(p)}_y\bar{p}, \quad (36)$$

where $Q^{(p)} = Q^{(g)}D^{(\sigma)}$, $U^{(g)}_x = U_\sigma^{(g)}D^{(\sigma)}$, $U^{(p)}_y = U_\sigma^{(p)}D^{(\sigma)}$,

$$Q^{(g)} = -Q^{(g)}D^{(\sigma)}L^{(b)} + Q^{(b)}, \quad U^{(g)}_x = -U^{(g)}_xD^{(\sigma)}L^{(b)} + U^{(b)}_x, \quad U^{(p)}_y = -U^{(p)}_yD^{(\sigma)}L^{(b)} + U^{(b)}_y,$$

and $\tilde{\varphi}$, $\tilde{\varphi}_x$, $\tilde{\varphi}_y$ are the approximations to $V_\sigma^{(r)}$, $V_\sigma^{(r)}[\partial \varphi/\partial x]$, $V_\sigma^{(r)}[\partial \varphi/\partial y]$, respectively.

Let

$$\tilde{n}_1 = V_{1D} \left[ \frac{\partial \varphi}{\partial y} \big|_{\Gamma_1} \right], \quad \tilde{n}_2 = V_{1D} \left[ \frac{\partial \varphi}{\partial x} \big|_{\Gamma_2} \right], \quad \tilde{n}_3 = V_{1D} \left[ \frac{\partial \varphi}{\partial y} \big|_{\Gamma_3} \right], \quad \tilde{n}_4 = V_{1D} \left[ \frac{\partial \varphi}{\partial x} \big|_{\Gamma_4} \right],$$

and let

$$B_\alpha = \text{diag}(\tilde{b}_\alpha^T, ..., \tilde{b}_\alpha^T) \in \mathbb{R}^{n \times n^2}, \quad \alpha \in \{1, 2\}.$$ 

Then using (36), we obtain

$$\tilde{n}_\alpha = T_\alpha \bar{g} + \bar{r}_\alpha, \quad \alpha \in \{1, 2, 3, 4\}, \quad (37)$$

where

$$T_1 = B_1P_\pi U^{(g)}_y, \quad T_2 = B_2U^{(g)}_x, \quad T_3 = B_2P_\pi U^{(g)}_y, \quad T_4 = B_1U^{(g)}_x,$$

$$\bar{r}_1 = B_1P_\pi U^{(p)}_y\bar{p}, \quad \bar{r}_2 = B_2U^{(p)}_x\bar{p}, \quad \bar{r}_3 = B_2P_\pi U^{(p)}_y\bar{p}, \quad \bar{r}_4 = B_1U^{(p)}_x\bar{p}.$$

In summary, equations (36), (37) represent the local solution operator and the local DtN operator, respectively. These operators can be used to find the approximate solution of equation (16) on the reference element. The convergence and accuracy of the proposed scheme is tested in Appendix B. The comparison of the proposed scheme with the spectral collocation method is discussed as well.

### 3.4.2. The hierarchical Poincaré - Steklov scheme

The second stage of the method for solving the Poisson equation is the HPS scheme. The detailed description of this scheme can be found in [16, 17, 19]. The HPS scheme includes the following steps.

First, the local solution operator and the local DtN operator are constructed for each finite element using the discretization scheme presented in section 3.4.1. The reference element is then mapped back to the original coordinate system and the obtained local operators are transformed accordingly.

Next, the DtN operator for each inner node of the computational mesh is constructed by merging the DtN operators of its children nodes. We describe the merging procedure for the model problem shown in Fig. 4. Let $\Omega_a$, $\Omega_b$ be the children nodes of the node $\Omega_c$ and let $\Gamma_\alpha^{(k)}$ denote the edges of these elements for $\alpha \in \{1, 2, 3, 4\}$ and $k \in \{a, b, c\}$. We suppose that each edge $\Gamma_\alpha^{(k)}$ is discretized by a finite set of points. The values of $\varphi$ and its normal derivative at these points are used to construct the vectors $\bar{g}_\alpha^{(k)}$ and $\tilde{n}_\alpha^{(k)}$, respectively.

Since $\Omega_c = \Omega_a \cup \Omega_b$, the following holds:

$$\bar{g}_2^{(c)} = \bar{g}_2^{(b)}, \quad \bar{g}_4^{(c)} = \bar{g}_4^{(a)}, \quad \tilde{n}_2^{(c)} = \tilde{n}_2^{(b)}, \quad \tilde{n}_4^{(c)} = \tilde{n}_4^{(a)},$$

$$\bar{g}_\alpha^{(c)} = \left( \bar{g}_\alpha^{(a)}^T, \bar{g}_\alpha^{(b)}^T \right)^T, \quad \tilde{n}_\alpha^{(c)} = \left( \tilde{n}_\alpha^{(a)}^T, \tilde{n}_\alpha^{(b)}^T \right)^T, \quad \alpha \in \{1, 3\}. \quad (38)$$

The DtN operators for $\Omega_a$, $\Omega_b$, $\Omega_c$ can be written as

$$\tilde{n}_\alpha^{(k)} = \sum_{\gamma=1}^{4} T^{(k)}_{\alpha\gamma} \bar{g}_\gamma^{(k)} + \bar{r}_\alpha^{(k)}, \quad \alpha \in \{1, 2, 3, 4\}, \quad k \in \{a, b, c\}. \quad (39)$$
Note that, in general case, the grid points on $\Gamma_2^{(a)}$ and $\Gamma_4^{(b)}$ do not coincide with each other. We denote by $P_{ab}$ the operator that maps the vector of values at the grid points on $\Gamma_2^{(a)}$ to the vector of values at the grid points on $\Gamma_4^{(b)}$. The operator of the inverse transform is denoted by $P_{ba}$.

Using the DtN operators for $\Omega_a, \Omega_b$ and the continuity condition $\bar{\eta}_4^{(b)} = P_{ab}\bar{\eta}_2^{(a)}$, we obtain

$$\bar{g}_4^{(b)} = \left[P_{ab}T_{22}^{(a)}P_{ba} - T_{44}^{(b)}\right]^{-1}\left[\sum_{\gamma=1,2,3}T_{4\gamma}^{(b)}\bar{g}_{\gamma}^{(b)} - \sum_{\gamma=1,3,4}P_{ab}T_{2\gamma}^{(a)}\bar{g}_{\gamma}^{(a)} + \bar{r}_4^{(b)} - P_{ab}\bar{r}_2^{(a)}\right], \quad (40)$$

where it is taken into account that $\bar{g}_{2}^{(a)} = P_{ba}\bar{g}_4^{(b)}$. Expression (40) and relation $\bar{g}_{2}^{(a)} = P_{ba}\bar{g}_4^{(b)}$ are used to exclude $\bar{g}_{2}^{(a)}, \bar{g}_4^{(b)}$ from expressions (39) for $\alpha \in \{1, 3, 4\}$ with $k = a$ and for $\alpha \in \{1, 2, 3\}$ with $k = b$, respectively.

In this way, it is possible to construct the set of mappings that transform the potential on the boundary of $\Omega_c$ to the vectors $\bar{\eta}_1^{(a)}, \bar{\eta}_1^{(b)}, \bar{\eta}_3^{(a)}, \bar{\eta}_3^{(b)}, \bar{\eta}_2^{(a)}, \bar{\eta}_2^{(b)}$ and the corresponding parts of the solution on the boundary of $\Omega_c$. This step finalizes the construction of the DtN operator for the parent node $\Omega_c$.

The merging procedure is applied until the DtN operator for the root node, $\Omega$, is obtained. Then, the boundary conditions are used to compute the solution on the boundary of $\Omega$.

Further, the solution on the boundary of each inner node of the mesh (starting from the root node) is used to compute the solution on the boundaries of its children nodes. For the model problem in Fig. 4, this step is performed as follows. Given the solution on the boundary of $\Omega_c$, expression (40) and relation $\bar{g}_{2}^{(a)} = P_{ba}\bar{g}_4^{(b)}$ are used to compute the solution on $\Gamma_2^{(a)}$ and $\Gamma_4^{(b)}$. The potential on the remaining edges of $\Omega_a$ and $\Omega_b$ is obtained directly by taking the corresponding parts of the solution on the boundary of $\Omega_c$. As a result, we obtain the potential on the boundaries of the children nodes, $\Omega_a$ and $\Omega_b$.

The last step of the HPS scheme is to compute the potential and its derivatives at the internal Gauss nodes of each finite element using the local solution operators.

The set of the DtN operators for a group of the inner mesh nodes can be referred to as the partial solution operator. The solution operator for the entire computational mesh is usually called the global solution operator. If required, the partial solution operators can be constructed and saved independently. For example, in the present work, the solution operators for each block element (see section 3.1 for the details) are precomputed and reused during the simulation.

The global solution operator must be newly constructed every time the mesh or charge distribution are changed. When only the charge distribution is changed, there is only need to update the translation vectors in the DtN operators (i.e., $\bar{r}_{1}^{(b)}$ in (39)). In this case, we say that the global solution operator is updated rather than constructed. Clearly, the computational time required to update the solution operator is noticeably lower than that required to construct it.

The convergence properties and accuracy of the developed HPS solver were tested using the method of manufactured solutions. The corresponding results are presented in Appendix C.

![Fig. 4. Schematic representation of the merging procedure involved in the HPS scheme.](image-url)
3.5. Method for solving the electron continuity equation

The streamer model includes the electron continuity equation. This equation can be solved using the existing methods for advection-diffusion problems. The method implemented in the present work is based on the DGSEM scheme. Note that the DGSEM can be formulated in a number of ways [27]. Here, we use the weak form of the DGSEM based on the application of the Gauss quadrature rule. In what follows, we describe the key components of this method for the model equation of the form

$$\frac{\partial u}{\partial t} + \frac{\partial f(x)}{\partial x} + \frac{\partial f(y)}{\partial y} = q,$$  \hspace{1cm} (41)

where $q$ is the source term and

$$f(x) = a(x)u - \nu(x)\frac{\partial u}{\partial x}, \quad f(y) = a(y)u - \nu(y)\frac{\partial u}{\partial y}.$$  

It is assumed that $q$, $a(x)$, $a(y)$, $\nu(x)$, $\nu(y)$ are suitably smooth functions of $x, y$. Also, we assume that for problems with axial symmetry $q$ includes the contribution to the flux divergence given by $-f(x)/x$.

As it was already mentioned, the DGSEM has a number of attractive features. However, there are two issues that require special attention when applying this method. First, spurious oscillations may affect the results when the solution varies rapidly. To address this issue, we use the same approach as that described in [23]. Namely, we apply the subcell finite volume (FV) scheme instead of the DGSEM scheme if the solution can become unstable (e.g., near the front of the streamer discharge). Note, however, that in some cases the DGSEM performs well without applying any stabilization procedure (see section 4). To detect the elements, in which the FV scheme is to be used, we need to introduce some criterion. This can be done in a number of ways, e.g., by analysing the spectral behaviour of the solution. Instead, we used a simple approach: the FV scheme was used for the elements, which are at the same prescribed refinement level or are in the prescribed part of the computational domain. Despite its simplicity, this approach is suitable for the test problems considered in the present work.

The second issue is the definition of the diffusion flux for the DGSEM scheme. This problem has been addressed in many works [28, 29, 30, 31, 32, 33], however, the efficiency and flexibility of the diffusion flux approximation are still a subject of discussion. Here, we address this problem using the projection procedure similar to that proposed in [20]. Briefly, given the interpolation polynomials in the adjacent elements, it is possible to project them onto the space of functions that are continuous up to the first derivative at the interface between the elements. The value of the first derivative on this interface is used to define the diffusion flux. The resulting formulation of the flux function is relatively simple and easy to implement.

The rest of this section is organized as follows. We first describe the projection procedure involved in the definition of the diffusion flux and introduce the numerical flux for advection-diffusion problems. Further, the DGSEM scheme is introduced. Finally, we introduce the subcell finite volume scheme and describe how it is combined with the DGSEM scheme.

3.5.1. The numerical flux function

Let $u_1(x), u_2(x)$ be polynomial functions defined on $[-2, 0]$ and $[0, 2]$, respectively. These functions are represented as $u_\alpha(x_\alpha) = \sum_{k=1}^{n} u_{\alpha k} l_k(x_{\alpha})$ for $\alpha \in \{1, 2\}$, where $x_{1,2} = x \pm 1 \in [-1, 1]$.

The continuity conditions at $x = 0$ are given by

$$u_1(0) = u_2(0), \quad u'_1(0) = u'_2(0).$$  \hspace{1cm} (42)

Let $\bar{u}_1 = \mathcal{V}_1D[u_1(x)]$ and $\bar{u}_2 = \mathcal{V}_1D[u_2(x)]$. Then, conditions (42) can be expressed as

$$\bar{b}_2^T \bar{u}_1 = \bar{b}_1^T \bar{u}_2, \quad \bar{d}_2^T \bar{u}_1 = \bar{d}_1^T \bar{u}_2,$$  \hspace{1cm} (43)
where \( \vec{b}_{1,2} = (l_1(\mp 1), ..., l_n(\mp 1))^T \) and \( \vec{d}_{1,2} = (l_1'(\mp 1), ..., l_n'(\mp 1))^T \). Let

\[
B = \begin{bmatrix}
\vec{b}_2^T & -\vec{b}_1^T \\
\vec{d}_2^T & -\vec{d}_1^T
\end{bmatrix} \in \mathbb{R}^{2 \times 2n}.
\]

Then, conditions (43) can be written as

\[
B \vec{u} = 0, \tag{44}
\]

where \( \vec{u} = (\vec{u}_1^T, \vec{u}_2^T)^T \). Equation (44) implies that if conditions (42) hold then \( \vec{u} \in \text{Null}(B) \). In general case, \( \vec{u} \) does not satisfy (44) a priori, but it can be projected onto the space spanned by the basis vectors for \( \text{Null}(B) \).

To construct the projection matrix, we consider the singular value decomposition \( B = U \Sigma V^T \), where \( \Sigma \) is the matrix of singular values and \( U, V \) are the matrices of left and right singular vectors, respectively. The matrix of singular values is of the form \( \Sigma = [\text{diag}(\tilde{\sigma}_1, \tilde{\sigma}_2) \mid 0_{2 \times 2n-2}] \), where \( \tilde{\sigma}_1, \tilde{\sigma}_2 \) are the non-zero singular values of \( B \). Let \( \nabla \in \mathbb{R}^{2n \times 2n-2} \) denote the matrix formed by the last \( 2n - 2 \) columns of \( V \). Then, the projection of \( \vec{u} \) onto the null space of \( B \) is given by

\[
\vec{v} = W \vec{u}, \tag{45}
\]

where \( W = \nabla \nabla^T \).

The vector \( \vec{v} \) can be expressed as \( \vec{v} = (\vec{v}_1^T, \vec{v}_2^T)^T \), with \( \vec{v}_1, \vec{v}_2 \in \mathbb{R}^n \). Further, we introduce \( v_\alpha(x_\alpha) = \sum_{k=1}^n v_{\alpha k} l_k(x_\alpha) \) for \( \alpha \in \{1, 2\} \), where \( v_{\alpha k} \) are the components of \( \vec{v}_\alpha \), respectively. Also, we introduce

\[
v(x) = \begin{cases}
v_1(x), & x \in [-2, 0] \\
v_2(x), & x \in [0, 2]
\end{cases} \tag{46}
\]

Since \( \vec{v} \in \text{Null}(B) \), \( v_1 \) and \( v_2 \) satisfy the continuity conditions (42). Thus, \( v(x) \) is continuous up to the first derivative at \( x = 0 \). As an example, Fig. 5 shows \( u_1(x) \), \( u_2(x) \) and \( v(x) \) for the case when \( \vec{u}_1, \vec{u}_2 \) are formed by taking the values of \( \sin(x) \) at the corresponding Gauss nodes with \( n = 4 \).

The first derivative of \( v(x) \) at \( x = 0 \) can be used to construct a numerical diffusion flux. Taking, for example, \( v'(0) = v_1'(0) \), we obtain

\[
v'(0) = \vec{d}_2^T \vec{v}_1. \tag{47}
\]

Further, we express the projection matrix \( W \) as

\[
W = \begin{bmatrix}
W_{11} & W_{12} \\
W_{21} & W_{22}
\end{bmatrix},
\]

Fig. 5. Interpolation polynomials \( u_1(x), u_2(x) \) and the piecewise function \( v(x) \) that is continuous up to the first derivative at \( x = 0 \). Right panel shows the neighbourhood of \( x = 0 \). The polynomials \( u_1(x), u_2(x) \) are constructed using the values of \( \sin(x) \).
where \( W_{i,j} \in \mathbb{R}^{n \times n} \) for \( i, j \in \{1, 2\} \). According to (45), we get \( \bar{v}_1 = W_{11} \bar{u}_1 + W_{12} \bar{u}_2 \). Substituting this expression into (47), we obtain

\[
\nu'(0) = \bar{w}_1^T \bar{u}_1 + \bar{w}_2^T \bar{u}_2, \tag{48}
\]

where \( \bar{w}_1^T = \bar{d}_2^T W_{11} \) and \( \bar{w}_2^T = \bar{d}_2^T W_{12} \). Alternatively, one can assume that \( \nu'(0) = \nu_2'(0) \). In this case we obtain \( \bar{w}_1^T = \bar{d}_1^T W_{21} \) and \( \bar{w}_2^T = \bar{d}_1^T W_{22} \). Therefore, there are two equivalent definitions of \( \bar{w}_1^T, \bar{w}_2^T \) in (48).

Let \( \bar{a}, \nu \) denote the advection velocity and diffusion coefficient at \( x = 0 \), respectively. Then, the advection-diffusion flux at \( x = 0 \) can be approximated as

\[
\mathbf{F}(\bar{u}_1, \bar{u}_2, \bar{a}, \nu) = \bar{a} \bar{u} - \nu \left[ \bar{w}_1^T \bar{u}_1 + \bar{w}_2^T \bar{u}_2 \right], \tag{49}
\]

where

\[
\bar{u} = \begin{cases} 
\bar{b}_1^T \bar{u}_1 & \bar{a} \geq 0 \\
\bar{b}_2^T \bar{u}_2 & \bar{a} < 0 
\end{cases}. 
\]

The first term in (49) is the conventional upwind flux for advection problems and the second term is the numerical diffusion flux. The consistency of the proposed approach was verified by performing a number of tests for a one-dimensional advection-diffusion equation. The corresponding results are presented in Appendix D.

### 3.5.2. The discontinuous Galerkin scheme

This section introduces the DGSEM scheme for equation (41). The details of the method are presented for the reference element, \( \Omega_E \). The solution is approximated by the interpolation polynomial, namely,

\[
\bar{u}(x, y) \approx \sum_{k=1}^{n} \sum_{m=1}^{n} u_{km} l_k(x) l_m(y).
\]

The other parameters, such as \( q,(x), \nu,(x) \), etc. are approximated in the same way.

According to the DGSEM scheme, equation (41) is multiplied by the basis functions \( l_k(x) l_m(y) \) for \( k, m \in \{1, \ldots, n\} \) and integrated over the element area. The Green’s identity is applied once to the left side of (41) and all integrals are approximated using the Gauss quadrature rule. Finally, we obtain

\[
\frac{\partial u_{km}}{\partial t} - \sum_{i=1}^{n} \gamma_i \left[ \gamma_k^{-1} D_{ki} f_{im}^{(x)} + \gamma_m^{-1} D_{mi} f_{ki}^{(y)} \right] + F_{km}^{(x)} + F_{km}^{(y)} = q_{km}, \tag{50}
\]

where

\[
D_{km} = \frac{\partial l_k(x)}{\partial x} \bigg|_{x=x_m}, \quad k, m \in \{1, \ldots, n\},
\]

is the differentiation matrix and \( F_{km}^{(x)}, F_{km}^{(y)} \) are the boundary contributions given by

\[
F_{km}^{(x)} = \left[ l_k(1) f^{(x)}(1, \xi_m) - l_k(-1) f^{(x)}(-1, \xi_m) \right] \gamma_k^{-1}, \\
F_{km}^{(y)} = \left[ l_m(1) f^{(y)}(\xi_k, 1) - l_m(-1) f^{(y)}(\xi_k, -1) \right] \gamma_m^{-1}. \tag{51}
\]

The flux components at the internal Gauss nodes are approximated as

\[
f_{km}^{(x)} \approx a_{km}^{(x)} u_{km} - \nu_{km}^{(x)} \sum_{i=1}^{n} D_{ik} u_{im}, \quad f_{km}^{(y)} \approx a_{km}^{(y)} u_{km} - \nu_{km}^{(y)} \sum_{i=1}^{n} D_{im} u_{ki}. \tag{52}
\]

The flux at the boundary Gauss nodes is computed either by using the numerical flux function or by approximating the boundary conditions.
The flux on the interelement boundaries is computed using the numerical flux function. We describe the details of this procedure for two model problems illustrated in Fig. 6. The first problem is introduced in Fig. 6(a). Here, we consider two adjacent finite elements, which are of equal dimensions and share a common vertical edge. The element on the left is mapped to \([-2, 0] \times [-1, 1]\) and the element on the right is mapped to \([0, 2] \times [-1, 1]\). Let \(u_1(x, y)\) and \(u_2(x, y)\) be the interpolation polynomials for \(u\) defined on the left and right elements, respectively. Then, the flux at the Gauss nodes \((0, \xi_m)\) for \(m \in \{1, \ldots, n\}\) on the interelement boundary is given by

\[
f^{(x)}(0, \xi_m) = F(\tilde{u}_1, \tilde{u}_2, a^{(e)}, \nu^{(e)}),
\]

where \(\tilde{u}_{1,2} = V_{1D}[u_{1,2}(x \pm 1, \xi_m)]\) and \(a^{(e)}, \nu^{(e)}\) are taken as the average of the left and right limits of \(a^{(x)}, \nu^{(x)}\) at \(x = 0\), respectively.

A more involved procedure is used to compute the flux on the boundary between the elements of different dimensions. An example of such an arrangement is shown in Fig. 6(b). Here, we consider the element \(\Omega_a\), which is in contact with the elements \(\Omega_b, \Omega_c\). The side length of \(\Omega_a\) is two times that of \(\Omega_b, \Omega_c\). For convenience, we assume that \(\Omega_a\) coincides with the reference element and introduce the following notations:

\[
\Gamma_{ca} = \{(x, y) \mid -1 \leq y \leq 0, x = -1\}, \quad \Gamma_{ba} = \{(x, y) \mid 0 \leq y \leq 1, x = -1\}, \quad \Gamma_a = \Gamma_{ca} \cup \Gamma_{ba}.
\]

To compute the flux on the interelement boundaries, we introduce the ghost elements, \(\Omega_{ba}, \Omega_{ca} \in \Omega_a\) shown in the right panel of Fig. 6(b). The values of \(u\) and other parameters at the internal Gauss nodes in \(\Omega_{ba}, \Omega_{ca}\) are computed using the respective interpolation polynomials defined on \(\Omega_a\). The flux at the Gauss nodes on the edge \(\Gamma_{ca}\), which is shared by \(\Omega_a\) and \(\Omega_{ca}\), is computed using the procedure described above for the elements of equal dimensions. The same procedure is used to compute the flux at the Gauss nodes on the edge \(\Gamma_{ba}\) shared by \(\Omega_b\) and \(\Omega_{ba}\). The final step is to compute the flux at the Gauss nodes on the edge \(\Gamma_a\). To do this, we use the projection procedure introduced in section 3.3. Namely, assuming that \(f_1, f_2\) represent the flux on the edges \(\Gamma_{ca}, \Gamma_{ba}\), respectively, equation (13) is used to compute the flux at the Gauss nodes on the edge \(\Gamma_a\).

The above examples represent the general types of interelement contacts encountered in this work. That is, the problem of computing the flux on the interelement boundary can always be treated analogously to one of those demonstrated in Fig. 6.

The flux on the edges, which belong to the boundary of the computational domain \(\Omega\), is computed by approximating the boundary conditions. For example, the boundary conditions (7) are implemented as follows. The diffusion flux is set to zero on the entire boundary of \(\Omega\) (i.e., on \(\Sigma_1 \cup \Sigma_2 \cup \Sigma_3 \cup \Sigma_4\)). The advection flux is set to zero on \(\Sigma_4\), which is the axis of symmetry. The advection flux on \(\Sigma_1 \cup \Sigma_2 \cup \Sigma_3\) is computed using the interpolation polynomials within the corresponding finite elements.

The computer code implementing the DGSEM scheme was tested using the method of manufactured solutions. The corresponding results are omitted for brevity but can be found as a part of the code repository (see data and code availability statement after section 5).
3.5.3. The subcell finite volume scheme

In addition to the DGSEM, we introduce the subcell finite volume scheme for equation (41). We first describe the general formulation of this scheme. Consider a regular grid \( \{(x_i, y_j)\}, i, j \in \mathbb{Z} \), where \( x_i = ih, y_j = jh, h > 0 \). Below, the subscripts \( i, j, i \pm 1/2, j \pm 1/2 \) refer to the grid points \( x_i, y_j, x_i \pm h/2, y_j \pm h/2 \), respectively. The subscripts are used to indicate the value of any function at the grid points, e.g., \( u_{i,j} = u(x_i, y_j) \) and so on. The FV scheme for equation (41) is formulated as

\[
\frac{\partial u_{i,j}}{\partial t} + h^{-1} \left[ f_{i+1/2,j}^{(x)} - f_{i-1/2,j}^{(x)} + f_{i,j+1/2}^{(y)} - f_{i,j-1/2}^{(y)} \right] = q_{i,j},
\]

(53)

The flux in (53) is calculated using the appropriate flux function. For example, \( f(x)(x_i + h/2, y_j) \) is given by

\[
f_{i+1/2,j}^{(x)} = \alpha_{i+1/2,j}^{(x)} u_{i+1/2,j} - h^{-1} \nu_{i+1/2,j}^{(x)} (u_{i+1,j} - u_{i,j}),
\]

(54)

where

\[
u_{i+1,j}^{(x)} = \left\{ \begin{array}{ll}
u_{i,j} + \mathcal{L} \left( \frac{\Delta_{i+1,j}}{\Delta_{i,j}} \right) \Delta_{i,j} & a_{i+1/2,j}^{(x)} \geq 0 \\
u_{i+1,j} - \mathcal{L} \left( \frac{\Delta_{i+1,j}}{\Delta_{i,j}} \right) \Delta_{i+2,j} & a_{i+1/2,j}^{(x)} < 0 
\end{array} \right.
\]

with \( \Delta_{i,j} = u_{i,j} - u_{i-1,j}, \Delta_{i+1,j} = u_{i+1,j} - u_{i,j}, \Delta_{i+2,j} = u_{i+2,j} - u_{i+1,j} \) and \( \mathcal{L} \) being the limiter function. In this work, the Koren limiter function is used [34]:

\[
\mathcal{L}(\xi) = \max \left( 0, \min(\xi, \min((1 + 2\xi)/6, 1)) \right).
\]

The advection velocity and diffusion coefficient in (54) are approximated as the average of the corresponding values at the points \( (x_i, y_j) \) and \( (x_{i+1}, y_j) \). The remaining flux values in equation (53) are computed analogously to \( f_{i+1/2,j}^{(x)} \).

For problems with axial symmetry, \( f_{i,j}^{(x)} \) needs to be computed as well. In this work, \( f_{i,j}^{(x)} \) is approximated as the average of the flux values \( f_{i+1/2,j}^{(x)} \) and \( f_{i-1/2,j}^{(x)} \).

The finite volume scheme introduced above is implemented in such a way that it can be combined with the DGSEM scheme. Namely, the finite elements are split into subcells as demonstrated in Fig. 7(a) for the reference element. The number of cells per direction is taken to be the same as the number of Gauss nodes. The cell-centered values of \( u \) within each element are approximated using the FV scheme (53).

The numerical flux near and on the interelement boundaries is computed using the FV cells located in the neighbouring elements. For clarity, let us consider the examples of interelement boundaries discussed in section 3.5.2. The first example, shown in Fig. 6(a), is the boundary between two finite elements of equal dimensions. The FV cells from these elements can be combined into a single regular grid. Hence, the computation of the flux near and on the interelement boundary is straightforward in this case.

The second example is that shown in Fig. 6(b). The FV cells for the considered elements are depicted in Fig. 7(b). To be specific, we assume that the interelement boundary coincides with the interval \( 0 \leq y \leq 1 \) at \( x = 0 \). The FV cells in the regions with \( x \leq 0 \) and \( x \geq 0 \) are referred to as the fine mesh and coarse mesh, respectively. The side length of the FV cells in the fine mesh region is denoted by \( h \). In order to compute the flux near and on the interelement boundary, we introduce the set of ghost points. The ghost points in \( \Omega_a \) are located at \((h/2, y_j)\) and \((3h/2, y_j)\), where \( y_j = h/2 + jh, j \in \{0, ..., 2n - 1\} \). The values of \( u \) and other parameters at these points are obtained using the bilinear interpolation from the nearest grid points in \( \Omega_a \). Further, the flux values \( f^{(x)}(x_i \pm h/2, y_j) \) at \( x_i = -h/2 \) are computed for the right-most cells in the fine mesh region.

The second stage is to compute the flux values \( f^{(x)}(x_i \pm h, y_j) \), at \( x_i = h, y_j = h + 2jh, j \in \{0, ..., n-1\} \) for the left-most cells in the coarse mesh region. Here, \( f^{(x)}(x_i - h, y_j) \) is computed as the average of the flux values at the nearest points of the fine mesh. To compute \( f^{(x)}(x_i + h, y_j) \), we introduce the ghost points at \((-h, y_j)\) in the fine mesh region. The values of \( u \) and other parameters at these points are approximated as the average of the corresponding values at the nearest four points of the fine mesh. Then, it is straightforward to compute \( f^{(x)}(x_i + h, y_j) \).
As it was mentioned in section 3.5.2, the other types of interelement boundaries can be considered analogously to the examples discussed above.

The flux at the FV grid points on the boundary of the computational domain is computed using the boundary conditions. In particular, the boundary conditions (7) are implemented analogously to the procedure described for the DGSEM in section 3.5.2. The only difference is the computation of the advection flux on $\Sigma_1 \cup \Sigma_2 \cup \Sigma_3$. For the FV scheme, the advection flux on this boundary is approximated using a linear interpolation from the nearest grid points.

The computer code implementing the FV scheme was tested using the method of manufactured solutions. The corresponding results are omitted for brevity but can be found as a part of the code repository (see data and code availability statement after section 5).

3.5.4 The mapping between the degrees of freedom

Finally, we address the mapping between the degrees of freedom used in the DGSEM and FV schemes. This mapping is first considered for the one-dimensional case.

Let $f(x)$ be a continuous function on $[-1, 1]$. We introduce a regular grid $c_i = -1 + h(i - 1/2)$ for $i \in \{1, \ldots, n\}$, with $h = 2/n$, and the intervals $C_i = [c_i - h/2, c_i + h/2]$. Suppose that $f(x)$ is approximated by the interpolation polynomial $\sum_{k=1}^{n} l_k(x)f(\xi_k)$. The mean value of the interpolation polynomial over $C_i$ is defined as

$$F_i = h^{-1} \sum_{k=1}^{n} \left[ \int_{C_i} l_k(x)dx \right] f(\xi_k). \tag{55}$$

The mean value of $f(x)$ over $C_i$ can also be approximated by the cell-centered function value and, consequently, $f(c_i) \approx F_i$. Using the Gauss quadrature rule to approximate the integrals in (55), we obtain

$$f(c_i) \approx \sum_{k=1}^{n} S_{ik} f(\xi_k), \tag{56}$$

where

$$S_{ik} = (1/2) \sum_{j=1}^{n} \gamma_j l_k \left( \eta_j^{(i)} \right), \quad i, k \in \{1, \ldots, n\},$$

with $\eta_j^{(i)}$ being the Gauss nodes on $C_i$.

Now let $f(x, y)$ be a continuous function on the reference element, $\Omega_E$. The mapping (56) can be applied sequentially along the $x$- and $y$-directions to construct the approximate transform between $f(\xi_i, \xi_j)$ and $f(c_i, c_j)$ for
\( i, j \in \{1, \ldots, n\} \). Namely, we obtain

\[
f(c_i, c_j) \approx \sum_{m=1}^{n} \sum_{k=1}^{n} S_{ik} S_{jm} f(\xi_k, \xi_m).
\] (57)

The inverse transform is given by

\[
f(\xi_i, \xi_j) \approx \sum_{m=1}^{n} \sum_{k=1}^{n} S_{ik}^{-1} S_{jm}^{-1} f(c_k, c_m).
\] (58)

In summary, equations (57), (58) represent the way to map the degrees of freedom used in the DGSEM scheme to those used in the FV scheme and vice versa. The degrees of freedom can be the values of the solution, the values of the respective terms in the DGSEM and FV formulations. Similarly, using equation (3), we obtain

\[
\frac{du}{dt} \approx \sum_{s=0}^{m} \frac{(I_{1-s} \mathcal{Q}) u_{0} + (s+1) I_{s} \mathcal{T}}{2s+1},
\]

with \( \mathcal{T} \) the transport coefficients or other parameters.

3.6. Time discretization and simulation steps

This section describes the algorithm used to discretize the governing equations in time. For convenience, we introduce some notations first. Let \( u \) be a function defined on the computational domain \( \Omega \) and let \( u_1^\alpha, \ldots, u_N^\alpha \) be the values of \( u \) at either the internal Gauss nodes (\( \alpha = G \)) or nodes of the FV mesh (\( \alpha = FV \)) for a set of finite elements \( \Theta_\beta \), where \( \beta \in \{DG, FV, \Omega\} \). Here, \( \Theta_{DG} \) and \( \Theta_{FV} \) comprise the elements, where the electron continuity equation is discretized using the DGSEM and FV schemes, respectively. The set \( \Theta_\Omega \) includes all finite elements, i.e., \( \Theta_\Omega = \Theta_{DG} \cup \Theta_{FV} \). In addition, we introduce the notation

\[
\tilde{u}^{[\alpha, \beta]} = (u_1^\alpha, \ldots, u_N^\alpha)^T, \quad \alpha \in \{G, FV\}, \quad \beta \in \{DG, FV, \Omega\}.
\]

and mappings \( S^G : \tilde{u}^{[FV, \beta]} \rightarrow \tilde{u}^{[G, \beta]} \), \( S^{FV} : \tilde{u}^{[G, \beta]} \rightarrow \tilde{u}^{[FV, \beta]} \), which are defined using the procedure described in section 3.5.4.

Applying the method introduced in section 3.5 to discretize equation (1), we obtain the system of ordinary differential equations

\[
\frac{d}{dt} (\tilde{n}_e^{[G, DG]}) = \mathcal{T}_e^{DG} (\tilde{n}_e^{[G, \Omega]}),
\]

\[
\frac{d}{dt} (\tilde{n}_e^{[FV, DG]}) = \mathcal{T}_e^{FV} (\tilde{n}_e^{[FV, \Omega]}),
\]

where \( q_e = \kappa_e n_e \) and the mappings \( \mathcal{T}_e^{DG}, \mathcal{T}_e^{FV} \) use the particular components of \( \tilde{n}_e^{[G, \Omega]} \) and \( \tilde{n}_e^{[FV, \Omega]} \) to determine the respective terms in the DGSEM and FV formulations. Similarly, using equation (3), we obtain

\[
\frac{d}{dt} (\tilde{q}_i^{[G, DG]}) = \mathcal{Q}_e^{[G, DG]} (\tilde{q}_i^{[G, \Omega]}),
\]

\[
\frac{d}{dt} (\tilde{q}_i^{[FV, DG]}) = \mathcal{Q}_e^{[FV, DG]} (\tilde{q}_i^{[FV, \Omega]}).
\]

Equations (59), (60) are solved using a second-order total variation diminishing Runge-Kutta (RK) scheme [35]. For the equation of the form \( du/dt = \mathcal{T}(u) \), one step of this scheme is written as

\[
u_1 = u_0 + \Delta t \mathcal{T}(u_0),
\]

\[
u_2 = \left[ u_0 + u_1 + \Delta t \mathcal{T}(u_1) \right]/2,
\]

where \( \Delta t \) is the time step and \( u_0 \approx u(t_0), u_2 \approx u(t_0 + \Delta t) \), with \( t_0 \) being the time moment.

When the RK scheme is used to solve equations (59), (60), the electric field, transport coefficients and ionization frequency need to be known at each stage of the scheme. Hence, the RK scheme involves solving the Poisson equation. Suppose that \( \tilde{u}_{s[n, \Omega]}^{[\alpha]} \), where \( u \in \{n_e, n_i\} \) and \( \alpha \in \{G, FV\} \), are known for either \( s = 0 \) or \( s = 1 \). One stage of the RK scheme is then performed as follows:
1. Update the global solution operator for the HPS scheme. Use the HPS scheme to compute $\vec{\varphi}[G, \Omega]_s$ and $\vec{\upsilon}[G, \Omega]_s$, where $\upsilon \in \{ E(x), E(y) \}$. Compute $\vec{\upsilon}[FV, \Omega]_s = S_{FV} [\vec{\upsilon}[G, \Omega]_s]$ and $\vec{\pi}[\alpha, \Omega]_1$, where $\pi \in \{ \kappa_e, \mu_e, D_e(x), D_e(y) \}$, $\alpha \in \{ FV, G \}$.

2. Use the corresponding part of the RK scheme to compute $\vec{u}[G, DG]_{s+1}$ and $\vec{u}[FV, FV]_{s+1}$.

3. Compute $\vec{u}[FV, DG]_{s+1} = S_{FV} [\vec{u}[G, DG]_{s+1}]$ and $\vec{u}[G, FV]_{s+1} = S_{G} [\vec{u}[FV, FV]_{s+1}]$.

According to (61), the above procedure is performed twice, for $s = 0$ and $s = 1$, to compute the densities of plasma components at the next time level. The RK scheme can be simplified, if the Poisson equation is solved only once at $s = 0$ and the assumption is made that $\vec{\pi}[\alpha, \Omega]_1 = \vec{\pi}[\alpha, \Omega]_0$. This variant of the time integration method is referred to as the simplified RK scheme.

As it was mentioned in section 3.3, AMR is used to reduce the number of unknowns. Block elements are added or removed using the procedure described in section 3.3. The densities of plasma components are first determined at the internal Gauss nodes of new elements. The corresponding values at the nodes of the FV mesh are then obtained using the mapping described in section 3.5.4. Typically, it is sufficient to invoke the AMR procedure after several ($\sim 5$) RK steps.

The RK steps and AMR procedure are combined into one simulation step, which is summarized as follows:

1. Perform $m$ steps of the RK scheme (using ether full or simplified version of the scheme).
2. Use the AMR procedure to add or remove block elements, if required. If the mesh has been changed, then:
   (i) Add new block elements iteratively to satisfy the refinement level constraint (see section 3.1).
   (ii) Construct a new global solution operator for the HPS scheme.

The entire simulation procedure includes the initialization step and a sequence of simulation steps. In the initialization step, we construct the computational mesh, define $\vec{n}[\alpha, \Omega]_0, \vec{n}[\alpha, \Omega]_1$ at $t = 0$ for $\alpha \in \{ G, FV \}$ and construct the global solution operator for the HPS scheme. Next, simulation steps are performed to reach a particular time moment.

3.7. Implementation details

The simulation scheme described in sections 3.1 - 3.6 was implemented in the Python programming language (namely, Python 2.7.16 was used). Clearly, this choice is not optimal from the computational point of view. But the proposed algorithm is quite involved and it seemed reasonable to use a high-level programming enviroment in this proof-of-concept study.

A collection of Python scripts was written to perform the numerical experiments presented in this paper. Also, a brief code description was created to provide the information required to reproduce the computation results. The source code and the corresponding supplementary files are freely available (see data and code availability statement provided after section 5).

4. Numerical examples

The method described in section 3 was applied to the test problems presented in [8, 15]. In this section, we report the results of the corresponding numerical experiments and discuss the computational efficiency of the method. The experiments were performed on a laptop with Intel(R) Core(TM) i5-7200U processor (the clock rate is 2.5 GHz).
4.1. Constant transport coefficients

First, we consider the test problem presented in [15] (originally in [36]). In this example, a simplified streamer model is used. The assumption is made that the electron mobility and diffusion coefficients are constant, namely: \( D_{n}(x) = 2190 \, \text{cm}^2 \text{s}^{-1} \), \( D_{y}(y) = 1800 \, \text{cm}^2 \text{s}^{-1} \), \( \mu_e = (2.9 \times 10^5 / p_g) \, \text{cm}^2 \text{V}^{-1} \text{s}^{-1} \), where \( p_g = 760 \) Torr is the gas pressure. The ionization frequency is given by \( \kappa_e = \alpha_e \mu_e |\vec{E}| \), where \( \alpha_e = 5.7 \, p_g \exp \left( -260 p_g / |\vec{E}| \right) \, \text{cm}^{-1} \) is the Townsend ionization coefficient. The size of the computational domain is \( R_\Omega = 10 \) mm and the background electric field is \( E_0 = -52 \, \text{kV cm}^{-1} \).

The initial conditions for \( n_e, n_i \) are given by

\[
n_{\alpha}(x, y) = n_0 + n_1 \exp \left[ - \left( \frac{x}{\delta_x} \right)^2 - \left( \frac{y - y_0}{\delta_y} \right)^2 \right], \quad \alpha \in \{e, i\},
\]

where \( n_0 = 10^8 \, \text{cm}^{-3} \) is the preionization level, \( n_1 = 10^{14} \, \text{cm}^{-3} \) is the amplitude of the density perturbation and \( y_0 = 5 \) mm, \( \delta_y = 0.27 \) mm, \( \delta_x = 0.21 \) mm. The initial conditions are chosen so that two streamers propagate in opposite directions along the axis of symmetry.

The considered test problem is relatively simple and remarkable for two reasons. First, it was possible to use the DGSEM scheme without applying the subcell stabilization procedure. Also, the time required to perform a single simulation was reasonably low (\( \sim 5 - 10 \) min). Thus, we used this test case to validate the particular parts of the code (e.g., the DGSEM and FV schemes, singly and in combination), to assess different mesh configurations and to estimate the accuracy of the simplified RK scheme.

The parameters of the simulations are summarized in Table 1. AMR was used in all simulations except for \( S_0 \). The first-level grid at the initial step of the simulations \( S_1 \) - \( S_4 \) is shown in Fig. 8(a). The first-level grid used in the simulation \( S_0 \) is shown in Fig. 8(b). In all simulations the second level-grid was constructed as follows. Each block element was split into 16 finite elements (4 elements per direction). The number of Gauss nodes per direction for each finite element was \( n = 6 \). These parameters were found experimentally in an attempt to minimize the required computational time. The minimum length of the finite element edge was approximately 39 \( \mu \)m in all cases.

The simulations were run until \( t = 2.5 \) ns. The time step was \( \Delta t = 2 \) ps, which is close to that used in [15]. The time step is mainly limited by the ionization time scale (\( \kappa_e^{-1} \)) and is sufficiently small to satisfy the stability conditions for the DGSEM and FV schemes. For comparison, simulations with \( \Delta t = 1 \) ps were performed as well. The corresponding results can be found in the supplementary material.

The AMR procedure was used every 10 ps. To implement AMR, we introduced the dimensionless Townsend coefficient for each block element: \( \tilde{\alpha}_e = h \alpha_e^{\max} \), where \( \alpha_e^{\max} \) is the maximum over all \( \alpha_e \) at the FV mesh points and \( h \) is the half-length of the block element edge. The AMR criterion we used is as follows: (i) block elements with \( \tilde{\alpha}_e > 1 \) are refined, (ii) terminal block elements are removed if the maximum over all \( \tilde{\alpha}_e \) for these nodes is below 0.2. Additionally, the following conditions were used: (i) the maximum refinement level was fixed at \( k = 6 \), (ii) the de-refinement procedure was only applied to the elements with \( k > 3 \), (iii) only the elements adjacent to the \( y \)-axis were refined. Certainly, the present AMR procedure is not universal. It was specially designed to perform the proof-of-concept tests only. The simulations \( S_0, S_1, S_2 \) were performed to validate the DGSEM scheme for the electron

| Simulation label | \( S_0 \) | \( S_1 \) | \( S_2 \) | \( S_3 \) | \( S_4 \) |
|------------------|--------|--------|--------|--------|--------|
| Discretization scheme | DGSEM | DGSEM | DGSEM | FV | DGSEM-FV |
| Time integration method | RK | RK | \( \overline{\text{RK}} \) | RK | RK |
| Adaptive mesh refinement | | | | | |

\( \overline{\text{RK}} \) denotes the simplified RK scheme.

Table 1. Details of the simulation runs.
Fig. 8. (a) The computational mesh at the beginning of the simulations $S_1 - S_4$. (b) The computational mesh used in the simulation $S_0$.

Fig. 9. Evolution of the electron density (left) and the electric field (right) in the axial plane. The results were obtained in the simulation $S_1$. 
continuity equation. The simulations $S_3$, $S_4$ were performed to validate the subcell FV scheme and its combination with the DGSEM scheme. In the simulation $S_4$, the FV scheme was used for the block elements with the highest possible refinement level (i.e., $k = 6$) only.

The results of the simulations are discussed below. For clarity, we present the solution in the entire axial plane (the plane $(x_\perp, y)$ that contains the axis of symmetry). The evolution of $n_e$ and $|\vec{E}|$ in the axial plane is illustrated in Fig. 9 using the results of the simulation $S_1$. The computational mesh (first-level grid) corresponding to the results in Fig. 9 at $t = 2.5$ ns is demonstrated in Fig. 10.

In Fig. 11(a) the axial electric field obtained in the simulation $S_1$ is compared to the results of [15]. As it can

Fig. 10. (a) The computational mesh at the end of the simulation $S_1$. (b) Block elements and the electric field distribution near the streamer fronts at the end of the simulation $S_1$.

Fig. 11. Evolution of the axial electric field (left) and the charge density (right) on the axis of symmetry. The results were obtained in the simulation $S_1$. Circles show the reference data taken from [15].
be seen, our results are in good agreement with the reference data. The same is true for the results obtained in other simulation runs. Note that the simulations of [15] were performed using the second-order FV scheme both for the Poisson and electron continuity equations. Fig. 11(b) shows the charge density corresponding to the results shown in Fig. 11(a). It can be seen that the streamer fronts are thin layers of either positive or negative charge. The charge density shown in Fig. 11(b) is close to that presented in [15].

Generally, the results of the simulations $S_0 - S_4$ are in reasonable agreement with each other, as illustrated in Fig. 12. The codes based on different schemes for the electron continuity equation (i.e., DGSEM, FV and combined DGSEM-FV schemes) were all capable of simulating the streamer in a stable manner. The results of the simulations $S_0$ and $S_1$ are in close agreement (the relative difference between the corresponding solutions is generally below 2%). This validates the use of the AMR procedure. The use of the simplified RK scheme has a noticeable effect on the simulation results. Nevertheless, the accuracy of this method may be sufficient in some cases (depending on the chosen time step and particular physical conditions).

In Table 2 we summarize the total computational time and the maximum number of degrees of freedom (DOF) for the simulations $S_0 - S_4$. The number of DOF is given by $N_{\text{DOF}} = N_{\text{FE}} N_{\text{G}}$, where $N_{\text{FE}}$ is the total number of finite elements and $N_{\text{G}}$ is the total number of Gauss nodes per finite element ($N_{\text{G}} = 36$ for the considered test problem). The computational time given in Table 2 is averaged over a number of simulation runs.

The results in Table 2 show the benefit of using AMR. Note that the mesh used in the simulation $S_0$ was initially adjusted to the expected width of the streamer. In this case, the effect of using AMR on the computational time is moderate but still significant. Clearly, the efficiency of the AMR procedure is affected by the operational costs and by the need to construct the new solution operator for the HPS scheme. Nevertheless, reducing the number of DOF by means of AMR can play an important role, since streamer models are often combined with complex chemical models.

The effect of using the simplified RK scheme on the computational time is moderate. The HPS scheme requires

![Fig. 12. The axial electric field (upper panels) and electron density (lower panels) on the axis of symmetry at the end of the simulations $S_0 - S_4$. Circles show the reference data taken from [15].](image_url)
| Simulation label | $S_0$ | $S_1$ | $S_2$ | $S_3$ | $S_4$ |
|------------------|------|------|------|------|------|
| Computational time, [s] | 735 | 252 | 184 | 250 | 252 |
| Number of DOF, [$10^4$] | 10.944 | 4.3776 | 4.3776 | 4.3776 | 4.3776 |

Table 2. The total computational time and the maximum number of DOF for the simulations $S_0 - S_4$.

approximately 70% of the time spent per one stage of the RK scheme. Hence, the simplified RK scheme is expected to be $\sim 1.5$ times faster than the full one. However, the resulting speed-up is lower due to the use of the AMR procedure. Nevertheless, the simplified RK scheme is convenient for performing preliminary experiments.

It is interesting to note that the code based on the FV scheme is as fast as the code based on the DGSEM scheme. This seems to be the consequence of using the Python programming language. In fact, the functions implementing the key components of the FV and DGSEM schemes are relatively fast (both of them are vectorized). Hence, the code execution time is likely to be dominated by the remaining operational costs, which are approximately the same for both methods. This shows that the software implementation of the method can be noticeably improved to further reduce the computational time.

4.2. Field-dependent transport coefficients

Further, we consider the test problem introduced in [8] (this paper presents the comparison of different simulation codes for streamer discharges). This example is based on a more realistic streamer model compared to that used in the previous section. Here, the transport coefficients depend on the magnitude of the electric field. The corresponding expressions are

$$\mu_e = 2.3987 \bar{E}^{-0.26} \text{m}^2 \text{V}^{-1} \text{s}^{-1}, \quad D_e^{(x)} = D_e^{(y)} = 4.3628 \times 10^{-3} \bar{E}^{0.22} \text{m}^2 \text{s}^{-1},$$

where $\bar{E} = |\vec{E}|$ in V m$^{-1}$. The ionization frequency is defined as $\kappa_i = (\alpha_e - \eta_e) \mu_e |\vec{E}|$, where

$$\alpha_e = (1.1944 \times 10^6 + 4.3666 \times 10^{26} \bar{E}^{-3}) \exp \left( -2.73 \times 10^7 / \bar{E} \right) \text{m}^{-1}$$

and $\eta_e = 340.75 \text{m}^{-1}$ is the electron attachment coefficient. The initial conditions are given by

$$n_i = n_0 + n_1 \exp \left[ - \left( \frac{x}{\delta} \right)^2 - \left( \frac{y - y_0}{\delta} \right)^2 \right], \quad n_e = n_0,$$

where the preionization level is $n_0 = 10^7 \text{cm}^{-3}$, the amplitude of the density perturbation is $n_1 = 5 \times 10^{12} \text{cm}^{-3}$ and $y_0 = 10 \text{mm}$, $\delta = 0.4 \text{mm}$. The background electric field is $E_0 = -15 \text{kV cm}^{-1}$ and the computational domain size is $R_\Omega = 12.5 \text{mm}$.

The considered test problem is more challenging from the computational point of view than that discussed in section 4.1. The preionization level as well as the background electric field are relatively low. This increases the characteristic time scale of the streamer development. Moreover, this reduces the thickness of the streamer front and, consequently, increases the number of DOF required to describe the streamer. The use of the field-dependent transport coefficients makes it important to use a high-order method for the electron continuity equation, especially in the regions of low refinement level.

A number of preliminary experiments were performed to find the optimal simulation parameters. Below we report the results for a selected set of parameters, which seems to be the most appropriate from the computational perspective. In these simulations, each block element was split into 64 finite elements (8 elements per direction). The number of Gauss nodes per direction for each finite element was $n = 6$. The minimum length of the finite element edge was approximately $12 \mu\text{m}$. The time step was $\Delta t = 2 \text{ps}$ (this value satisfies the stability conditions for the DGSEM and FV schemes). For convenience, we used the simplified RK scheme. The simulations were run until $t = 16 \text{ns}$.  

25
The AMR procedure was customized for the considered test problem. Namely, the AMR criterion was as follows: (i) block elements with \( \bar{\alpha}_e > 0.5 \) are refined, (ii) terminal block elements are removed if the maximum over all \( \bar{\alpha}_e \) for them is below 0.1. In addition, the following conditions were used: (i) the maximum refinement level was fixed at \( k = 7 \), (ii) the minimum refinement level in the streamer channel (the region with \( x < 2 \) mm) was fixed at \( k = 4 \), when applying the de-refinement procedure. The AMR procedure was used every 10 ps.

The DGSEM scheme for the electron continuity equation was stabilized by means of the subcell FV scheme. In particular, the FV scheme was used for the block elements with the highest refinement level \( (k = 7) \) and in the region with \( y > 9.5 \) mm. The latter condition was used for simplicity because the region above the initial density perturbation is not of particular interest.

The simulation results are summarized in Figs. 13 - 16. The computational mesh (first-level grid) and the electric field distribution at the beginning of the simulation are shown in Fig. 13. The evolution of \( n_e \) and \( |\vec{E}| \) in the axial plane is illustrated in Fig. 14. The computational mesh (first-level grid) at \( t = 10 \) ns is demonstrated in Fig. 10. The evolution of the axial electric field and electron density on the axis of symmetry is shown in Fig. 16.

To validate our results, the obtained streamer characteristics were compared with the reference values. In particular, following the notations of [8], we used the results obtained by the groups CWI and DE (these results are available at [https://doi.org/10.17026/dans-x7r-266f](https://doi.org/10.17026/dans-x7r-266f)). Precisely, the results for the first test case presented in Figs. 5, 6 of [8] were used. The characteristics compared were: the streamer length \( (L_s) \), the maximum electric field magnitude \( (|\vec{E}|_{\text{max}}) \), the total number of electrons \( (N_e) \), the total electric charge \( (Q_{\text{tot}}) \). The latter two parameters are defined as

\[
N_e = 2\pi \int_{\Omega} n_e x dx dy, \quad Q_{\text{tot}} = 2\pi \int_{\Omega} e (n_i - n_e) x dx dy. \tag{62}
\]

The integrals in (62) were expressed as the sum of integrals over the finite elements and the corresponding integrals for each finite element were approximated using the Gauss quadrature rule.

As it is shown in Fig. 17, our results are in good agreement with the reference values. In particular, our results are close to the those of the DE group, which were obtained using the conventional finite element (FE) method. As it was discussed in [8], the total charge should be nearly constant in time (the total current on the boundaries of \( \Omega \) is...
Fig. 14. Evolution of the electron density (left) and electric field (right) in the axial plane.

Fig. 15. (a) The computational mesh after 10 ns. (b) Block elements and the electric field distribution near the streamer front after 10 ns.
Fig. 16. Evolution of the axial electric field (left) and the electron density (right) on the axis of symmetry.

Fig. 17. Comparison of the basic streamer characteristics obtained in the present work with the reference values taken from the simulations of the groups DE and CWI.

negligible). Clearly, our method is not strictly conservative in this sense. Nevertheless, the observed error in the total charge conservation is within acceptable range (below 1%).

Finally, Table 3 summarizes the information on the simulations presented in [8] and performed in this work. For clarity, the notations of [8] are used to specify the codes of different research groups. The information in Table 3 corresponds to that given in Tables 1 and 2 of [8].

It can be seen that the method proposed in the present work performs reasonably well. The execution time of our code is lower than that of the existing ones, except for the most effective simulation framework developed by


| Simulation code               | CWI | FR  | ES  | TUE | CN  | DE  | Present work |
|------------------------------|-----|-----|-----|-----|-----|-----|--------------|
| Poisson equation<sup>a</sup> | FV (MG) | FV (FFT) | FV | FV | FE | FE | HPS |
| Electron continuity equation<sup>b</sup> | FV | FV | FV | FV | FE | FE | DGSEM-FV |
| Time integration method<sup>c</sup> | exp. | exp. | exp. | imp. | imp. | imp. | exp. |
| Time step<sup>d</sup> | dyn. | dyn. | 1 ps | dyn. | dyn. | dyn. | 2 ps |
| Adaptive mesh refinement    | ✓   | ✓   | ✓   | ✓   | ✓   | ✓   | ✓   |
| Number of CPU cores         | 4   | 1   | 1   | 1   | 4   | 6   | 1 |
| CPUs clock rate, [GHz]      | 3.6 | 3.4 | 4.0 | 3.3 | 3.1 | 2.9 | 2.5 |
| Number of DOF, [10^5]       | 1.2 | 11  | 20  | 42  | 6.5 | 5.1 | 3.3 |
| Computational time           | 5 min | 6 h | 20 h | 25 h | 18 h | 15 h | 1.8 h |

<sup>a</sup> This field denotes the method for solving the Poisson equation. The methods and codes used to solve the arising linear systems are as follows. The groups CWI and FR used the multigrid (MG) and fast Fourier transform (FFT) algorithms, respectively. The groups ES and TUE used the FISHPACK and SuperLU solvers, respectively. The groups CN and DE used the Pardiso solver.

<sup>b</sup> This field denotes the method for solving the electron continuity equation.

<sup>c</sup> The time integration method is either explicit (exp.) or implicit (imp.)

<sup>d</sup> A dynamically (dyn.) tuned time step was used by some groups. In the simulations of the groups TUE, CN and DE, the maximum time step was set to 1 ps, 10 ps and 5 ps, respectively.

Table 3. Details of the simulations performed by different groups.

the CWI group. Note that two aspects have to be taken into account when assessing the efficiency of our code. First, the software implementation of the present method is far from optimal. The same is true for the hardware we used (e.g., the CPU clock rate can be increased). Second, the code is not parallelized. Hence, it can be expected that the computational time of our simulation scheme will be reduced, if these issues are addressed properly. In total, the present method is sufficiently fast to be used in practical applications. Currently, simulation packages used in engineering applications are mainly based on the conventional FE method and utilize implicit time integration schemes (e.g., the codes used by the groups CN and DE). It can be seen in Table 3 that this approach is not optimal from the computational point of view when applied to the considered type of problems.

### 5. Conclusions and outlook

The paper describes a spectral element method for modelling streamer discharges in low-temperature atmospheric pressure plasmas. The method is presented for the case of a Cartesian grid but can potentially be extended to unstructured meshes. For simplicity, a minimal model of the streamer is considered. The model is based on the Poisson equation for the electric potential and continuity equation for electrons (ions are treated as immobile).

The Poisson equation is solved using the hierarchical Poincaré - Steklov (HPS) scheme and the electron continuity equation is solved using the discontinuous Galerkin spectral element method (DGSEM). A subcell fine volume scheme is used to stabilize the DGSEM scheme in critical regions. The existing formulations of the HPS and DGSEM schemes are extended in two ways: (i) an integral representation of the solution is used to construct a spectral discretization scheme for the Poisson equation, (ii) an alternative procedure is proposed to approximate the diffusion flux in the DGSEM. A number of tests are considered to validate the key components of the HPS and DGSEM schemes.

The developed spectral element method is validated by reproducing the results of previous simulation studies. The computational efficiency of the method is found to be sufficient for its use in practical applications. At the same time, the computational efficiency of the proposed method is higher than that of the available finite element packages. The method is implemented in high-level programming environment. Thus, the developed computer code can easily be adapted for research or educational purposes.

The main advantage of the proposed simulation scheme is its flexibility and potential ability to simulate plasma...
discharges in complex geometries. For example, after further development, the method can be applied to complement current experimental studies on nanosecond pulsed streamer discharges [37, 38, 39] or reveal discharge properties for novel plasma jet devices [40]. The flexibility of the HPS scheme can be utilized to develop a decomposition technique for resolving boundary layers on the electrode surfaces with high accuracy. In addition, the ideas presented in this work can help to develop efficient simulation frameworks for other types of low-temperature plasma discharges.

Authorship contribution statement

I. L. Semenov: Conceptualization, Methodology, Software, Validation, Visualization, Writing - Original Draft.
K.-D. Weltmann: Funding acquisition, Project administration, Writing - Review & Editing.

Data and code availability statement

The datasets generated during this work are available at: https://doi.org/10.17632/hs4r96vydz.1
The computer code that was used to perform numerical experiments presented in this paper can be found at https://github.com/igsemenov/Streamer_HPS_DGSEM

Appendix A. Approximation of the functions $I_0, I_1$

Here, we discuss how to approximate the functions $I_0(\xi), I_1(\xi)$ given by equation (26). First, using the definition of $G(\xi, \eta)$, we derive

$$I_\alpha(\xi) = \frac{1}{2}(\xi - 1)^{1-\alpha} \int_{-1}^{\xi}(\eta + 1)f(\eta)d\eta + \frac{1}{2}(\xi + 1)^{1-\alpha} \int_{\xi}^{1}(\eta - 1)f(\eta)d\eta, \quad \alpha \in \{0, 1\}. \tag{A.1}$$

The function $f$ can be approximated by the truncated series of Chebyshev [21] or Legendre polynomials. In this work, the Legendre polynomials are used. Let $I_\alpha^{(k)}(\xi)$ for $k \in \{1, ..., n\}$ denote the functions $I_\alpha(\xi)$ with $f(\xi) = P_{k-1}(\xi)$, where $P_{k-1}(\xi)$ is the Legendre polynomial of degree $k - 1$. Assume that $f(\xi) = \sum_{k=0}^{n-1} c_k P_k(\xi)$, where $c_k$ are the Legendre expansion coefficients. In this case, we obtain

$$I_\alpha(\xi) = \sum_{k=1}^{n} c_k I_\alpha^{(k)}(\xi). \tag{A.2}$$

The integrals of $P_k(\xi), \xi P_k(\xi)$ over $[-1, 1]$ can be calculated using the recurrence relations for the Legendre polynomials [25]. Namely, we use $(2k+1)P_k(\xi) = d/d\xi [P_{k+1} - P_{k-1}], k > 0$, to obtain

$$\int_{-1}^{\xi} P_k(\eta)d\eta = [P_{k+1}(\xi) - P_{k+1}(-1) - P_{k-1}(\xi) + P_{k-1}(-1)]/(2k+1), \quad k > 0.$$ 

For $k = 0$, we have $\int_{-1}^{\xi} P_0(\eta)d\eta = \xi + 1$. The Legendre polynomials are calculated using the recurrence relation $(k+1)P_{k+1}(\xi) = (2k+1)\xi P_k(\xi) - kP_{k-1}(\xi), k > 0$, $P_0(\xi) = 1, P_1(\xi) = \xi$. The same relation is used to derive

$$\int_{-1}^{\xi} \eta P_k(\eta)d\eta = \frac{k + 1}{2k + 1} \int_{-1}^{\xi} P_{k+1}(\eta)d\eta + \frac{k}{2k + 1} \int_{-1}^{\xi} P_{k-1}(\eta)d\eta, \quad k > 0.$$ 

For $k = 0$, we get $\int_{-1}^{\xi} \eta P_0(\eta)d\eta = (\xi^2 - 1)/2$. The integrals of $P_k(\xi), \xi P_k(\xi)$ over $[\xi, 1]$ are calculated using the same procedure. Then, it is straightforward to calculate $I_0^{(k)}(\xi)$ and $I_1^{(k)}(\xi)$. Let $K_\alpha$ be the matrix defined as

$$K_\alpha = [I_\alpha^{(j)}(\xi_i)]_{1 \leq i,j \leq n}.$$
Then, using (A.2), we obtain
\[ \vec{I}_\alpha = K_\alpha \vec{c}, \]  
(A.3)
where \( \vec{c} = (c_0, ..., c_{n-1})^T \) and \( \vec{I}_\alpha = V_{1D} [I_\alpha] \).

To calculate the Legendre expansion coefficients, we use the alternative representation \( f(\xi) = \sum_{k=1}^{n} f(\xi_k) l_k(\xi) \).

Then, using equations (9) and (11), we derive
\[ \vec{c} = C \vec{f}, \]  
(A.4)
where \( \vec{f} = V_{1D} [f] \) and the matrix \( C \) is given by
\[
C = [(i - 1/2) P_{i-1}(\xi_j) \gamma_j]_{1 \leq i, j \leq n}.
\]

Finally, using equations (A.3) and (A.4), we obtain
\[ \vec{I}_\alpha = G_\alpha \vec{f}, \]  
(A.5)
where \( G_\alpha = K_\alpha C \).

In summary, equation (A.5) can be used to approximate \( I_\alpha \) at the Gauss nodes, when \( f \) is approximated by the corresponding interpolation polynomial.

**Appendix B. Validation of the spectral method for the Poisson equation**

The spectral discretization scheme presented in section 3.4.1 was tested using the method of manufactured solutions. The tests were performed for the equation of the form
\[
\frac{\partial^2 \varphi}{\partial x^2} + \frac{\partial^2 \varphi}{\partial y^2} + \frac{\alpha}{1 + x} \frac{\partial \varphi}{\partial x} = \rho, \quad (x, y) \in \Omega_E = [-1, 1]^2, \quad (B.1)
\]
where \( \alpha \in \{0, 1\} \). The Dirichlet boundary conditions were used to determine \( \varphi \) on the boundary of \( \Omega_E \).

The reference solution was chosen as \( \varphi = \sin (2x + 2y) \) for \( \alpha = 0 \) and \( \varphi = J_0(2x + 2) \sin(2y) \) for \( \alpha = 1 \), where \( J_0(\xi) \) for \( \xi \geq 0 \) is the zero order Bessel function of the first kind. Accordingly, the right-hand side of equation (B.1) was given by \( \rho = -8 \sin(2x + 2y) \) for \( \alpha = 0 \) and \( \rho = -8 J_0(2x + 2) \sin(2y) \) for \( \alpha = 1 \).

The problem was solved using two different methods: the discretization scheme presented in section 3.4.1 and the spectral collocation method used in [16]. To implement the method proposed in this work, the reference element \( \Omega_E \) was discretized with a \( n \times n \) tensor product grid of Gauss nodes. In turn, a \( n \times n \) tensor product grid of Chebyshev nodes was used to implement the spectral collocation scheme (the Chebyshev nodes are the points \( \cos [k\pi/(n - 1)] \) for \( k \in \{0, ..., n - 1\} \)).

Figure B1 shows the error in \( \varphi \) and \( \partial \varphi/\partial x \) obtained for different numbers of discretization nodes (the error in \( \partial \varphi/\partial y \) behaves similarly to the error in \( \partial \varphi/\partial x \)). The error in \( \varphi \) is defined as \( \Delta = \max_i |\delta \varphi_i| \), where \( \delta \varphi_i \) is the difference between the computed and exact solutions at the \( i \)-th discretization node. The error in \( \partial \varphi/\partial x \), denoted as \( \Delta_x \), is defined in the same way.

It can be seen in Fig. B1 that the accuracy of our method is close to the accuracy of the spectral collocation scheme (when comparing the results for the same number of nodes in the interior of \( \Omega_E \)). Numerous tests (not presented here) showed that the same holds for other second-order equations as well.

Nevertheless, our approach has two advantages. First, the values of \( \varphi \) at the corners of \( \Omega_E \) can be eliminated during the construction of the solution operator. This simplifies the implementation of the HPS scheme. Second, the discretization scheme can be formulated using the values of the solution at the internal Gauss nodes only. This makes it easier to join the HPS solver with the DGSEM scheme for the electron continuity equation.
Fig. B1. Error in the solution and its derivative for different numbers of discretization nodes. The results were obtained using the spectral collocation method (SCM) and the discretization scheme proposed in this work.

Appendix C. Validation of the HPS solver

The spectral element solver for the Poisson equation was tested using the method of manufactured solutions. The tests were performed for the equation of the form

$$\frac{\partial^2 \varphi}{\partial x^2} + \frac{\partial^2 \varphi}{\partial y^2} + \frac{\alpha}{x} \frac{\partial \varphi}{\partial x} = \rho, \quad (x, y) \in \Omega,$$

where $\alpha = 0$ corresponds to the formulation in Cartesian coordinates and $\alpha = 1$ corresponds to the problems with axial symmetry. The Dirichlet boundary conditions were used to determine $\varphi$ on the boundary of $\Omega$. All computations were performed for $R_\Omega = 4$.

The reference solution was chosen as $\varphi = \sin(x+y)$ for $\alpha = 0$, and $\varphi = J_0(x) \sin(y)$ for $\alpha = 1$. Accordingly, the right-hand side of equation (C.1) was given by $\rho = -2 \sin(x+y)$ for $\alpha = 0$, and $\rho = -2J_0(x) \sin(y)$ for $\alpha = 1$. The computational domain was discretized uniformly, i.e., the computational mesh was represented by a perfect binary tree. For this case, the length of the finite element edge is given by $R_\Omega/2^k$, with $k \geq 0$. Each finite element was discretized with a $n \times n$ tensor product grid of Gauss nodes.

In the present work, the spectral element method for the Poisson equation is combined with the DGSEM for the electron continuity equation. Typically, the number of Gauss nodes per direction used in the DGSEM scheme is relatively low. For this reason, we present the computation results for $n = 4$ and $n = 6$ only. But, similar to [16, 17, 18, 19], higher values of $n$ can be used as well.

Tables C1, C2 report the global $L^2$ error in $\varphi$, $\partial \varphi / \partial x$, $\partial \varphi / \partial y$ for $n = 4$ and $n = 6$, respectively. The results are presented for different values of $k$ to demonstrate the convergence properties of the method. The $L^2$ error in $\varphi$ is defined as $\Delta = \sqrt{\sum_i \int_{\Omega_i} (\delta \varphi)^2 dx dy}$, where $\Omega_i$ is the $i$-th finite element and $\delta \varphi$ is the difference between the computed and exact solutions. The integral of $(\delta \varphi)^2$ over $\Omega_i$ is calculated using the Gauss integration method.
$L^2$ errors in $\partial \varphi / \partial x$, $\partial \varphi / \partial y$, denoted as $\Delta_x$, $\Delta_y$, respectively, are defined in the same way. Note that $\Delta_x$ and $\Delta_y$ are almost identical at $\alpha = 0$ (thus, only $\Delta_x$ is reported for this case).

It can be seen that the computed solution converges to the exact one as the mesh is refined. The observed order of convergence is close to $n$ for both the solution and its derivatives. The similar rate of convergence was observed for the maximum absolute value of the error in $\varphi$, $\partial \varphi / \partial x$, $\partial \varphi / \partial y$ (i.e., for the discrete $L^\infty$ error).

| $\alpha = 0$ | $\alpha = 1$ |
|--------------|--------------|
| $k$ | $\Delta$ | $O_{L^2}$ | $\Delta_x$ | $O_{L^2}$ | $\Delta_y$ | $O_{L^2}$ | $\Delta_y$ | $O_{L^2}$ |
| 3 | 2.8605e-03 | 2.8882e-03 | 1.3598e-03 | 2.8882e-03 | 1.6215e-03 |
| 4 | 1.8519e-04 | 1.8385e-04 | 4.6060e-05 | 1.8385e-04 | 4.93 | 4.7052e-05 |
| 5 | 1.1682e-05 | 1.1514e-05 | 2.8090e-06 | 1.1541e-05 | 3.58 | 4.0750e-06 |
| 6 | 7.3183e-07 | 7.2206e-07 | 1.7721e-07 | 7.2206e-07 | 4.72 | 1.6052e-07 |
| 7 | 4.5759e-08 | 4.5134e-08 | 1.1437e-08 | 4.5134e-08 | 3.92 | 1.0104e-08 |

\[\text{Table C1. Error in the solution and its derivatives for different refinement levels at } n = 4.\]

| $\alpha = 0$ | $\alpha = 1$ |
|--------------|--------------|
| $k$ | $\Delta$ | $O_{L^2}$ | $\Delta_x$ | $O_{L^2}$ | $\Delta_y$ | $O_{L^2}$ | $\Delta_y$ | $O_{L^2}$ |
| 2 | 4.1337e-04 | 4.2737e-04 | 1.0853e-03 | 4.2737e-04 | 8.5881e-04 |
| 3 | 7.2378e-06 | 7.3028e-06 | 3.6501e-06 | 7.3028e-06 | 7.34 | 5.0464e-06 |
| 4 | 1.1698e-07 | 1.1162e-07 | 2.6455e-08 | 1.1612e-07 | 7.68 | 2.6776e-08 |
| 5 | 1.8427e-09 | 1.8204e-09 | 4.0970e-10 | 1.8204e-09 | 6.32 | 3.6791e-10 |
| 6 | 2.3722e-11 | 2.3416e-11 | 7.0772e-12 | 2.3416e-11 | 5.94 | 6.2493e-12 |

\[\text{Table C2. Error in the solution and its derivatives for different refinement levels at } n = 6.\]

### Appendix D. Validation of the numerical flux function

To validate the numerical flux function proposed in section 3.5.1, we performed a number of tests for the equation of the form

$$\frac{\partial u}{\partial t} + a \frac{\partial u}{\partial x} = \nu \frac{\partial^2 u}{\partial x^2}, \quad (x,t) \in [-\pi, \pi] \times [0,T], \quad T > 0, \quad (D.1)$$

where $a$, $\nu$ are the advection velocity and diffusion coefficient, respectively. The initial condition for equation (D.1) is defined as $u(x,0) = \sin(x)$. The periodic boundary conditions are used at $t > 0$. The exact solution of the considered problem is $u(x,t) = \sin(x - at) \exp(-\nu t)$. Note that similar test problems were used in [41, 42].

The interval $[-\pi, \pi]$ was divided into $k$ equal finite elements and each finite element was discretized with a grid of $n$ Gauss nodes. Equation (D.1) was discretized using the DGSEM scheme analogous to that described in section 3.5.2. The numerical flux at the boundary points of finite elements was computed using equation (49).

The resulting set of equations was discretized in time using a third-order total variation diminishing Runge-Kutta scheme [35]. For the equation of the form $\frac{du}{dt} = T(u)$, one step of this scheme is given by

\[u_1 = u_0 + \Delta t T(u_0), \]
\[u_2 = \left[3u_0 + u_1 + \Delta t T(u_1)\right] / 4, \]
\[u_3 = \left[u_0 + 2u_2 + 2\Delta t T(u_2)\right] / 3, \quad (D.2)\]
where $\Delta t$ is the time step and $u_0 \approx u(t_0)$, $u_3 \approx u(t_0 + \Delta t)$, with $t_0$ being the time moment. The time step was set to $\Delta t = 2.5 \times 10^{-5}$ and the solution was analyzed at $T = 0.5$.

The computations were performed for $n = 4$, $n = 6$ and different values of $a, \nu$. The number of elements was varied to examine the convergence of the solution. Tables D1 - D5 report the computed $L^\infty$ and $L^2$ errors in $u$. The $L^\infty$ error is defined as $\Delta L^\infty = \max_i |\delta u_i|$, where $\delta u_i$ is the difference between the computed and exact solutions at the $i$-th discretization node. The $L^2$ error is defined as $\Delta L^2 = \sqrt{\sum_i \int (\delta u)^2 dx}$, where $\Omega_i$ is the $i$-th finite element and $\delta u$ is the difference between the computed and exact solutions. The integral of $(\delta u)^2$ over $\Omega_i$ is calculated using the Gauss quadrature rule.

It can be seen that the computed solution converges to the exact one as the mesh is refined. The order of convergence is close to $n$ in all cases. These results indicate that the numerical flux function proposed in section 3.5.1 can be used to construct a consistent discontinuous Galerkin scheme.

| $k$ | $\Delta L^\infty$ | $O L^\infty$ | $\Delta L^2$ | $O L^2$ | $\Delta L^\infty$ | $O L^\infty$ | $\Delta L^2$ | $O L^2$ |
|-----|------------------|-------------|-------------|---------|------------------|-------------|-------------|---------|
| 4   | 1.9814e-03       | 3.8085e-03  | 1.1537e-05  |         | 1.6430e-05       |             |             |         |
| 8   | 1.3516e-04       | 3.87       | 1.8905e-07  | 5.93    | 2.2603e-07       | 6.18        |             |         |
| 16  | 8.6618e-06       | 3.96       | 2.9688e-09  | 5.99    | 3.4100e-09       | 6.05        |             |         |
| 32  | 5.4485e-07       | 3.99       | 4.6408e-11  | 6.00    | 5.2788e-11       | 6.01        |             |         |
| 64  | 3.4108e-08       | 4.00       | 7.1665e-13  | 6.02    | 8.1437e-13       | 6.02        |             |         |

Table D1. Error in the solution for different numbers of elements at $a = 0$, $\nu = 1$.

| $k$ | $\Delta L^\infty$ | $O L^\infty$ | $\Delta L^2$ | $O L^2$ | $\Delta L^\infty$ | $O L^\infty$ | $\Delta L^2$ | $O L^2$ |
|-----|------------------|-------------|-------------|---------|------------------|-------------|-------------|---------|
| 4   | 2.3320e-03       | 3.2685e-03  | 1.4140e-05  |         | 1.5454e-05       |             |             |         |
| 8   | 1.3974e-04       | 4.06       | 1.9117e-07  | 6.13    | 2.2139e-07       | 6.13        |             |         |
| 16  | 8.7932e-06       | 3.99       | 2.9687e-09  | 6.01    | 3.3814e-09       | 6.03        |             |         |
| 32  | 5.4755e-07       | 4.01       | 4.6411e-11  | 6.00    | 5.2663e-11       | 6.00        |             |         |
| 64  | 3.4297e-08       | 4.00       | 7.2553e-13  | 6.00    | 8.2394e-13       | 6.00        |             |         |

Table D2. Error in the solution for different numbers of elements at $a = 1$, $\nu = 1$.

| $k$ | $\Delta L^\infty$ | $O L^\infty$ | $\Delta L^2$ | $O L^2$ | $\Delta L^\infty$ | $O L^\infty$ | $\Delta L^2$ | $O L^2$ |
|-----|------------------|-------------|-------------|---------|------------------|-------------|-------------|---------|
| 4   | 1.2733e-03       | 2.1542e-03  | 8.1585e-06  |         | 1.2834e-05       |             |             |         |
| 8   | 1.2521e-04       | 3.35       | 1.9374e-07  | 5.89    | 2.1648e-07       | 5.89        |             |         |
| 16  | 8.0673e-06       | 3.96       | 2.6911e-09  | 6.04    | 3.2852e-09       | 6.04        |             |         |
| 32  | 5.2309e-07       | 3.95       | 4.1181e-11  | 6.04    | 4.9976e-11       | 6.04        |             |         |
| 64  | 3.1512e-08       | 4.05       | 6.3216e-13  | 6.02    | 7.7196e-13       | 6.02        |             |         |

Table D3. Error in the solution for different numbers of elements at $a = 1$, $\nu = 0.1$.  

34
Table D4. Error in the solution for different numbers of elements at \( a = 1, \nu = 0.01 \).

| \( k \) | \( \Delta L_\infty \) | \( O_{L_\infty} \) | \( \Delta L_2 \) | \( O_{L_2} \) | \( \Delta L_\infty \) | \( O_{L_\infty} \) | \( \Delta L_2 \) | \( O_{L_2} \) |
|------|----------------|----------|----------------|----------|----------------|----------|----------------|----------|
| 4    | 1.5336e-03    | 2.6525e-03 | 7.0549e-06    | 1.2360e-05 |
| 8    | 1.1667e-04    | 3.72      | 1.3671e-07    | 5.69     | 1.9331e-07    | 6.00     |
| 16   | 7.2128e-06    | 4.02      | 2.1411e-09    | 5.78     | 3.1659e-09    | 5.93     |
| 32   | 4.9283e-07    | 3.87      | 4.3061e-11    | 5.91     | 5.2555e-11    | 5.91     |
| 64   | 3.2354e-08    | 3.93      | 7.8459e-13    | 5.99     | 8.2795e-13    | 5.99     |

Table D5. Error in the solution for different numbers of elements at \( a = 1, \nu = 0.001 \).

| \( k \) | \( \Delta L_\infty \) | \( O_{L_\infty} \) | \( \Delta L_2 \) | \( O_{L_2} \) | \( \Delta L_\infty \) | \( O_{L_\infty} \) | \( \Delta L_2 \) | \( O_{L_2} \) |
|------|----------------|----------|----------------|----------|----------------|----------|----------------|----------|
| 4    | 1.5932e-03    | 2.7794e-03 | 7.5029e-06    | 1.3135e-05 |
| 8    | 1.3358e-04    | 3.58      | 1.3651e-07    | 5.78     | 1.9863e-07    | 6.05     |
| 16   | 7.0061e-06    | 4.25      | 1.9460e-09    | 6.13     | 2.9233e-09    | 6.09     |
| 32   | 4.0061e-07    | 4.13      | 3.2117e-11    | 5.92     | 4.6507e-11    | 5.97     |
| 64   | 2.5088e-08    | 4.00      | 5.2414e-13    | 5.98     | 7.3832e-13    | 5.98     |

References

[1] S. M. Starikovskaia, Plasma assisted ignition and combustion, Journal of Physics D: Applied Physics 39 (2006) R265–R299.

[2] I. V. Adamovich, W. R. Lempert, Challenges in understanding and predictive model development of plasma-assisted combustion, Plasma Physics and Controlled Fusion 57 (2014) 014001.

[3] E. Moreau, Airflow control by non-thermal plasma actuators, Journal of Physics D: Applied Physics 40 (2007) 605–636.

[4] T. C. Corke, C. L. Enloe, S. P. Wilkinson, Dielectric barrier discharge plasma actuators for flow control, Annual Review of Fluid Mechanics 42 (2010) 505–529.

[5] K.-D. Weltmann, T. von Woedtke, Plasma medicine – current state of research and medical application, Plasma Physics and Controlled Fusion 59 (2016) 014031.

[6] P. J. Bruggeman, M. J. Kushner, B. R. Locke, J. G. E. Gardeniers, W. G. Graham, D. B. Graves, R. C. H. M. Hofman-Caris, D. Maric, J. P. Reid, E. Ceriani, D. F. Rivas, J. E. Foster, S. C. Garrick, Y. Gorbanev, S. Hamaguchi, F. Iza, H. Jablonowski, E. Klimova, J. Kolb, F. Krcma, P. Lukes, Z. Machala, I. Marinov, D. Mariotti, S. M. Thagard, D. Minakata, E. C. Neyts, J. Pawlat, Z. L. Petrovic, R. Pfieger, S. Reuter, D. C. Schram, S. Schröter, M. Shiraiwa, B. Tarabová, P. A. Tsai, J. R. R. Verlet, T. von Woedtke, K. R. Wilson, K. Yasui, G. Zvereva, Plasma–liquid interactions: a review and roadmap, Plasma Sources Science and Technology 25 (2016) 053002.

[7] S. Nijdam, J. Teunissen, U. Ebert, The physics of streamer discharge phenomena, Plasma Sources Science and Technology 29 (2020) 103001.
[8] 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, M. Yousfi, Comparison of six simulation codes for positive streamers in air, Plasma Sources Science and Technology 27 (2018) 095002.

[9] P. Viegas, F. Pêchereau, A. Bourdon, Numerical study on the time evolutions of the electric field in helium plasma jets with positive and negative polarities, Plasma Sources Science and Technology 27 (2018) 025007.

[10] P. Viegas, E. Slikboer, A. Obrusník, Z. Bonaventura, A. Sobota, E. Garcia-Caurel, O. Guaitella, A. Bourdon, Investigation of a plasma–target interaction through electric field characterization examining surface and volume charge contributions: modeling and experiment, Plasma Sources Science and Technology 27 (2018) 094002.

[11] J. Teunissen, U. Ebert, Simulating streamer discharges in 3D with the parallel adaptive Afivo framework, Journal of Physics D: Applied Physics 50 (2017) 474001.

[12] J. Teunissen, U. Ebert, Afivo: A framework for quadtree/octree AMR with shared-memory parallelization and geometric multigrid methods, Computer Physics Communications 233 (2018) 156–166.

[13] J.-M. Plewa, O. Eichwald, O. Ducasse, P. Dessante, C. Jacobs, N. Renon, M. Yousfi, 3D streamers simulation in a pin to plane configuration using massively parallel computing, Journal of Physics D: Applied Physics 51 (2018) 095206.

[14] R. Marskar, An adaptive cartesian embedded boundary approach for fluid simulations of two- and three-dimensional low temperature plasma filaments in complex geometries, Journal of Computational Physics 388 (2019) 624–654.

[15] B. Lin, C. Zhuang, Z. Cai, R. Zeng, W. Bao, An efficient and accurate MPI-based parallel simulator for streamer discharges in three dimensions, Journal of Computational Physics 401 (2020) 109026.

[16] P. Martinsson, A direct solver for variable coefficient elliptic PDEs discretized via a composite spectral collocation method, Journal of Computational Physics 242 (2013) 460–479.

[17] A. Gillman, P. G. Martinsson, A direct solver with $O(N)$ complexity for variable coefficient elliptic PDEs discretized via a high-order composite spectral collocation method, SIAM Journal on Scientific Computing 36 (2014) A2023–A2046.

[18] S. Hao, P.-G. Martinsson, A direct solver for elliptic PDEs in three dimensions based on hierarchical merging of Poincaré–Steklov operators, Journal of Computational and Applied Mathematics 308 (2016) 419–434.

[19] P. Geldermans, A. Gillman, An adaptive high order direct solution technique for elliptic boundary value problems, SIAM Journal on Scientific Computing 41 (2019) A292–A315.

[20] D. Fortunato, N. Hale, A. Townsend, The ultraspherical spectral element method, Journal of Computational Physics 436 (2021) 110087.

[21] L. Greengard, V. Rokhlin, On the numerical solution of two-point boundary value problems, Communications on Pure and Applied Mathematics 44 (1991) 419–452.

[22] W. Leeb, V. Rokhlin, On the numerical solution of fourth-order linear two-point boundary value problems, SIAM Journal on Scientific Computing 42 (2020) A1789–A1808.

[23] N. Krais, A. Beck, T. Bolemann, H. Frank, D. Flad, G. Gassner, F. Hindenlang, M. Hoffmann, T. Kuhn, M. Sonntag, C.-D. Munz, FLEXI: A high order discontinuous Galerkin framework for hyperbolic–parabolic conservation laws, Computers & Mathematics with Applications 81 (2021) 186–219.
[24] F. Hindenlang, G. J. Gassner, C. Altmann, A. Beck, M. Staudenmaier, C.-D. Munz, Explicit discontinuous Galerkin methods for unsteady problems, Computers & Fluids 61 (2012) 86–93.

[25] W. W. Bell, Special functions for scientists and engineers, Courier Corporation, 2004.

[26] D. A. Kopriva, A conservative staggered-grid Chebyshev multidomain method for compressible flows. II. A semi-structured method, Journal of Computational Physics 128 (1996) 475–488.

[27] D. A. Kopriva, G. Gassner, On the quadrature and weak form choices in collocation type discontinuous Galerkin spectral element methods, Journal of Scientific Computing 44 (2010) 136–155.

[28] F. Bassi, S. Rebay, A high-order accurate discontinuous finite element method for the numerical solution of the compressible Navier–Stokes equations, Journal of Computational Physics 131 (1997) 267–279.

[29] B. Cockburn, C.-W. Shu, The local discontinuous Galerkin method for time-dependent convection-diffusion systems, SIAM Journal on Numerical Analysis 35 (1998) 2440–2463.

[30] D. N. Arnold, F. Brezzi, B. Cockburn, L. D. Marini, Unified analysis of discontinuous Galerkin methods for elliptic problems, SIAM Journal on Numerical Analysis 39 (2002) 1749–1779.

[31] G. Gassner, F. Lörcher, C.-D. Munz, A contribution to the construction of diffusion fluxes for finite volume and discontinuous Galerkin schemes, Journal of Computational Physics 224 (2007) 1049–1063.

[32] S. Jöns, C.-D. Munz, An approximate Riemann solver for advection–diffusion based on the generalized Riemann problem, Communications on Applied Mathematics and Computation 2 (2019) 515–539.

[33] P. E. Johnson, E. Johnsen, The compact gradient recovery discontinuous Galerkin method for diffusion problems, Journal of Computational Physics 398 (2019) 108872.

[34] B. Koren, A robust upwind discretization method for advection, diffusion and source terms, in: C. B. Vreugdenhil, B. Koren (Eds.), Numerical Methods for Advection-Diffusion Problems, Vieweg, Braunschweig/Wiesbaden, 1993, pp. 117–138.

[35] S. Gottlieb, C.-W. Shu, Total variation diminishing Runge-Kutta schemes, Mathematics of Computation of the American Mathematical Society 67 (1998) 73–85.

[36] D. Bessières, J. Paillol, A. Bourdon, P. Ségur, E. Marode, A new one-dimensional moving mesh method applied to the simulation of streamer discharges, Journal of Physics D: Applied Physics 40 (2007) 6559–6570.

[37] T. Huiskamp, Nanosecond pulsed streamer discharges Part I: Generation, source-plasma interaction and energy-efficiency optimization, Plasma Sources Science and Technology 29 (2020) 023002.

[38] H. Höft, M. M. Becker, J. F. Kolb, T. Huiskamp, Double-propagation mode in short-gap spark discharges driven by HV pulses with sub-ns rise time, Plasma Sources Science and Technology 29 (2020) 085002.

[39] S. Jahanbakhsh, V. Brüser, R. Brandenburg, Experimental investigation of single microdischarges in a barrier corona arrangement with a cathodic metal pin, Plasma Sources Science and Technology 29 (2020) 015001.

[40] X. Lu, S. Reuter, M. Laroussi, D. Liu, Nonequilibrium Atmospheric Pressure Plasma Jets, CRC Press, 2019.

[41] M. Zhang, C.-W. Shu, An analysis of three different formulations of the discontinuous Galerkin method for diffusion equations, Mathematical Models and Methods in Applied Sciences 13 (2003) 395–413.

[42] G. Gassner, F. Lörcher, C.-D. Munz, A contribution to the construction of diffusion fluxes for finite volume and discontinuous Galerkin schemes, Journal of Computational Physics 224 (2007) 1049–1063.