Fluid plasma simulations with cut-cell adaptive mesh refinement

Abstract

A parallelized computational framework for planar two- and fully three-dimensional plasma fluid simulations of gaseous electrical discharges is investigated. Our approach uses cut-cell, multilevel grids with embedded boundaries for accurately describing electrode and dielectric surfaces. The plasma equation are then solved in space by using finite volume methods over this discretization, and are then evolved transiently using linear multistep methods. Numerical examples of two- and three-dimensional propagation in centimeter-sized gaps are then provided, with an emphasis on the formalism used.

1. Introduction

Streamers are non-thermal, self-propagating weakly ionized plasma filaments with bright (i.e. radiating) heads and are precursors to electrical sparks, sprites, and lightning. A streamer moves as an ionization wave due to space-charge screening of the plasma filament and electron impact ionization in the streamer head that leads to self-enhancement of the electric field at the tip fo the filament.

Computational modelling of propagating weakly ionized plasmas is inherently difficult due to length scales that differ by several orders of magnitude. For example, in air at atmospheric pressure the plasma filament is surrounded by a space-charge layer with a thickness of tens to hundreds of micrometers, while the streamer itself can propagate tens of centimeters. Such large aspect ratios set particularly stringent requirements on the spatial grid. The presence of numerical stiffness in the equations of motion further complicates numerical modeling. In the plasma modeling community, it is commonly accepted that adaptive mesh refinement (AMR) is required for numerically resolving such phenomena, especially in three dimensions. The inclusion of internal boundary conditions, such as electrodes and dielectric materials inside the computational domain leads to additional numerical complexities, and it essential that these are resolved in a way that remain compatible with AMR.

Other difficulties of a more fundamental nature are related to the treatment of the various species (neutral, ions, electrons). For weakly ionized plasmas, neutrals and ions are usually close to equilibrium, and the velocity distribution function (VDF) is almost Maxwellian so that they can be treated in the continuum (i.e. fluid) approximation. This is not necessarily the case for electrons whose VDF may be far from Maxwellian and may therefore require a kinetic (i.e. particle based) treatment. Hybrid models that combine kinetic and fluid approaches are, strictly speaking, still in their infancy but are starting to gain traction in the plasma community. The stability of such methods have been a challenge due to the inherent statistical noise that is present in e.g. Direct Monte Carlo Simulations (DMCS) or Particle-In-Cell (PIC) based approaches. Methods for direct solutions of the Boltzmann equation are also being developed. The Discrete Velocity Method, for example, reduces the Boltzmann equation to a system of hyperbolic equations for a set of velocities and is therefore a direct analogue of the Discrete Ordinates Method (DOM) for the radiative transfer equation (RTE). The DVM is, strictly speaking, different from Lattice-Boltzmann methods due to the number of velocities that are included in the discretization of phase space (usually 10^3 for DVM versus 10^1 for LBM). Nonetheless, simplified plasma models are still of great interest because of their simplicity and their straightforward compatibility with embedded boundary conditions.

In this paper, we discuss implementations of fluid plasma models on cut-cell Cartesian grids and their applications. Cut-cell grids resolve some of the disadvantages that are commonly encounted with unstructured, body fitted grids, such as convergence loss with skewed elements, and lack of efficient automatic mesh adaption. The purpose of this paper is to derive a software platform that is as general as possible within

the minimal plasma model, while simultaneously demonstrate how we resolve some of the numerical issues that one encounter on adaptive cut-cell grids. The results of these efforts is a scalable software platform for fluid plasma simulations; we demonstrate it's use for various plasma kinetic schemes in different geometries.

The structure of this paper is as follows: In Sec. 2 we review the minimal plasma model and present the equations of motion. We outline our numerical methods in Sec. 3 and provide numerical examples in Sec. 5. Final remarks regarding the minimal streamer model are given in Sec. 6.

2. Theory

The minimal plasma model consists of charge (or mass) conservation for the massive species (i.e. electrons, ions, and neutral molecules), the Poisson equation for the electric field, radiative transfer equations for photon transport, and a scalar conservation equation for charge conservation on dielectric surfaces.

2.1. Electrodynamic equations

For a weakly ionized plasma such as a streamer, the streamer current is sufficiently small so that one may safely disregard magnetic field effects. The Maxwell equation for the evolution of the electromagnetic field is Poisson's equation:

$$\nabla \cdot (\epsilon \nabla \Phi) = -\frac{\rho}{\epsilon_0}, \tag{1}$$

where Φ is the electric potential, ϵ the relative permittivity, and ρ the charge density. The remaining Maxwell equation is $\boldsymbol{E} = -\nabla\Phi$ is $\boldsymbol{J} + \epsilon_0 \frac{\partial \boldsymbol{E}}{\partial t} = 0$, but we only use this for estimating a time step size which ensures numerical stability during our simulations.

2.2. Plasma-fluid equations

The spatiotemporal evolution of electrons, ions, and neutrals is solved in the plasma-fluid approximation. We use the minimal plasma model which solves for mass conservation for each species μ:

$$\frac{\partial n_\mu}{\partial t} + \nabla \cdot (\boldsymbol{v}_\mu n_\mu - D_\mu \nabla n_\mu) = S_\mu, \tag{2}$$

where n_μ is the volumetric density of species μ; $\boldsymbol{v}_\mu$ its drift velocity, D_μ its diffusion coefficient, and S_μ a source term (describing the interplay between attachment, impact ionization, photoionization etc.). Equation (2) is a convection-diffusion-reaction (CDR) equation which describes the evolution of individual species densities.

Equation (2) is supplemented by boundary conditions describing the mass flux into or out of the computational domain. Our approach is to use outflow conditions on domain walls, and to expose the boundary conditions on embedded electrodes and dielectrics in an abstract plasma-kinetic framework which allows us to modify the surface kinetics of the plasma without affecting the underlying solver code.

2.3. Radiative transfer

Radiative transfer is handled by solving the radiative transfer equation (RTE) in the diffusive approximation. The RTE is

$$\frac{\partial f_\nu(\boldsymbol{x}, \boldsymbol{\Omega}, t)}{\partial t} + c\boldsymbol{\Omega} \cdot \nabla f_\nu(\boldsymbol{x}, \boldsymbol{\Omega}, t) = -c\kappa_\nu(\boldsymbol{x}) f_\nu(\boldsymbol{x}, \boldsymbol{\Omega}, t) + \frac{1}{4\pi}\eta_\nu(\boldsymbol{x}), \tag{3}$$

where f_ν is the photon distribution function (i.e. the number of photon with frequency ν at $(\boldsymbol{x}, t)$ traveling in direction $\boldsymbol{\Omega}$), κ_ν is the Beer's length for photons ν, and η_ν is an isotropic source term. Thus, η_ν is the total number of photons produced at $(\boldsymbol{x}, t)$ (hence the factor of 4π).

2.3.1. The multigroup approximation

In the RTE, the frequency ν is generally speaking a continuous variable. For most applications it becomes necessary to reduce the computational load by invoking the monochromatic multigroup approximation. That is, one assumes that f_ν consists of a number of frequency bands γ where each frequency band is sufficiently sharp-line in order to individually invoke a monochromatic approximation for each frequency band. That is, $f_\nu = \sum_\gamma f_\gamma \delta(\gamma)$ which essentially replaces Eq. (3) with a finite set of equations for each frequency band γ:

$$\frac{\partial f_\gamma\left(\boldsymbol{x},\boldsymbol{\Omega},t\right)}{\partial t} + c\boldsymbol{\Omega}\cdot\nabla f_\gamma\left(\boldsymbol{x},\boldsymbol{\Omega},t\right) = -c\kappa_\gamma(\boldsymbol{x}) f_\gamma\left(\boldsymbol{x},\boldsymbol{\Omega},t\right) + \frac{1}{4\pi}\eta_\gamma(\boldsymbol{x}), \tag{4}$$

The multigroup RTE is solved in the diffusive SP_1 approximation (i.e. the *Eddington* approximation) by closing the first order moment equation. That is, taking the first and second moments of Eq. (3) with respect to $\boldsymbol{\Omega}$ yields

$$\frac{\partial E_\gamma}{\partial t} + \nabla\cdot\boldsymbol{F}_\gamma = -c\kappa_\gamma E_\gamma + \eta_\gamma, \tag{5a}$$

$$\frac{\partial \boldsymbol{F}_\gamma}{\partial t} + \nabla\cdot\boldsymbol{\Pi}_\gamma = -\kappa_\gamma \boldsymbol{F}_\gamma, \tag{5b}$$

where $E_\gamma = \int_{4\pi} f_\gamma \mathrm{d}\Omega$ is the radiative density, $\boldsymbol{F}_\gamma = c\int_{4\pi} f_\gamma \boldsymbol{\Omega}\mathrm{d}\Omega$ is the radiative flux, and $\Pi_\gamma^{ij} = c\int_{4\pi} f_\gamma \Omega_i\Omega_j \mathrm{d}\Omega$ is the Eddington tensor. This system is closed by assuming

$$\Pi_\gamma^{ij} = \frac{c}{3}\delta_{ij}E_\gamma, \tag{6}$$

which is equivalent to the Eddington approximation. Insertion of Eq. (6) into Eq. (5b) in the stationary approximation $\partial_t f_\gamma = 0$ yields the photon flux

$$\boldsymbol{F}_\gamma = -\frac{c}{3\kappa_\gamma}\nabla E_\gamma, \tag{7}$$

which is what one expects from a diffusive approximation. Insertion of Eq. (7) into Eq. (5a) yields

$$\nabla\cdot\left(\frac{1}{3\kappa_\gamma}\nabla E_\gamma\right) - \kappa_\gamma E_\gamma = \frac{\eta_\gamma}{c}. \tag{8}$$

This equation must be supplemented by appropriate boundary conditions for radiative transport. In the Eddington approximation, the appropriate outflow boundary is of the Robin type:

$$\partial_n E_\gamma + \frac{3\kappa_\gamma}{2}\frac{1+3r_2}{1-2r_1}E_\gamma = \frac{g}{\kappa_\gamma}, \tag{9}$$

where r_1, and r_2 are reflection coefficients and g is a surface source. For most of our applications we do not consider reflection of photons from boundaries, nor injection of photons into the domain, so we normally take $r_1 = r_2 = g = 0$ (although our implementation of Eq (9) is not restricted to this). The outflow boundary condition on photons on a boundary with outward (i.e. out of the gas-phase) normal vector $\boldsymbol{n}$ therefore simplifies to

$$\frac{\partial E_\gamma}{\partial n} + \frac{3\kappa_\gamma}{2}E_\gamma = 0, \tag{10}$$

where $\partial_n E_\gamma = \boldsymbol{n}\cdot\nabla E_\gamma$. Thus, for free outflow of photons the boundary flux is

$$\boldsymbol{F}_\gamma\cdot\boldsymbol{n} = \frac{c}{2}E_\gamma. \tag{11}$$

2.4. Surface charge conservation

Our final equation of motion is local conservation of charge on *dielectric* surfaces. We do not solve an equivalent problem on electrodes because we assume that complete neutralization occurs on both anodes and cathodes, and that the voltage on the live and grounded electrodes are maintained by an external circuit. Charge conservation on dielectric surfaces is given by:

$$\frac{\partial \sigma}{\partial t} = F_\sigma, \tag{12}$$

where F_σ is the charge flux out of the surface and into the gas phase. F_σ is always coupled to the boundary conditions for the species densities n_μ since charge must be conserved at the surface. Thus, F_σ is simply the sum of all species fluxes at the boundary.

2.5. Minimal plasma model

To summarize, our plasma model consists of the following equations:

$$\nabla \cdot (\epsilon_r \nabla)\phi = -\frac{\rho}{\epsilon_0}, \tag{13a}$$

$$\frac{\partial \sigma}{\partial t} = F_\sigma, \tag{13b}$$

$$\nabla \cdot \left(\frac{1}{3\kappa_\gamma}\nabla E_\gamma\right) - \kappa_\gamma E_\gamma = -\frac{\eta_\gamma}{c}, \tag{13c}$$

$$\frac{\partial n_\mu}{\partial t} + \nabla \cdot (\boldsymbol{v}_\mu n_\mu) - \nabla \cdot (D_\mu \nabla n_\mu) = S_\mu, \tag{13d}$$

where the final two equations denote *sets* of equations for μ and γ.

For streamer discharges Eq. (13) is nonlinearly coupled through source terms and species velocities. Our approach has been to develop a code framework that exposes only the *coupling* rather than the actual solvers, so that the user can specify his own bulk and surface plasma kinetics within the minimal model. For this reason we discuss the discretization process in the next section, and discuss specific cases in Sec. 5.

3. Numerical methods

Today, two main methods exist within Cartesian AMR. The first is category is patch-based AMR (see Fig. 1) where the domain is subdivided into a collection of hierarchically nested overlapping patches. Each patch is a rectangular block of cells which, in space, exists on a subdomain of a larger patch with a coarser resolution. The most obvious advantage of patch-based AMR is that entire Cartesian blocks can be sent into solvers, while two notable disadvantages are that i) Overlapping grids can inflate memory and ii) Refinement occurs on a patch-by-patch basis, which can lead to over-refinement of regions of interest. The second category is octree AMR. In essence, octrees are data structures that define a hierarchy of parent/children relations and allow one to rapidly navigate the mesh. Typically, in octree AMR the leaves of an octree are *cells*, and the mesh is refined on a cell-by-cell basis. The most obvious advantage of octree AMR is that refined cells are non-overlapping, so that when a cell is bisected, the parent cell is destroyed. A third category that combine patch and octree based AMR also exists. Hybrid octree/patch based AMR use patches instead of cells as the leaves in an octree layout, and in essence yields patch based AMR without the use of overlapping grids.

We have implemented Eq. (13) into the Chombo library. In essence, Chombo is a library for performing patch-based AMR and also includes support for cut-cells. In addition, Chombo comes with some templated solvers (such as geometric multigrid), and several examples that can be used as stepping stones for more complex codes.

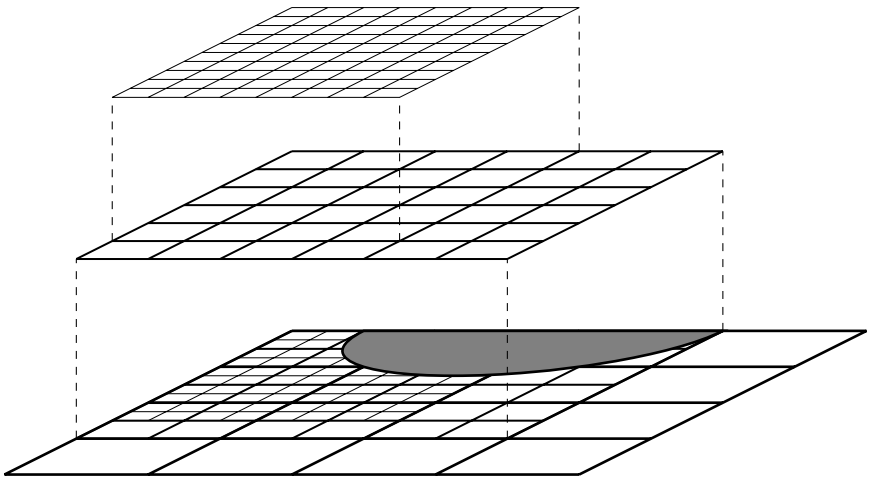

Figure 1: Patch-based AMR. The figure shows two refined grids that hierarchically overlap a base grid. On each grid, boundary cells are cut by an external surface (shaded region).

3.1. Spatial discretization

Here, we give a brief summary to the spatial discretization. We discretize the physical domain by a set of Cartesian control volumes centered on points $\boldsymbol{i} = (i_0, \ldots, i_{\boldsymbol{D}-1})^\intercal \in \mathbb{Z}^{\boldsymbol{D}}$ where $\boldsymbol{D}$ is the dimension (typically, $\boldsymbol{D} = 2$ or $\boldsymbol{D} = 3$). Each control volume occupies the region $[\boldsymbol{i} - \frac{1}{2}\boldsymbol{u}\Delta x, \boldsymbol{i} + \frac{1}{2}\boldsymbol{u}\Delta x]$ where Δx is a resolution and $\boldsymbol{u}$ is a vector whose components are all one (i.e. $\boldsymbol{u} = (1,1)^\intercal$ in 2D). Thus, a mesh $\Gamma \subset \mathbb{Z}^{\boldsymbol{D}}$ consists of a set of connected cells $\boldsymbol{i} \subset \Gamma$. Internal boundary conditions are resolved by cutting some of the cells with a level-set function $s(\boldsymbol{x}) = 0$ which describes the boundary interface. Inside each cut-cell, the boundary is linearized so that it intersects through the cell as a line in 2D, and a plane in 3D. Control volume faces that are cut by this surface are referred to as cut-faces, and the boundary itself is referred to as an embedded boundary (EB). More complex boundaries are built through constructive solid geometry (CSG) by using boolean operations such as unions, intersections, sums etc. Patch-based adaptive Mesh Refinement (AMR) is used by allowing recursively refined overlapping grids. We work with a set of grids Γ^l that are nested hierarchically from the coarsest grid to the finest grid; the refinement factor between each level is always 2. We always require that a grid Γ^l covers the overlapping region of a coarser grid Γ^{l-1} and a finer grid Γ^{l+1}. Figure 1 shows the grids in greater detail.

Figure 2 shows the the cut cells in greater details. Here, a cell is identified by it's index $\boldsymbol{i}$. We will take $\boldsymbol{x}_{\boldsymbol{i}}$ to be the *cell center*, and $\overline{\boldsymbol{x}}_{\boldsymbol{i}}$ to the be *cell centroid*. The volume fraction is $0 \leq \kappa_{\boldsymbol{i}} \leq 1$: Cells with $\kappa_{\boldsymbol{i}} = 0$ are denoted as *covered* cells and lie completely outside the computational domain; cells with volume fractions $0 < \kappa_{\boldsymbol{i}} < 1$ are called *irregular* cells and lie partially inside the computational domain (see e.g. the middle cell in Fig. 2). Furthermore, cell faces are denoted by $f_d^{\pm}(\boldsymbol{i})$ where $\pm$ indicates the high $(+)$ or low $(-)$ face of the cell $\boldsymbol{i}$ in the coordinate direction d. A face center for face $f_d^{\pm}(\boldsymbol{i})$ is therefore located at $\boldsymbol{x}_{\boldsymbol{i}\pm\frac{1}{2}\mathbf{e}_d}$ where $\mathbf{e}_d$ is a unit vector in the d-direction. Face centroids are likewise denoted $\overline{\boldsymbol{x}}_{\boldsymbol{i}\pm\frac{1}{2}\mathbf{e}_d}$ and the corresponding area fractions are denoted by $\alpha_{\boldsymbol{i}\pm\frac{1}{2}\boldsymbol{e}_d}$. Finally, the EB centroid is denoted by $\overline{\boldsymbol{x}}_{\boldsymbol{i}}^{\mathrm{EB}}$ with a corresponding area fraction $\alpha_{\boldsymbol{i}}^{\mathrm{EB}}$ and an outward normal vector $\boldsymbol{n}_{\boldsymbol{i}}$.

To derive the above quantities, let $V_{\boldsymbol{i}}$ denote the part of the Cartesian cell inside the computational domain, $A_{\boldsymbol{i}}^{\pm,d}$ be the un-cut part of a face $f_d^{\pm}(\boldsymbol{i})$, and $A_{\boldsymbol{i}}^{\mathrm{EB}}$ be the part of the EB that intersects with the

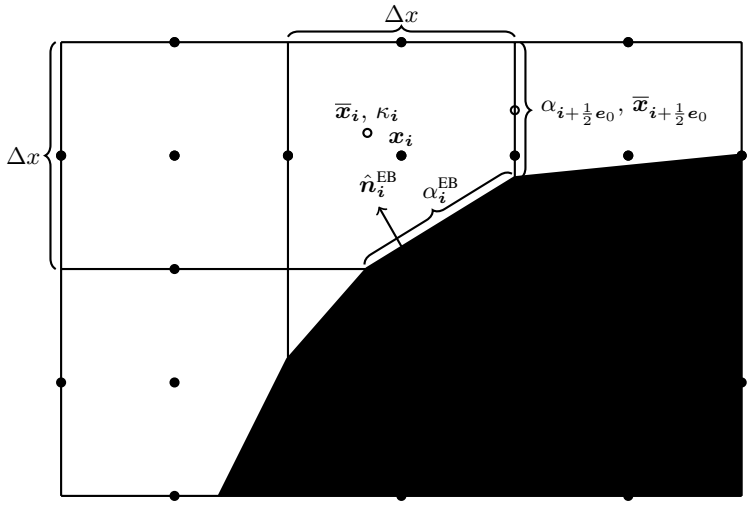

Figure 2: Cut-cell discretization.

cell. The above quantities are given by

$$\overline{\boldsymbol{x}}_{\boldsymbol{i}} = \frac{1}{\Delta x^{\boldsymbol{D}}} \int_{V_{\boldsymbol{i}}} \boldsymbol{x} \mathrm{d}V_{\boldsymbol{i}}, \tag{14a}$$

$$\overline{\boldsymbol{x}}_{\boldsymbol{i}\pm\frac{1}{2}\boldsymbol{e}_d} = \frac{1}{\Delta x^{\boldsymbol{D}-1}} \int_{A_{\boldsymbol{i}}^{\pm,d}} \boldsymbol{x} \mathrm{d}A_{\boldsymbol{i}}^{\pm,d}, \tag{14b}$$

$$\overline{\boldsymbol{x}}_{\boldsymbol{i}}^{\mathrm{EB}} = \frac{1}{\Delta x^{\boldsymbol{D}-1}} \int_{A_{\boldsymbol{i}}^{\mathrm{EB}}} \boldsymbol{x} \mathrm{d}A_{\boldsymbol{i}}^{\mathrm{EB}}, \tag{14c}$$

$$\kappa_{\boldsymbol{i}} = \frac{1}{\Delta x^{\boldsymbol{D}}} \int_{V_{\boldsymbol{i}}} \mathrm{d}V_{\boldsymbol{i}}, \tag{14d}$$

$$\alpha_{\boldsymbol{i}\pm\frac{1}{2}\boldsymbol{e}_d} = \frac{1}{\Delta x^{\boldsymbol{D}-1}} \int_{A_{\boldsymbol{i}}^{\pm,d}} \boldsymbol{x} \mathrm{d}A_{\boldsymbol{i}}^{\pm,d}, \tag{14e}$$

$$\alpha_{\boldsymbol{i}}^{\mathrm{EB}} = \frac{1}{\Delta x^{\boldsymbol{D}-1}} \int_{A_{\boldsymbol{i}}^{\mathrm{EB}}} \mathrm{d}A_{\boldsymbol{i}}^{\mathrm{EB}}, \tag{14f}$$

$$\boldsymbol{n}_{\boldsymbol{i}} = \frac{1}{\Delta x^{\boldsymbol{D}-1}} \int_{A_{\boldsymbol{i}}^{\mathrm{EB}}} \boldsymbol{n} \mathrm{d}A_{\boldsymbol{i}}^{\mathrm{EB}}. \tag{14g}$$

For brevity's sake, we will from now on omit the dependency on $\boldsymbol{i}$, and re-specify this when necessary.

3.2. Poisson & Eddington equations

We use the finite volume method for the Poisson equation and Eddington equations. For brevity, we restrict our discussion to the Poisson equation; the Eddington equations contain additional diagonal terms which can be added into the analysis below. Writing the Poisson equation in integral form yields

$$\oint_A \epsilon \nabla\phi \cdot \mathrm{d}\boldsymbol{A} = -\frac{1}{\epsilon_0} \int_V \rho \mathrm{d}V \tag{15}$$

As an example, we consider the cell shown in Fig. 3. Here, the volume $V_{\boldsymbol{i}}$ is a cut-cell at a domain boundary. Integration of Poisson's equation yields

$$\oint_A \epsilon \nabla\phi \cdot \mathrm{d}\boldsymbol{A} = F_1 + F_2 + F_3 + F_{\mathrm{D}} + F_{\mathrm{EB}}, \tag{16}$$

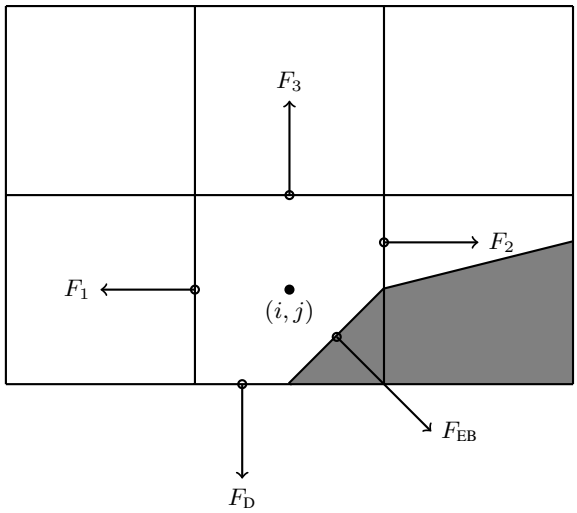

Figure 3: Cut cell at domain face.

where the fluxes are centroid-centered on their respective faces. We use second order differences to discretize the face-centered fluxes. For example, for the flux through the top face in Fig. 3 we find

$$F_3 = F_{i,j+1/2} = \epsilon_{i,j+1/2} \frac{\phi_{i,j+1} - \phi_{i,j}}{\Delta x}. \tag{17}$$

For fluxes through face centroids we interpolate face-centered to their centroids. For example, the flux F_2 in Fig. 3 is given by

$$F_2 = \alpha_{i+1/2,j} \left[F_{i+1/2,j}(1-s) + s F_{i+1/2,j+1} \right], \tag{18}$$

where s is the normalized distance from the face center to the face centroid, and $F_{i+1/2,j}$ and $F_{i+1/2,j+1}$ are face-centered fluxes.

3.2.1. Boundary conditions

For Neumann boundary conditions the domain and embedded boundary fluxes are specified directly, whereas Dirichlet boundary conditions are more involved. For Dirichlet conditions on domain faces we apply finite differences in order to evaluate the flux. For example, for a constant Dirichlet boundary condition $\phi = \phi_0$, then the face-centered flux at the bottom face is, to first order,

$$F_{i,j-1/2} = -\epsilon_{i,j-1/2} \frac{\phi_{i,j} - \phi_0}{\Delta x}. \tag{19}$$

Higher order boundary conditions are built by including more interior cells. Typically, we use second order boundary conditions in which

$$F_{i,j-1/2} = -\frac{\epsilon_{i,j-1/2}}{\Delta x} \left(3\phi_{i,j+1} - \frac{1}{3}\phi_{i,j} - \frac{8}{3}\phi_0 \right) \tag{20}$$

As with the flux F_2 on the interior face, fluxes on domain faces are also interpolated to face centroids. Thus, F_{D} becomes

$$F_{\text{D}} = \alpha_{j-1/2} \left[F_{i,j-1/2}(1-t) + t F_{i-1,j-1/2} \right], \tag{21}$$

where t is the distance from the face center to the face centroid.

The evaluation of Dirichlet boundary conditions on the EB is more complicated because the EB normal does not, in general, align with any of the coordinate directions. Various approaches for evaluating this term

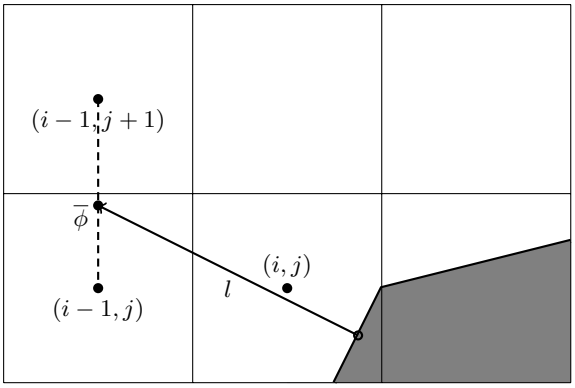

Figure 4: Ray casting at the EB for obtaining the normal gradient.

exist. Typically, we use a combination of two approaches. The first approach is a ray-casting method. Here, we cast a ray along the EB normal from the EB centroid into the domain and compute the intersection points between this ray and a line (plane in 3D) that connects two or more cell centers, see Fig. 4. At these points we interpolate the value of ϕ and then compute a finite difference approximation to the normal gradient. For example, to first order, the flux in 4 is

$$F_{\mathrm{EB}} = \alpha_{\mathrm{EB}} \epsilon_{\mathrm{EB}} \frac{\phi_{\mathrm{EB}} - \overline{\phi}}{l}, \tag{22}$$

where $\overline{\phi}$ is the interpolated value at the intersection of the ray and the line that connects $\boldsymbol{x}_{i-1,j}$ and $\boldsymbol{x}_{i-1,j+1}$. Higher-order approximations to the flux can be built in a similar way by including more interior cells. Typically, we use second order boundary conditions. If we cannot find a stencil for computing the normal derivative this way, which can occur if there aren't enough cells available, we use least squares for computing the gradient. Before discussing least squares reconstruction, we mention that the Eddington equations require a boundary condition of the Robin type. Equation (9) shows that the photon boundary flux is proportional to the solution at the boundary. Since this value is not known, we extrapolate it from the interior. On domain faces we simply use a first or second order finite difference approximation in order to extrapolate the solution from the interior to the domain face. Embedded boundaries require a more sophisticated approach which we discuss below. Typically, we use a second order Taylor-based finite difference stencil, which usually yields a comparatively compact stencil. If that fails we reconstruct the solution at the boundary by using weighted least squares, as discussed below.

Least squares data reconstruction. Our least squares methods are based on the minimization of a multidimensional Taylor series

$$\phi_{\boldsymbol{i}} = \sum_{|q| \le Q} \frac{1}{\boldsymbol{q}!} \left(\boldsymbol{x}_{\boldsymbol{i}} - \boldsymbol{x}_0\right)^{\boldsymbol{q}} \phi^{\boldsymbol{q}}(\boldsymbol{x}_0) + \mathcal{O}_{\boldsymbol{i}}\left(h^{Q+1}\right) \tag{23}$$

where h is the largest displacement $|\boldsymbol{x}_{\boldsymbol{i}} - \boldsymbol{x}_0|$ and $\boldsymbol{q}$ is a multi-index:

$$\boldsymbol{q}! = q_1! q_2! q_3! \tag{24a}$$

$$\left(\boldsymbol{x}_{\boldsymbol{i}} - \boldsymbol{x}_0\right)^{\boldsymbol{q}} = \left(x_1 - x_0\right)^{q_1} \left(x_2 - x_0\right)^{q_2} \left(x_3 - x_0\right)^{q_3}, \tag{24b}$$

$$\phi^{\boldsymbol{q}} = \phi^{\boldsymbol{q}}(\boldsymbol{x}_0) = \frac{\partial^{q_1}}{\partial x_1^{q_1}} \frac{\partial^{q_2}}{\partial x_2^{q_2}} \frac{\partial^{q_3}}{\partial x_3^{q_3}} \phi\Big|_{\boldsymbol{x}_0}. \tag{24c}$$

In the above, $\boldsymbol{x}_{\boldsymbol{i}}$ and $\phi_{\boldsymbol{i}}$ are known values, $\boldsymbol{x}_0$ is the point that we extrapolate to and $\phi^{\boldsymbol{q}}$ for $|\boldsymbol{q}| < Q$ are the unknown states at this point. We define a column vector of the unknown states $\Phi^{(Q)} = (\{\phi^{\boldsymbol{q}}\})^{\intercal}$ and

solve a system of equations by minimizing the residual

$$
\begin{aligned}
f\left(\Phi^{\boldsymbol{Q}}\right) &= \sum_{\boldsymbol{i}\in\mathcal{N}} w_{\boldsymbol{i}} \mathcal{O}_{\boldsymbol{i}}^2 \left(h^{Q+1}\right) \\
&= \sum_{\boldsymbol{i}\in\mathcal{N}} w_{\boldsymbol{i}} \left[\phi_{\boldsymbol{i}} - \sum_{|q|\leq Q} \frac{1}{\boldsymbol{q}!} \left(\boldsymbol{x_i} - \boldsymbol{x}_0\right)^{\boldsymbol{q}} \phi^{\boldsymbol{q}}(\boldsymbol{x}_0)\right]^2,
\end{aligned}
\tag{25}
$$

where $w_{\boldsymbol{i}}$ are weights and $\mathcal{N}$ is some neighborhood of cells (see Fig. 5). Typically, we weight with the inverse distance $w_{\boldsymbol{i}} = |\boldsymbol{x_i} - \boldsymbol{x}_0|^{-1}$. We then minimize with respect to all unknown states, i.e. $\frac{\partial f}{\partial \phi^{\boldsymbol{p}}} = 0$, which yields $|\boldsymbol{p}| \leq Q$ equations in the form

$$
\sum_{\boldsymbol{i}\in\mathcal{N}} w_{\boldsymbol{i}} \frac{1}{\boldsymbol{p}!} \left(\boldsymbol{x_i} - \boldsymbol{x}_0\right)^{\boldsymbol{p}} \left[\phi_{\boldsymbol{i}} - \sum_{|q|\leq Q} \frac{1}{\boldsymbol{q}!} \left(\boldsymbol{x_i} - \boldsymbol{x}_0\right)^{\boldsymbol{q}} \phi^{\boldsymbol{q}}(\boldsymbol{x}_0)\right] = 0, \quad |\boldsymbol{p}| \leq Q. \tag{26}
$$

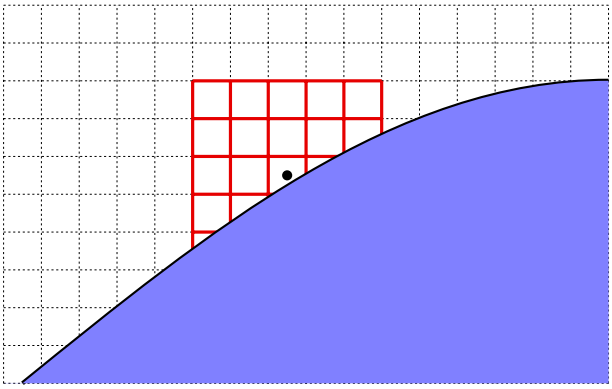

Figure 5: Example neighborhood for least squares data reconstruction.

The above set of equations yields a linear system $\mathbb{A}\Phi^{(Q)} = \mathbb{B}\Phi^{(\mathcal{N})}$ where $\Phi^{(\mathcal{N})}$ is a column vector of the known values $\phi_{\boldsymbol{i}}$. In practice, we construct $\Phi^{(Q)}$, $\mathbb{A}$, and the rows of $\mathbb{B}$ in lexicographical order. For example, in two dimensions for Q = 2, we find

$$
\Phi^{Q=2} = \begin{pmatrix} \phi_0 \\ \partial_x \phi_0 \\ \partial_x^2 \phi_0 \\ \partial_y \phi_0 \\ \partial_{xy} \phi_0 \\ \partial_y^2 \phi_0 \end{pmatrix}. \tag{27}
$$

The system of equations is solved by

$$
\Phi^{(Q)} = \left(\mathbb{A}^+ \mathbb{B}\right) \Phi^{(\mathcal{N})}, \tag{28}
$$

where $\mathbb{A}^+$ is the Moore-Penrose pseudoinverse. We compute this by first obtaining the singular value decomposition of $\mathbb{A} = \mathbb{V}\Sigma^+\mathbb{U}^*$, which directly yields $\mathbb{A}^+ = \mathbb{U}\Sigma\mathbb{V}^*$.

Once $\mathbb{C} = \mathbb{A}^+\mathbb{B}$ is computed, we can obtain stencils for the entire unknown state $\Phi^{(Q)}$. The stencil for obtaining the interpolated value $\phi(\boldsymbol{x}_0)$ is found in the first row of $\mathbb{C}$. Likewise, for $Q = 2$ in two dimensions, the stencil for the gradient is found in the second and fourth rows, respectively (see Eq. (27)).

3.2.2. Multigrid solver

The discretized Poisson equation is solved by embedding it in a geometric multigrid solver in residual-correction form. Multigrid involves iterative relaxation on progressively coarsened grids and is also compatible with AMR. On the coarsest level, the discretized system is solved directly with a biconjugate gradient stabilized method. Typically, better convergence for each cycle is achieved the further one coarsens; for domains without embedded boundaries one may coarsen up to a level where the domain only consists of two cells in one of the coordinate directions. However, for embedded boundary applications the multigrid convergence rate usually deteriorates when one coarsens too far. This occurs because, past a certain point, the coarse grid operator no longer well represents the longer wavelength modes of the fine grid operator (because the EB stencils have changed). Over-coarsening may also lead to geometrical under-resolution. We try to avoid these issues by dropping to the bottom solver prematurely; we usually drop to the bottom solver after 1-4 multigrid coarsenings of the coarsest grid.

3.3. Convection-Diffusion-Reaction equations

For the mass transport equation we apply the method of Godunov with van Leer limiting to the equation:

$$\frac{\partial n}{\partial t} = -\nabla \cdot (\boldsymbol{v} n) + \nabla \cdot (D \nabla n) + S. \tag{29}$$

For the diffusive term $\nabla \cdot (D \nabla n)$ we use the discretization used for the Poisson and Eddigton equations. It is thus sufficient to discuss the construction of the advective term $\nabla \cdot (\boldsymbol{v} n)$.

3.3.1. Advective discretization

3.3.2. Hybrid divergence

A problem with cut-cell grids is that the CFL condition on the time step is at best

$$\Delta t = \mathcal{O}\left(\frac{\Delta x}{\boldsymbol{v}_{\boldsymbol{i}}^{\max}} \kappa_{\boldsymbol{i}}^{\frac{1}{D}} \right), \tag{30}$$

which is the well-known small-cell problem with embedded boundary grids. The approach that we take to this problem is to stabilize the advective discretization by computing a hybrid discretization which expands the range of influence of small control volumes. Firstly, we compute the non-conservative divergence (i.e. which does not include the effect of the embedded boundary)

$$\mathrm{Div}^{\mathrm{NC}}(F) = \sum \tag{31}$$

Note that the computation of the non-conservative divergence requires second order solutions covered faces. The full algorithm for extrapolation can be found in [van Straalen, Graves: an adaptive cartesian grid embedded...]

Secondly, we compute the conservative divergence

$$\mathrm{Div}^{\mathrm{NC}}(F) = \sum \tag{32}$$

We then use a hybridization of these two estimates in order to obtain a stable method:

$$\nabla \cdot (\boldsymbol{u} n) = \kappa_{\boldsymbol{i}} \mathrm{Div}^{\mathrm{C}}(F) + (1 - \kappa_{\boldsymbol{i}}) \, \mathrm{Div}^{\mathrm{NC}}(F). \tag{33}$$

The above method fails to conserve by the difference between the hybrid and conservative divergence,

$$\delta M_{\boldsymbol{i}} = \kappa_{\boldsymbol{i}} \, (1 - \kappa_{\boldsymbol{i}}) \left[\mathrm{Div}^{\mathrm{C}}(F) - \mathrm{Div}^{\mathrm{NC}}(F) \right]. \tag{34}$$

In order to maintain global conservation, this mass difference is redistributed into nearby cells,

$$\nabla \cdot (\boldsymbol{u} n) \rightarrow \nabla \cdot (\boldsymbol{u} n) + w_{\boldsymbol{i},\boldsymbol{i}'} \delta M_{\boldsymbol{i}'}, \tag{35}$$

where $w_{\boldsymbol{i}} > 0$ are weights which fulfill

$$\sum_{\boldsymbol{i}' \in N(\boldsymbol{i})} w_{\boldsymbol{i},\boldsymbol{i}'} \kappa_{\boldsymbol{i}'} \tag{36}$$

where $N(\boldsymbol{i})$ is a set of indices in the neighborhood of $\boldsymbol{i}$. We use volume weighted redistribution

$$w_{\boldsymbol{i},\boldsymbol{i}'} = \frac{1}{\sum_{\boldsymbol{i}' \in N(\boldsymbol{i})} \kappa_{\boldsymbol{i}'}}. \tag{37}$$

3.3.3. Refluxing and redistribution

If the embedded boundary does not cross a coarse-fine interface, straightforward refluxing at the coarse-fine interface is sufficient for closing the advective discretization. We do this by precomputing a set of flux registers that hold the face centered fluxes on both sides of the coarse-fine interface. Refluxing is then just a matter of subtracting the coarse flux from the divergence computation, and adding in the sum of the fine face fluxes. I.e. let $\{f_{\mathrm{f}}(f_{\mathrm{c}})\}$ be the set of fine faces that are obtained when coarsening of a coarse face f_{c}. In the reflux step, the divergence in the coarse cell is then modified as

$$\nabla \cdot \boldsymbol{F} \rightarrow \nabla \cdot \boldsymbol{F} + \frac{1}{\Delta x} \left(\sum_f F_f - F_c \right), \tag{38}$$

where F_c and F_f are the coarse and fine-face fluxes, and the sum runs over all the fine faces.

If coarse-fine interfaces are allowed to cross the embedded boundary, the reflux-redistribution process is more complex since it must also account for mass moving between fine and coarse level. Thisf also involves "re-redistribution" where we correct the mass redistribution that was redistributed to the coarse grid from a fine grid.

4. Temporal discretization

We have implemented various explicit and semi-implicit discretizations for advancing the equations in time. We do not have an implicit solver, although work in that direction is ongoing.

4.1. Explicit 2nd order Runge-Kutta

As a first step, we have chosen to integrate our equations of motion by using explicit, second order Runge-Kutta methods. For equations of the type

$$\frac{\partial f}{\partial t} = g, \tag{39}$$

the second order Runge-Kutta method between time steps $k\Delta t$ and $(k+1)\Delta t$ is given by:

$$f^{k+1} = f^k + \Delta t \left[\left(1 - \frac{1}{2\alpha} \right) g^k + \frac{1}{2\alpha} g^{k+\alpha} \right]. \tag{40}$$

Here, $f^{k+\alpha}$ denotes the state at the intermediate time step $(k+\alpha)\Delta t$ where α denotes the Runge-Kutta method. For example, the midpoint and Heun's method (which is TVD) are given by $\alpha = 1/2$ and $\alpha = 1$, respectively. Note that the two-stage Runge-Kutta method requires two right-hand side evaluations for each time step, making it necessary to solve the Poisson and Eddington equations twice during a single time step.

Applying the above time stepping method to the minimal plasma model yields

$$n^{k+\alpha} = n^k + \alpha \Delta t G^k, \tag{41a}$$

$$\sigma^{k+\alpha} = \sigma^k + \alpha \Delta t F^k, \tag{41b}$$

where $G = S - \nabla \cdot (\boldsymbol{v}n) + \nabla \cdot (D\nabla n)$. These values then used to obtain new solutions for the Poisson and Eddington equations, which yields intermediate photon densities and fields $E^{k+\alpha}$, $\boldsymbol{E}^{k+\alpha}$ which are then used to advance the remaining fraction of the time step:

$$n^{k+1} = n^k + \Delta t \left[\left(1 - \frac{1}{2\alpha} \right) G^k + \frac{1}{2\alpha} G^{k+\alpha} \right], \tag{42a}$$

$$\sigma^{k+1} = \sigma^k + \Delta t \left[\left(1 - \frac{1}{2\alpha} \right) F^k + \frac{1}{2\alpha} G^{k+\alpha} \right]. \tag{42b}$$

After the new densities and surface charges have been obtained, we re-evaluate the Poisson and Eddington equations at the end of the time step which yields the new values E^{k+1}, $\boldsymbol{E}^{k+1}$.

4.2. 4th order Runge-Kutta

The classical fourth order Runge-Kutta method for (39) is

$$k_1 = \Delta t G \left(t^k, n^k \right), \tag{43a}$$

$$k_2 = \Delta t G \left(t^k + \frac{\Delta t}{2}, n^k + \frac{k_1}{2} \right), \tag{43b}$$

$$k_3 = \Delta t G \left(t^k + \frac{\Delta t}{2}, n^k + \frac{k_2}{2} \right), \tag{43c}$$

$$k_4 = \Delta t G \left(t^k + \Delta t, n^k + k_3 \right), \tag{43d}$$

$$n^{k+1} = n^k + \frac{1}{6} \left(k_1 + 2k_2 + 2k_3 + k_4 \right), \tag{43e}$$

and an equivalent equation is found for advancing $\sigma^k \to \sigma^{k+1}$. After evaluating each k, it is necessary to solve the Poisson and Eddington equations before evaluating the next one. For example, evaluation of k_2 requires the solution of the Poisson equation with the space charge density computed as $\rho = \sum q_n \left(n^k + \frac{k_1}{2} \right)$.

4.3. Semi-implicit TGA scheme

We have also implemented a semi-implicit time stepping method. Our aim is to advance semi-implicitly by computing $\boldsymbol{v}$, D, and S at half time steps, and then apply either a time-centered scheme (e.g. Crank-Nicholson). First, we decoupled the Poisson equation by solving it semi-implicitly at $t^{k+1/2}$. We do this by estimating variables explicitly at the half time step:

$$\widetilde{n}^{k+1/2} = n^k + \Delta t G^k \tag{44a}$$

$$\widetilde{\sigma}^{k+1/2} = \sigma^k + \Delta t F^k. \tag{44b}$$

These values are then used to compute the Poisson and Eddington equations at the half time, i.e.

$$\nabla \left(\epsilon \nabla \widetilde{\phi}^{k+1/2} \right) = -\frac{\rho^{k+1/2}}{\epsilon_0}, \tag{45}$$

where $\rho^{k+1/2} = \rho \left(\widetilde{n}^{k+1/2}, \widetilde{\sigma}^{k+1/2} \right)$. This essentially semi-implicitly decouples the elliptic equations from the hyperbolic ones, yielding

$$\partial_t n = -\nabla \cdot \left(\widetilde{\boldsymbol{v}}^{k+1/2} n \right) + \nabla \cdot \left(\widetilde{D}^{k+1/2} n \right) + \widetilde{S}^{k+1/2}, \tag{46a}$$

$$\partial_t \sigma = F \left(\widetilde{\boldsymbol{E}}^{k+1/2}, \widetilde{n}^{k+1/2}, \widetilde{E}_\nu^{k+1/2} \right). \tag{46b}$$

4.4. Time step restrictions

Multiple time step constraints are put in place in our code in order to ensure numerical stability, and accurate temporal evolution of the plasma as a whole. Firstly, the time step is restricted according to the CFL condition

$$\Delta t_{\mathrm{CFL}} = \alpha_{\mathrm{CFL}} \mathrm{Min}\left(\mathrm{Min}_d\left(\frac{\Delta x}{u_d}\right), \frac{\Delta x^2}{2\boldsymbol{D}_{\max}}\right), \tag{47}$$

where α_{CFL} is the CFL number which we use for fine-tuning the time step. For explicit integration $\alpha_{\mathrm{CFL}} < 1$, but this is not a contraint for our implicit integrator.

In addition to the above two requirements, the electric field should evolve slowly across a single time step. We estimate a largest possible time step by applying explicit Euler integration to Ampere's law:

$$\epsilon_0 \frac{\partial \boldsymbol{E}}{\partial t} = -\boldsymbol{J}, \tag{48}$$

We then discretize $\partial_t E_d = (E_d^{k+1} - E_d^k)/\Delta t$ which yields the dielectric relaxation time

$$\Delta t_\epsilon = \alpha_\epsilon \mathrm{Min}_d\left(\frac{\epsilon_0 E_d}{J_d}\right), \tag{49}$$

where the Min_d operator runs over all spatial directions d. Again, α_ϵ is a fudge-factor used for controlling the time step.

Finally, we restrict the time step by

$$\Delta t = \mathrm{Min}\left(\Delta t_{\mathrm{CFL}}, \Delta t_\epsilon\right). \tag{50}$$

Note that we never "backtrack" in a simulation; if the time step yields a fractional increase which is larger than e.g. α_ϵ (which can happen due to the use of multistep methods), we simply accept the result rather than backtracking to the previous time step.

5. Numerical examples

In this section we provide a few numerical examples that demonstrate the use of cut-cell grids for plasma simulations. For simplicity, we use well-established plasma kinetic models and restrict our discussions to comparatively simple geometries.

5.1. Simplified bulk plasma kinetics

We use the Morrow-Lowke plasma model for air and flue gases. The conservation equations that we solve are

$$\frac{\partial n_e}{\partial t} + \nabla\cdot(\boldsymbol{u}_e n_e) - \nabla\cdot(D_e \nabla n_e) = S_e, \tag{51a}$$

$$\frac{\partial n_+}{\partial t} + \nabla\cdot(\boldsymbol{u}_+ n_+) - = S_+, \tag{51b}$$

$$\frac{\partial n_-}{\partial t} + \nabla\cdot(\boldsymbol{u}_+ n_-) - = S_-, \tag{51c}$$

$$\tag{51d}$$

where n_e is the electron density and $n_\pm$ are the positive and negative ion densities. The source terms are

$$S_e = \alpha n_e \left|\boldsymbol{u}_e\right| - \eta n_e \left|\boldsymbol{u}_e\right| - \beta_{\mathrm{e},+} n_e n_+ + S_{\mathrm{ph}}, \tag{52a}$$

$$S_+ = \alpha n_e \left|\boldsymbol{u}_e\right| - \beta_{\mathrm{e},+} n_e n_+ - \beta_{+,\text{-}} n_+ n_- + S_{\mathrm{ph}}, \tag{52b}$$

$$S_- = \eta n_e \left|\boldsymbol{u}_e\right| - \beta_{+,\text{-}} n_+ n_-, \tag{52c}$$

where α is the Townsend ionization coefficient, η the Townsend attachment coefficient, $\beta_{\mu,\mu'}$ the recombination coeffcient of two species μ, μ'. Finally, S_{ph} is the photoionization source term discussed below. Several of the quantities above are field-dependent; for example, the electron velocity is given by $\boldsymbol{u}_e = -\mu_e \boldsymbol{E}$ where μ_e is the electron mobility. The ionization and attachment coefficients α and η are also field dependent, and so is the electron diffusion coefficient. Expressions for these may be found in Morrow-Lowke[citation neeeded].

In addition to the above reactions, we consider a three-group Eddigton SP1 photoionization model. That is, for $\gamma = 1, 2, 3$ we solve

$$\nabla \cdot \left(\frac{1}{3\kappa_\gamma} \nabla E_\gamma \right) - \kappa_\gamma E_\gamma = \frac{\eta_\gamma}{c}. \tag{53}$$

The corresponding absorption lengths are $\kappa_\gamma = \lambda_\gamma p_{O_2}$ where $\lambda_1 = 4.15 \times 10^{-2}\,\text{mPa}^{-1}$, $\lambda_2 = 1.09 \times 10^{-1}\,\text{mPa}^{-1}$, and $\lambda_3 = 6.69 \times 10^{-1}\,\text{mPa}^{-1}$. We use the same source term for each equation:

$$\eta_\gamma = \frac{p_q}{p + p_q} \frac{\nu_e}{\nu_i} \alpha n_e \left| \boldsymbol{v}_e \right|, \tag{54}$$

where the factor $p_q/(p + p_q)$ accounts for collisional quenching of an excited species of molecular nitrogen, and ν_e/ν_i is the ratio between the electron impact excitation rate of N_2 and the electron impact ionization rate of N_2. Following Bourdon et. al we take $\nu_e/\nu_i = 0.6$.

The photoionization source term is given by a weighted sum of the solution for E_γ:

$$S_{\text{ph}} = \sum_{\gamma=1}^{3} \xi_\gamma c p_{\text{O}_2} A_\gamma E_\gamma, \tag{55}$$

where ξ_γ is the photoionization efficiency, taken to be $\xi_\gamma = 0.1$ for simplicity. The coefficients A_γ are weighting coefficients that are adapted to fit with the original Zheleznyak integral model, and are given by $A_1 = 1.12 \times 10^{-4}\,\text{mPa}^{-2}$, $A_2 = 2.88 \times 10^{-3}\,(\text{mPa})^{-2}$, and $A_3 = 2.76 \times 10^{-1}\,(\text{m Pa})^{-2}$.

5.1.1. Dielectric boundary conditions

Electron emission from dielectric surfaces can occur through a number of different field-dependent process such as ion-bombardment, secondary electron impact, photoemission and field field-induced emission, to name a few. We consider a simple model for surface emission which consists of photoemission and ion bombardment. Photoemission is implemented by taken the influx of electrons from surface to be proportional to the outflux of photons into the dielectric. The Robin boundary condition for the Eddington equations has already been discussed [recall Eq. (11)], so the influx of electrons is

$$F_e = F_{e,\text{pe}} + F_{e,+}, \tag{56}$$

where $F_{e,\text{pe}}$ and $F_{e,+}$ are fluxes due to photoemission and positive ion bombardment, respectively.

The photoemission term is given by

$$F_e, \text{pe} = \begin{cases} -\gamma_{\text{pe}} \frac{c}{2} \sum_{\gamma=1}^{3} E_\gamma, & \boldsymbol{E} \cdot \boldsymbol{n} < 0, \\ 0, & \boldsymbol{E} \cdot \boldsymbol{n} \geq 0, \end{cases} \tag{57}$$

where γ_{pe} is the photoemission coefficient. The conditional clause is included because photo-emission is only possible in an external field that drives electrons away from the surface.

In the same way, we model inflow of electrons by electron bombardment by

$$F_{e,+} = \begin{cases} -\gamma_+ F_+ & \boldsymbol{E} \cdot \boldsymbol{n} < 0, \\ 0, & \boldsymbol{E} \cdot \boldsymbol{n} \geq 0, \end{cases} \tag{58}$$

where F_+ is the positive ion outflux on the dielectric and γ_+ is the second Townsend coefficient.

5.2. Razor-blade gaps

First, we consider examples of

5.3. Triple-junction setups

Here, we consider the inception of a streamer on a wall protrusion and its connection to a dielectric surface. The geometry, initial mesh and field distribution is shown in Fig. **??**.

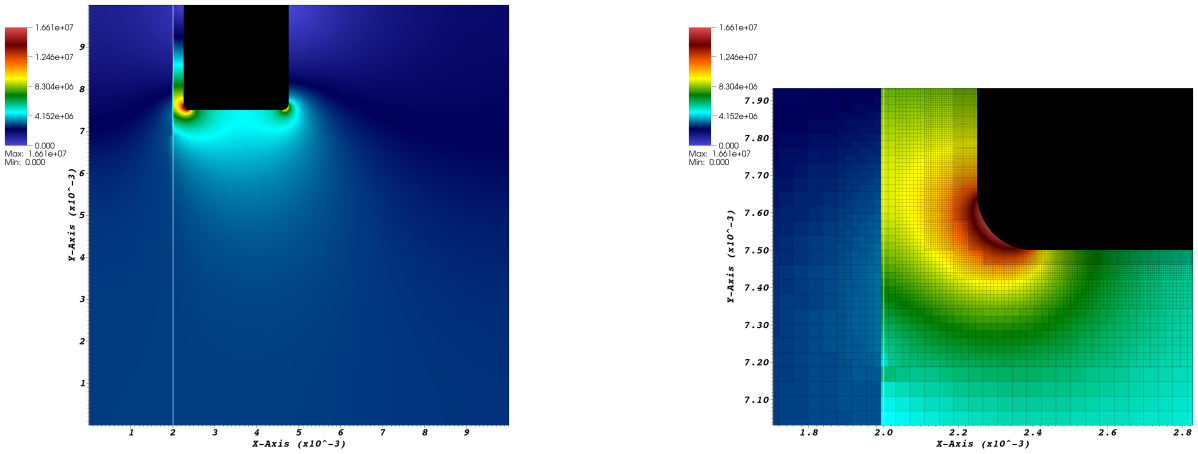

Figure 6: Geometry and initial field distribution for a "triple-junction" setup.

5.4. Streamer propagation from non-smooth electrodes

In this section we consider inception and propagation of a positive streamer from an needle-like electrode. The electrode is a cylinder with a rounded cap at the end. The length and radius of the cylinder are $L =$ and $R =$.

We incorporate surface roughness of the electrode by displacing the level-set electrode surface with a noise function. We use Ken Perlin's algorithm for creating landscape noise. The backbone of this algorithm is the creation of random gradients at various cell nodes across a mesh. This is mesh is not related to the AMR grids on which our equations are solved. For a given position $\boldsymbol{x}$ we find the control volume which contains this cell and then compute the dot product between the displacement vector to each cell node and the gradient at the those nodes. Interpolation of these four nodal values (8 in 3D) then yield a pseudorandom smooth noise inside the cell. Several improvements to this algorithm exist. For example, we use the improved Perlin noise model which uses a randomly permuted hash table for selecting gradient directions, and an improved interpolant that removes some artifacts on the noise landscape. Figure 7 shows typical examples of Perlin noise.

We use three octaves of Perlin noise folded in on itself. The spatial base frequency is $\boldsymbol{k} = 10^3\,(\hat{\boldsymbol{x}} + \hat{\boldsymbol{y}})$ and the base amplitude is $20\,\mu$m. For each octave that we add the spatial frequency is doubled and the amplitude is reduced by a factor of 2. For reproducibility, the gradient permutation vector is kept equal to Ken Perlin's original one (i.e. we do not reseed the noise functions). This function is then used to intersect the level-set function of a cylinder with a hemisphere at its tip.

The computation domain is a $4\,$cm $\times$ $4\,$cm and the needle protrudes $2\,$cm from the top edge of the domain (see Fig. 8. For Poisson's equation we use a constant-voltage Dirichlet boundary conditions on the needle and the top edge. A homogeneous Dirichlet boundary condition is applied on the bottom plate, and homogeneous Neumann boundary conditions are applied on the left and right domain edges.

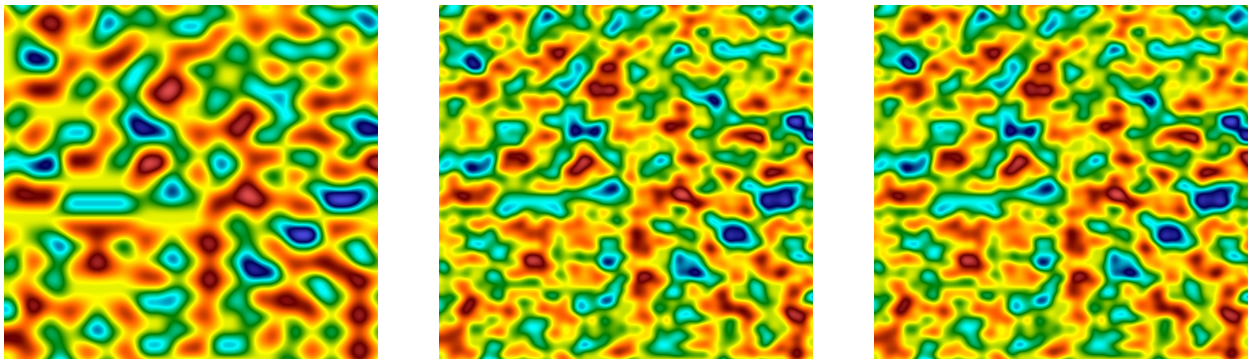

Figure 7: Perlin noise across a domain (-1, 1). The base frequency is $\boldsymbol{k} = 5\,(\hat{\boldsymbol{x}} + \hat{\boldsymbol{y}})$. Left: One octave. Middle: Two octaves. Right: Three octaves.

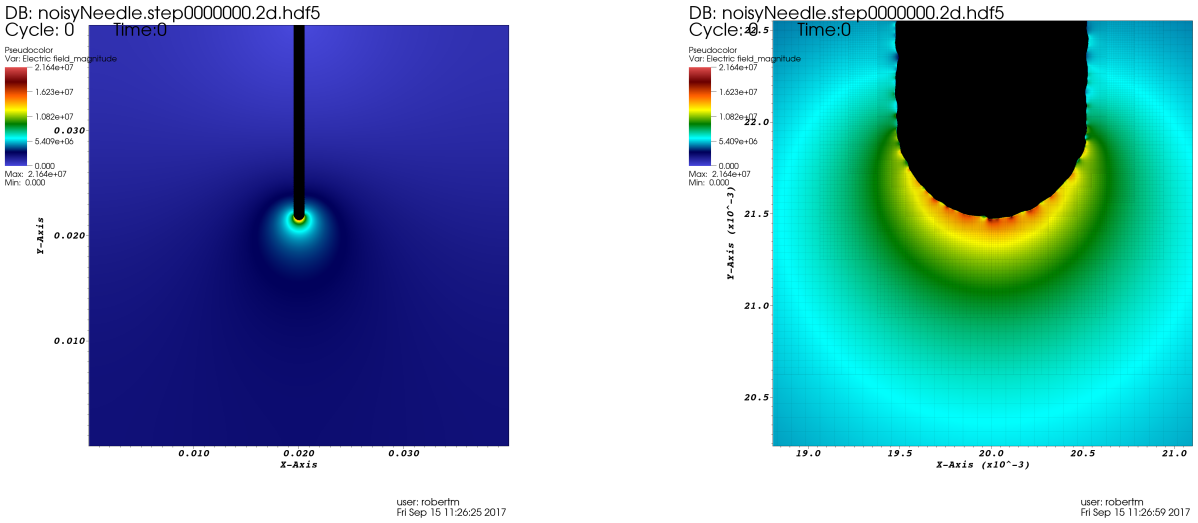

Figure 8: Cylindrical rod with surface roughness. Left: Electric field magnitude over the full domain.

For the charged species we apply, for simplicity, an outflow condition on the domain edges. On the needle surface we apply outflow boundary conditions for electrons and negative ions if $\boldsymbol{E} \cdot \boldsymbol{n} > 0$ (i.e. the needle is an anode) and a wall boundary condition (i.e. zero-flux) for positive ions. For the

6. Concluding remarks

Acknowledgements

This work was financially supported by the Norwegian Research Council, ABB Skien, and ABB CRC. The author expresses his gratitude to D. T. Graves for advice on the Chombo infrastructure, and S. Pancheshnyi for fruitful discussions regarding plasma kinetics.