Ionic Wind Simulation Notes

July 10, 2023

1 Objectives

The objective of this research is to replicate the results of Chen et al. (2017), "A Self-Consistent Model of Ionic Wind Generation by Negative Corona Discharges in Air With Experimental Validation".

2 Governing Equations

The following equations constitute equations 1-5 in Chen et al. They will be solved simultaneously in Exasim using the Convection-Diffusion transport model (Model D). The newly-introduced DAE "subproblem" module will be used to separately solve the species transport/diffusion problem and the electrostatic problem.

$$\frac{\partial n_e}{\partial t} + \nabla \cdot (-\mu_e \vec{E} n_e - D_e \nabla n_e) = \alpha n_e |\mu_e \vec{E}| - \eta n_e |\mu_e \vec{E}| - k_{ep} n_e n_p \tag{1}$$

$$\frac{\partial n_p}{\partial t} + \nabla \cdot (\mu_p \vec{E} n_p - D_p \nabla n_p) = \alpha n_e |\mu_e \vec{E}| - k_{np} n_n n_p - k_{ep} n_e n_p \tag{2}$$

$$\frac{\partial n_n}{\partial t} + \nabla \cdot (-\mu_n \vec{E} n_n - D_n \nabla n_n) = \eta n_e |\mu_e \vec{E}| - k_{np} n_n n_p \tag{3}$$

$$\nabla^2 \Phi = -\frac{e(n_p - n_e - n_n)}{\epsilon} \tag{4}$$

$$\vec{E} = -\nabla \Phi \tag{5}$$

Additionally, the one-way coupling equations from the EHD model to the gas dynamics model (N-S) is provided in Eqns. 6-8:

$$f_{ehd} = e(n_p - n_e - n_n)\vec{E} \tag{6}$$

$$\nabla \cdot \vec{u} = 0 \tag{7}$$

$$\rho_g \left(\frac{\partial \vec{u}}{\partial t} + \vec{u} \cdot \nabla \vec{u} \right) = -\nabla p + \mu_v \nabla^2 \vec{u} + f_{ehd} \tag{8}$$

3 Cylindrical Coordinates

The gradient and divergence operators in cylindrical coordinates (r, θ, z) are given, for a scalar function f and a vector field $\boldsymbol{f}$, by

$$\begin{aligned} \nabla f &= \frac{\partial f}{\partial r}\boldsymbol{e}_r + \frac{1}{r}\frac{\partial f}{\partial \theta}\boldsymbol{e}_\theta + \frac{\partial f}{\partial z}\boldsymbol{e}_z \\ \nabla \cdot \boldsymbol{f} &= \frac{1}{r}\frac{\partial (r f_r)}{\partial r} + \frac{1}{r}\frac{\partial f_\theta}{\partial \theta} + \frac{\partial f_z}{\partial z} \end{aligned}$$

If we assume axial symmetry all variable are function of (r, z) only.

4 Fully Conservative Form

Assuming axial symmetry, the above equations can be written as

$$\boldsymbol{M}\frac{\partial \mathbf{U}}{\partial t} + \frac{1}{r}\frac{\partial (r\mathbf{F}_r)}{\partial r} + \frac{\partial \mathbf{F}_z}{\partial z} - \mathbf{S} = \mathbf{0}$$

Were

$$\boldsymbol{U} = \begin{pmatrix} n_e \\ n_p \\ n_n \\ \Phi \end{pmatrix}, \quad \boldsymbol{F}_r = \begin{pmatrix} -\mu_e n_e E_r - D_e(q_e)_r \\ \mu_p n_p E_r - D_p(q_p)_r \\ -\mu_n n_n E_r - D_n(q_n)_r \\ -E_r \end{pmatrix}, \quad \boldsymbol{F}_z = \begin{pmatrix} -\mu_e n_e E_z - D_e(q_e)_z \\ -\mu_p n_p E_z - D_p(q_p)_z \\ -\mu_n n_n E_z - D_n(q_n)_z \\ -E_z \end{pmatrix}, \quad (9)$$

$$\boldsymbol{S} = \begin{pmatrix} \alpha n_e |\mu_e \vec{E}| - \eta n_e |\mu_e \vec{E}| - k_{ep} n_e n_p \\ \alpha n_e |\mu_e \vec{E}| - k_{np} n_n n_p - k_{ep} n_e n_p \\ \eta n_e |\mu_e \vec{E}| - k_{np} n_n n_p \\ -\frac{e(n_p - n_e - n_n)}{\epsilon} \end{pmatrix}, \quad \boldsymbol{M} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix} \quad (10)$$

and

$$\boldsymbol{Q} = \begin{pmatrix} (q_e)_r & (q_e)_z \\ (q_p)_r & (q_p)_z \\ (q_n)_r & (q_n)_z \\ -E_r & -E_z \end{pmatrix}, \quad (11)$$

with

$$\boldsymbol{Q} = \nabla \boldsymbol{U}. \quad (12)$$

Note that under the axisymmetry assumption, the gradient operators in cylindrical and cartesian coordinates are the same.

5 Weak form and the Divergence Theorem

The elemental volume becomes $dV = rdrdz$ and for any $\mathbf{W}$ we can write the following weighted residual form

$$\int_V \left(\boldsymbol{M}\frac{\partial \mathbf{U}}{\partial t} + \frac{1}{r}\frac{\partial (r\mathbf{F}_r)}{\partial r} + \frac{\partial \mathbf{F}_z}{\partial z} + \mathbf{S} \right) \mathbf{W}\, rdrdz = 0,$$

or, integrating by parts,

$$\int_V \left((\boldsymbol{M} r) \frac{\partial \mathbf{U}}{\partial t} + r\mathbf{S} \right) \mathbf{W} \, drdz + \int_S (r\mathbf{F}_r \boldsymbol{n}_r + r\mathbf{F}_z \boldsymbol{n}_z) \, \mathbf{W} \, dS$$

$$- \int_V \left(r\mathbf{F}_r \frac{\partial \mathbf{W}}{\partial r} + r\mathbf{F}_z \frac{\partial \mathbf{W}}{\partial z} \right) \, drdz = \mathbf{0}$$

Note that in the integrals in the second and third lines the *effective* dV becomes $drdz$. The only differences between cylindrical and cartesian coordinates thus

- Multiply $\boldsymbol{M}$ and $\boldsymbol{S}$ by r. Note that this will require a modified mass matrix.
- Multiply $\boldsymbol{F}_r$ and $\boldsymbol{F}_z$ by r

6 Nondimensionalization

Because of the wide range of scales used, it is important to nondimensionalize the problem to prevent numerical instability. The solution variables as well as spatial and temporal variables were nondimensionalized.

6.1 Nondimensional groups

The following nondimensional groups were chosen:

- $n_{ref} = \frac{\epsilon_0 E_{bd}}{e r_{tip}}$
- $E_{ref} = E_{bd}$
- $\Phi_{ref} = E_{bd} r_{tip}$
- $t_{ref} = \frac{r_{tip}}{\mu_{e,ref} E_{bd}}$
- $l_{ref} = r_{tip}$

Where $\mu_{e,ref}$ is taken at the reduced electric field value of $\frac{E_{bd}}{N}$, where N, the neutral number density, is computed using the ideal gas equation of state at standard temperature and pressure:

- $pV = N k_B T$
- $P = 101325 \; Pa$
- $V = 1 \; m^3$
- $T = 273.15 \; K$
- $k_B = 1.380649 \times 10^{-23} \; \frac{m^2 kg}{s^2 K}$

- $\implies N = 2.6868 \times 10^{25} \ \frac{particles}{m^3}$

Thus, the independent and dependent variables may be expressed as the following. The asterisk indicates a nondimensional quantity.

- $n_e = \frac{n_e^* \epsilon_0 E_{bd}}{e r_{tip}}$
- $n_p = \frac{n_p^* \epsilon_0 E_{bd}}{e r_{tip}}$
- $n_n = \frac{n_n^* \epsilon_0 E_{bd}}{e r_{tip}}$
- $\vec{E} = \vec{E^*} E_{bd}$
- $\Phi = \Phi^* E_{bd} r_{tip}$
- $t = \frac{t^* r_{tip}}{\mu_{e,ref} E_{bd}}$
- $r = r^* r_{tip}$
- $z = z^* r_{tip}$

Where r_{tip} is the needle tip radius of curvature, $220 \mu m$, and E_{bd} is the breakdown electric field strength in air, $3 \times 10^6 \frac{V}{m}$.

We can re-write the governing equations (section 2) using the nondimensional groups, taking care to nondimensionalize the partial derivatives with respect to space and time as well:

$$\frac{\partial (n_e^* n_{ref})}{\partial \left(\frac{t^* r_{tip}}{\mu_e E_{bd}} \right)} + \nabla \cdot \frac{\left(-\mu_e \vec{E^*} E_{bd} (n_e^* n_{ref}) - D_e \nabla \left(\frac{n_e^* n_{ref}}{r_{tip}} \right) \right)}{r_{tip}} = (\alpha - \eta)(n_e^* n_{ref}) |\mu_e (\vec{E^*} E_{bd})| - k_{ep} n_e^* n_p^* n_{ref}^2$$

$$\frac{\partial (n_p^* n_{ref})}{\partial \left(\frac{t^* r_{tip}}{\mu_e E_{bd}} \right)} + \nabla \cdot \frac{\left(\mu_p \vec{E^*} E_{bd} (n_p^* n_{ref}) - D_p \nabla \left(\frac{n_p^* n_{ref}}{r_{tip}} \right) \right)}{r_{tip}} = \alpha n_e^* n_{ref} |\mu_e (\vec{E^*} E_{bd})| - n_p^* n_{ref}^2 \left(k_{np} n_n^* + k_{ep} n_e^* \right)$$

$$\frac{\partial (n_n^* n_{ref})}{\partial \left(\frac{t^* r_{tip}}{\mu_e E_{bd}} \right)} + \nabla \cdot \frac{\left(-\mu_n \vec{E^*} E_{bd} (n_n^* n_{ref}) - D_n \nabla \left(\frac{n_n^* n_{ref}}{r_{tip}} \right) \right)}{r_{tip}} = \eta n_e^* n_{ref} |\mu_e (\vec{E^*} E_{bd})| - k_{np} n_n^* n_p^* n_{ref}^2$$

$$\nabla^2 \left(\frac{\Phi^* E_{bd} r_{tip}}{r_{tip}^2} \right) = -\frac{\epsilon_0 e E_{bd}}{\epsilon_0 e r_{tip}} (n_p^* - n_e^* - n_n^*)$$

Simplifying:

$$\frac{\partial n_e^*}{\partial t^*} + \nabla \cdot \left(-\frac{\mu_e}{\mu_{e,ref}} \vec{E^*} n_e^* - \frac{D_e}{\mu_{e,ref} E_{bd} r_{tip}} \nabla(n_e^*) \right) = (\alpha - \eta) \frac{\mu_e}{\mu_{e,ref}} r_{tip} n_e^* |\vec{E^*}| - \frac{k_{ep} \epsilon_0}{e \mu_{e,ref}} n_e^* n_p^*$$

$$\frac{\partial n_p^*}{\partial t^*} + \nabla \cdot \left(\frac{\mu_p}{\mu_{e,ref}} \vec{E^*} n_p^* - \frac{D_p}{\mu_{e,ref} E_{bd} r_{tip}} \nabla(n_p^*) \right) = \alpha \frac{\mu_e}{\mu_{e,ref}} r_{tip} n_e^* |\vec{E^*}| - \left(\frac{\epsilon_0 n_p^*}{e \mu_{e,ref}} \right) (k_{np} n_n^* + k_{ep} n_e^*)$$

$$\frac{\partial n_n^*}{\partial t^*} + \nabla \cdot \left(-\frac{\mu_n}{\mu_{e,ref}} \vec{E^*} n_n^* - \frac{D_n}{\mu_{e,ref} E_{bd} r_{tip}} \nabla(n_n^*) \right) = \eta \frac{\mu_e}{\mu_{e,ref}} r_{tip} n_e^* |\vec{E^*}| - \left(\frac{k_{np} \epsilon_0}{e \mu_{e,ref}} \right) n_n^* n_p^*$$

$$\nabla^2 \Phi^* = n_e^* + n_n^* - n_p^*$$

7 Boundary Conditions

Note: Boundary numbering follows the boundary numbering in the paper

7.1 Boundary 1

Equation	Boundary condition	Boundary condition type
1	Total flux $-\vec{n} \cdot \left(-\mu_{\mathrm{e}} \vec{E} - D_{\mathrm{e}} \nabla n_{\mathrm{e}} \right) = \gamma n_{\mathrm{p}} \lvert \mu_{\mathrm{p}} \vec{E} \rvert$	Neumann
2	Outflow, $\vec{n} \cdot (-D_{\mathrm{p}} \nabla n_{\mathrm{p}}) = 0$	Neumann
3	$n_n = 0$	Dirichlet
4	$\Phi = -U_a$	Dirichlet

Table 1: Boundary conditions for the emitter tip (Boundary surface 1)

7.2 Boundary 2

Equation	Boundary condition	Boundary condition type
1	Axial symmetry $\frac{\partial n_\mathrm{e}}{\partial r} = 0$	Neumann
2	Axial symmetry $\frac{\partial n_\mathrm{p}}{\partial r} = 0$	Neumann
3	Axial symmetry $\frac{\partial n_\mathrm{n}}{\partial r} = 0$	Neumann
4	Axial symmetry $\frac{\partial \phi}{\partial r} = 0$	Neumann

Table 2: Boundary conditions for (Boundary surface 2)

7.3 Boundary 3

Equation	Boundary condition	Boundary condition type
1	Open boundary $\begin{array}{r} \vec{n} \cdot (-D_\mathrm{e} \nabla n_\mathrm{e}) = 0; \vec{n} \cdot \left(-\mu_\mathrm{e} \vec{E}\right) \geqslant 0 \\ n_\mathrm{e} = 0; \vec{n} \cdot \left(-\mu_\mathrm{e} \vec{E}\right) < 0 \end{array}$	Neumann/Dirichlet
2	Open boundary $\begin{array}{r} \vec{n} \cdot (-D_\mathrm{p} \nabla n_\mathrm{p}) = 0; \vec{n} \cdot \left(-\mu_\mathrm{p} \vec{E}\right) \geqslant 0 \\ n_\mathrm{p} = 0; \vec{n} \cdot \left(-\mu_\mathrm{p} \vec{E}\right) < 0 \end{array}$	Neumann/Dirichlet
3	Open boundary $\begin{array}{r} \vec{n} \cdot (-D_\mathrm{n} \nabla n_\mathrm{n}) = 0; \vec{n} \cdot \left(-\mu_\mathrm{n} \vec{E}\right) \geqslant 0 \\ n_\mathrm{n} = 0; \vec{n} \cdot \left(-\mu_\mathrm{n} \vec{E}\right) < 0 \end{array}$	Neumann/Dirichlet
4	Ground $\phi = 0$	Dirichlet

Table 3: Boundary conditions for (Boundary surface 3)

7.4 Boundary 4

Equation	Boundary condition	Boundary condition type
1	Outflow $\vec{n} \cdot (-D_\mathrm{e} \nabla n_\mathrm{e}) = 0$	Neumann
2	$n_p = 0$	Dirichlet
3	Outflow $\vec{n} \cdot (-D_\mathrm{n} \nabla n_\mathrm{n}) = 0$	Neumann
4	Ground $\phi = 0$	Dirichlet

Table 4: Boundary conditions for (Boundary surface 4)

7.5 Boundary 5 and 6

Equation	Boundary condition	Boundary condition type
1	Open boundary $\vec{n}\cdot(-D_{\mathrm{e}}\nabla n_{\mathrm{e}})=0;\vec{n}\cdot\left(-\mu_{\mathrm{e}}\vec{E}\right)\geqslant 0$ $n_{\mathrm{e}}=0;\vec{n}\cdot\left(-\mu_{\mathrm{e}}\vec{E}\right)<0$	Neumann/Dirichlet
2	Open boundary $\vec{n}\cdot(-D_{\mathrm{p}}\nabla n_{\mathrm{p}})=0;\vec{n}\cdot\left(-\mu_{\mathrm{p}}\vec{E}\right)\geqslant 0$ $n_{\mathrm{p}}=0;\vec{n}\cdot\left(-\mu_{\mathrm{p}}\vec{E}\right)<0$	Neumann/Dirichlet
3	Open boundary $\vec{n}\cdot(-D_{\mathrm{n}}\nabla n_{\mathrm{n}})=0;\vec{n}\cdot\left(-\mu_{\mathrm{n}}\vec{E}\right)\geqslant 0$ $n_{\mathrm{n}}=0;\vec{n}\cdot\left(-\mu_{\mathrm{n}}\vec{E}\right)<0$	Neumann/Dirichlet
4	Zero charge $\vec{n}\cdot\left(\epsilon\vec{E}\right)<0$	Neumann

Table 5: Boundary conditions for (Boundary surfaces 5 and 6)

8 Code Review

pdeapp.m

```
% clear exasim data from memory
clear pde mesh master dmd sol;

% Add Exasim to Matlab search path
cdir = pwd(); ii = strfind(cdir, "Exasim");
run(cdir(1:(ii+5)) + "/Installation/setpath.m");

% create pde and mesh for each PDE model
pdeapp1;
pdeapp2;

% call exasim to generate and run C++ code to solve the PDE models
[sol,pde,mesh,master,dmd,compilerstr,runstr] = exasim(pde,mesh);

% visualize the numerical solution of the PDE model using Paraview
for m = 1:length(pde)
    pde{m}.visscalars = {"temperature", 1}; % list of scalar fields
        for visualization
    pde{m}.visvectors = {"temperature gradient", [2 3]}; % list of
        vector fields for visualization
    pde{m}.visfilename = "dataout" + num2str(m) + "/output";
    vis(sol{m},pde{m},mesh{m}); % visualize the numerical solution
end
```

pdeapp1.m

```
% Physical parameters
Kep = 2e-13;        % mu[1] Recombination coeff - pos and neg ions
   [m^3/s]
Knp = 2e-13;        % mu[2] Recombination coeff - pos ions and
   electrons [m^3/s]
mu_p = 2.43e-4;     % mu[3] Pos ion mobility [m^2/(Vs)]
mu_n = 2.7e-4;      % mu[4] Neg mobility [m^2/(Vs)]
De = 0.18;          % mu[5] Electron diffusion coefficient [m^2/s]
Dp = 0.028e-4;      % mu[6] Pos ion diffusion coefficient [m^2/s]
Dn = 0.043e-4;      % mu[7] Neg diffusion coefficient [m^2/s]
Nmax = 1e16;        % mu[8] Max number density for initial charge
   distribution [particles/m^3]
r0 = 0.0;           % mu[9] r-pos of emitter tip in reference frame
   [m]
z0 = 0.045;         % mu[10]z-pos of emitter tip in reference frame
   [m]
s0 = 1e-2;          % mu[11]Std deviation of initial charge
   distribution [m]
e = 1.6022e-19;     % mu[12]Charge on electron [C]
epsilon = 8.854e-12; % mu[13]absolute permittivity of air
   [C^2/(N*m^2)]
Ua = -10e3;         % mu[14]Emitter potential relative to ground [V]
gamma = 0.001;      % mu[15]Secondary electron emission coefficient
   [1/m]
E_bd = 3e6;         % mu[16]Breakdown E field in air [V/m]
r_tip = 220e-6;     % mu[17] Tip radius of curvature [m]

% Set discretization parameters, physical parameters, and solver
   parameters
                %   1  2   3    4    5 6  7   8   9  10 11 12  13   14
                    15   16   17
pde{1}.physicsparam = [Kep, Knp, mu_p, mu_n, De, Dp, Dn, Nmax, r0,
   z0, s0, e, epsilon, Ua, gamma, E_bd, r_tip];



% Mesh
[mesh{1}.p,mesh{1}.t] = gmshcall(pde{1}, "chen_geom_coarse.msh", 2,
   0);

% expressions for domain boundaries
eps = 1e-4;
xmin = min(mesh{1}.p(1,:));
xmax = max(mesh{1}.p(1,:));
ymin = min(mesh{1}.p(2,:));
ymax = max(mesh{1}.p(2,:));
```

```
x2 = 0.017;
x3 = 0.015;

bdry1 = @(p) (p(1,:) < xmin+eps); % axis symmetric boundary
bdry2 = @(p) (p(1,:) > xmax - eps); % open boundary 1
bdry3 = @(p) (p(2,:) > ymax - eps); % open boundary 2
bdry4 = @(p) (p(2,:) < ymin+eps) && (p(1,:) < x3+eps); % grounded
   boundary - open
bdry5 = @(p) (p(2,:) < ymin+eps) && (p(1,:) > x2-eps); % grounded
   boundary
bdry6 = @(p) (p(2,:) < 0.04);              % grounded boundary -
   cylinder
bdry7 = @(p) (p(1,:) < x2+eps);            % needle tip

mesh{1}.boundaryexpr = {bdry1, bdry2, bdry3, bdry4, bdry5, bdry6,
   bdry7};
mesh{1}.boundarycondition = [2, 5, 5, 3,   4,   4,   1]; % Set
   boundary condition for each boundary



% Solver configuration parameters
pde{1}.porder = 2;    % polynomial degree
pde{1}.tau = 1.0;     % DG stabilization parameter
pde{1}.NLtol = 1.0e-6;
pde{1}.linearsolvertol = 1.0e-4;
pde{1}.ppdegree = 20;
pde{1}.precMatrixType = 2;

% solver parameters
pde{1}.torder = 1;    % time-stepping order of accuracy
pde{1}.nstage = 1;    % time-stepping number of stages
pde{1}.dt = 1.0e-7*ones(1,3); % time step sizes
pde{1}.visdt = 1.0e-6; % visualization timestep size
pde{1}.soltime = 1:pde{1}.visdt:length(pde{1}.dt); % steps at which
   solution are collected
pde{1}.GMRESrestart=25;    % number of GMRES restarts
pde{1}.linearsolveriter=50; % number of GMRES iterations
pde{1}.NLiter=2;           % Newton iterations

% set indices to obtain v from the solutions of the other PDE models
% first column : model index
% second column: solution index
pde{1}.vindx = [2 1; 2 2; 2 3]; % check this
pde{1}.subproblem = 1;
```

pdeapp2.m

```
% set indices to obtain v from the solutions of the other PDE models
% first column : model index
% second column: solution index
pde{2}.porder = 2;    % polynomial degree
pde{2}.vindx = [1 1];
pde{2}.subproblem = 1;

pde{2}.NLtol = 1.0e-6;
pde{2}.linearsolvertol = 1.0e-4;
pde{2}.ppdegree = 20;
pde{2}.precMatrixType = 2;

% solver parameters
pde{2}.torder = 1;    % time-stepping order of accuracy
pde{2}.nstage = 1;    % time-stepping number of stages
pde{2}.dt = 1.0e-7*ones(1,3); % time step sizes
pde{2}.visdt = 1.0; % visualization timestep size
pde{2}.soltime = 1:pde{2}.visdt:length(pde{2}.dt); % steps at which
   solution are collected
pde{2}.GMRESrestart=25;    % number of GMRES restarts
pde{2}.linearsolveriter=50; % number of GMRES iterations
pde{2}.NLiter=2;           % Newton iterations
```

pdemodel1.m

```
function pde = pdemodel1
pde.mass = @mass;
pde.flux = @flux;
pde.source = @source;
pde.fbou = @fbou;
pde.ubou = @ubou;
pde.initu = @initu;
pde.initw = @initw;
end

function m = mass(u, q, w, v, x, t, mu, eta)
   r = x(1);
   m = r*[1.0, 1.0, 1.0]; % Multiply by r for axisymmetric
end

function s = source(u, q, w, v, x, t, mu, eta)
   s = sym([0, 0, 0, 0]);
end

function f = flux(u, q, w, v, x, t, mu, eta)
```

```
r = x(1);

disp(size(w))
Ex = w(2);   % Check to make sure the sign is correct
Ey = w(3);

mu_e = 1.9163*((Ex^2 + Ey^2)^0.5)^(-0.25); % Ionization
   coefficient [1/m]
mu_p = mu(3);
mu_n = mu(4);
De = mu(5);
Dp = mu(6);
Dn = mu(7);
E_bd = mu(16);
r_tip = mu(17);

ne = u(1);
np = u(2);
nn = u(3);

% Nondimensional groups
A1 = 1;
B1 = De/(r_tip*mu_e*E_bd);

A2 = mu_p/mu_e;
B2 = Dp/(r_tip*mu_e*E_bd);

A3 = mu_n/mu_e;
B3 = Dn/(r_tip*mu_e*E_bd);

%%%% Eqn 1
%%% Note + sign in '+De*q' because q = -grad(u)
f11 = -A1*ne*Ex +B1*q(1); % x
f12 = -A1*ne*Ey +B1*q(4); % y

%%%% Eqn 2
f21 = A2*np*Ex +B2*q(2); % x
f22 = A2*np*Ey +B2*q(5); % y

%%%% Eqn 3
f31 = -A3*nn*Ex +B3*q(3); % x
f32 = -A3*nn*Ey +B3*q(6); % y

fx = [f11 f21 f31];
fy = [f12 f22 f32];
f = r*[fx(:) fy(:)];
```

```
end

function fb = fbou(u, q, w, v, x, t, mu, eta, uhat, n, tau)
    % pde fluxes
    f = flux(u, q, w, v, x, t, mu, eta);

    % numerical flux
    fh = f(:,1)*n(1) + f(:,2)*n(2) + tau*(u - uhat);

    ne = u(1);
    np = u(2);
    nn = u(3);
    disp(size(w))
    Ex = w(2);   % Check the dimension of w and make sure the sign
       is correct
    Ey = w(3);

    mu_e = 1.9163*((Ex^2 + Ey^2)^0.5)^(-0.25); % Ionization
       coefficient [1/m]
    mu_p = mu(3);
    mu_n = mu(4);
    gamma = mu(15);

    % inviscid fluxes
    fix = [-ne*Ex (mu_p/mu_e)*np*Ex -(mu_n/mu_e)*nn*Ex]; fix =
       fix(:); % Nondimensionalized
    fiy = [-ne*Ey (mu_p/mu_e)*np*Ey -(mu_n/mu_e)*nn*Ey]; fiy =
       fiy(:);
    fih = fix*n(1) + fiy*n(2); %+ tau*(u - uhat); check

    % boundary flux on the needle tip - Chen boundary 1
    fb1 = fh;
    fb1(1) = -gamma*np*(((mu_p/mu_e)*Ex)^2 +
       ((mu_p/mu_e)*Ey)^2)^0.5; % Nondimensionalized check sign here
       (-n)
    fb1(2) = fih(2);

    % axis symmetric boundary condition - axisymmetric BC means the
       diffusive flux=0
    fb2 = [0; 0; 0];

    % Chen bdry 3
    En = Ex*n(1) + Ey*n(2);
    signEn = tanh(1e3*(-En));
    alpha = 0.5 + 0.5*signEn; % Alpha=1 when (-En) is positive:
    fb3 = alpha*fih + (1-alpha)*fh; % Check for this sign being
       flipped in the switch
```

```
    % Grounded boundary -> 4 in paper
    fb4 = fih;
    fb4(2) = fh(2);

    fb = [fb1 fb2 fb3 fb4 fb3]; % Note: For eqns 1-3, BCs 3,5&6 are
        the same (open boundary)
end

function ub = ubou(u, q, w, v, x, t, mu, eta, uhat, n, tau)
    E_bd = mu(16);
    r_tip = mu(17);
    Ua = mu(14);

    % needle tip - boundary 1
    ub1 = u;
    ub1(3) = 0;

    % axis symmetric boundary 2
    ub2 = u;

    % grounded boundary
    Ex = w(2);   % Check to make sure the sign is correct
    Ey = w(3);
    En = Ex*n(1) + Ey*n(2);
    signEn = tanh(1e3*(-En));
    alpha = 0.5 + 0.5*signEn;
    ub3 = alpha*u + (1-alpha)*[0,0,0];

    ub4 = u;
    ub4(2) = 0;

    ub = [ub1 ub2 ub3 ub4 ub3]; % Note: For eqns 1-3, BCs 3,5&6 are
        the same (open boundary)
end

function u0 = initu(x, mu, eta)
    r = x(1);
    z = x(2);

    r0 = mu(9);
    z0 = mu(10);
    s0 = mu(11);

    g = exp(-(r-r0)^2/(2*s0^2) - (z-z0)^2/(2*s0^2)); %
        Nondimensionalized by N_max
```

```
    % Eqn 1
    u1_0 = g;

    % Eqn 2
    u2_0 = g;

    % Eqn 3
    u3_0 = 0;

    u0 = [u1_0, u2_0, u3_0];
end

function w0 = initw(x, mu, eta)
    w0 = sym([0; 0; 0]);
end
```

pdemodel2.m

```
function pde = pdemodel2
pde.mass = @mass;
pde.flux = @flux;
pde.source = @source;
pde.fbou = @fbou;
pde.ubou = @ubou;
pde.initu = @initu;
pde.initw = @initw;
end

function m = mass(u, q, w, v, x, t, mu, eta)
    m = sym(0.0);   % Multiply by r for axisymmetric
end

function f = flux(u, q, w, v, x, t, mu, eta)
f = [-q(1) -q(2)]; % Check the sign here!
end

function s = source(u, q, w, v, x, t, mu, eta)
    s = sym(0.0);
end

function fb = fbou(u, q, w, v, x, t, mu, eta, uhat, n, tau)
    f = flux(u, q, w, v, x, t, mu, eta);
    fh = f(1)*n(1) + f(2)*n(2) + tau*(u(1)-0.0);

    fb = [fh 0 fh fh 0];
end
```

```
function ub = ubou(u, q, w, v, x, t, mu, eta, uhat, n, tau)
    E_bd = mu(16);
    r_tip = mu(17);
    Ua = mu(14);
    ub = [-Ua/(E_bd*r_tip), w(1), 0, 0, w(1)]; % Need to check that
        the solution variable for the current equation is w here and
        not u
end

function u0 = initu(x, mu, eta)
    u0 = sym(0.0);
end

function w0 = initw(x, mu, eta)
    w0 = sym(0.0);
end
```

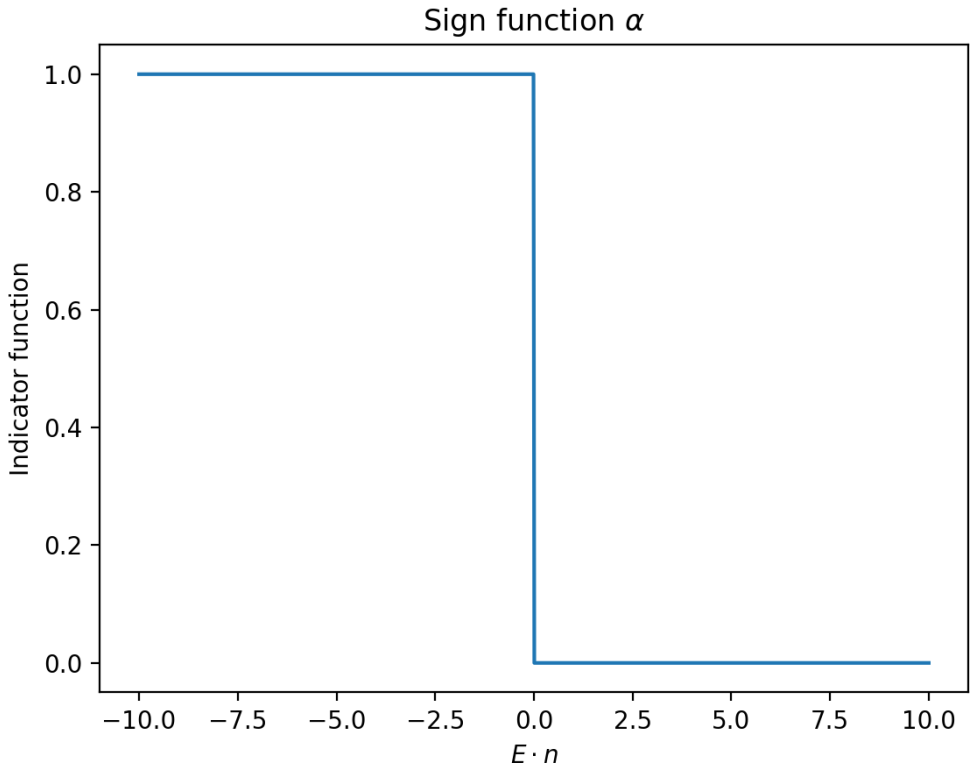

Figure 1: Indicator function for $E \cdot n$ used in this problem

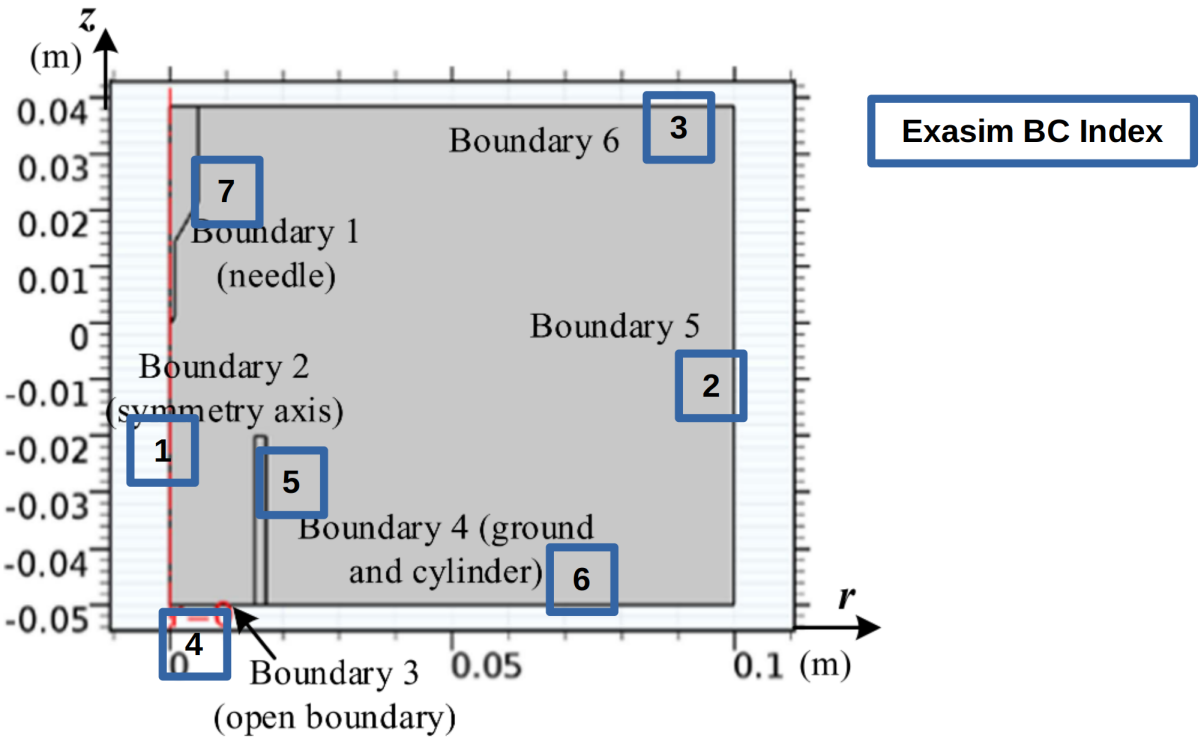

Figure 2: Computational domain of Chen 2017 with Exasim BC indices overlaid