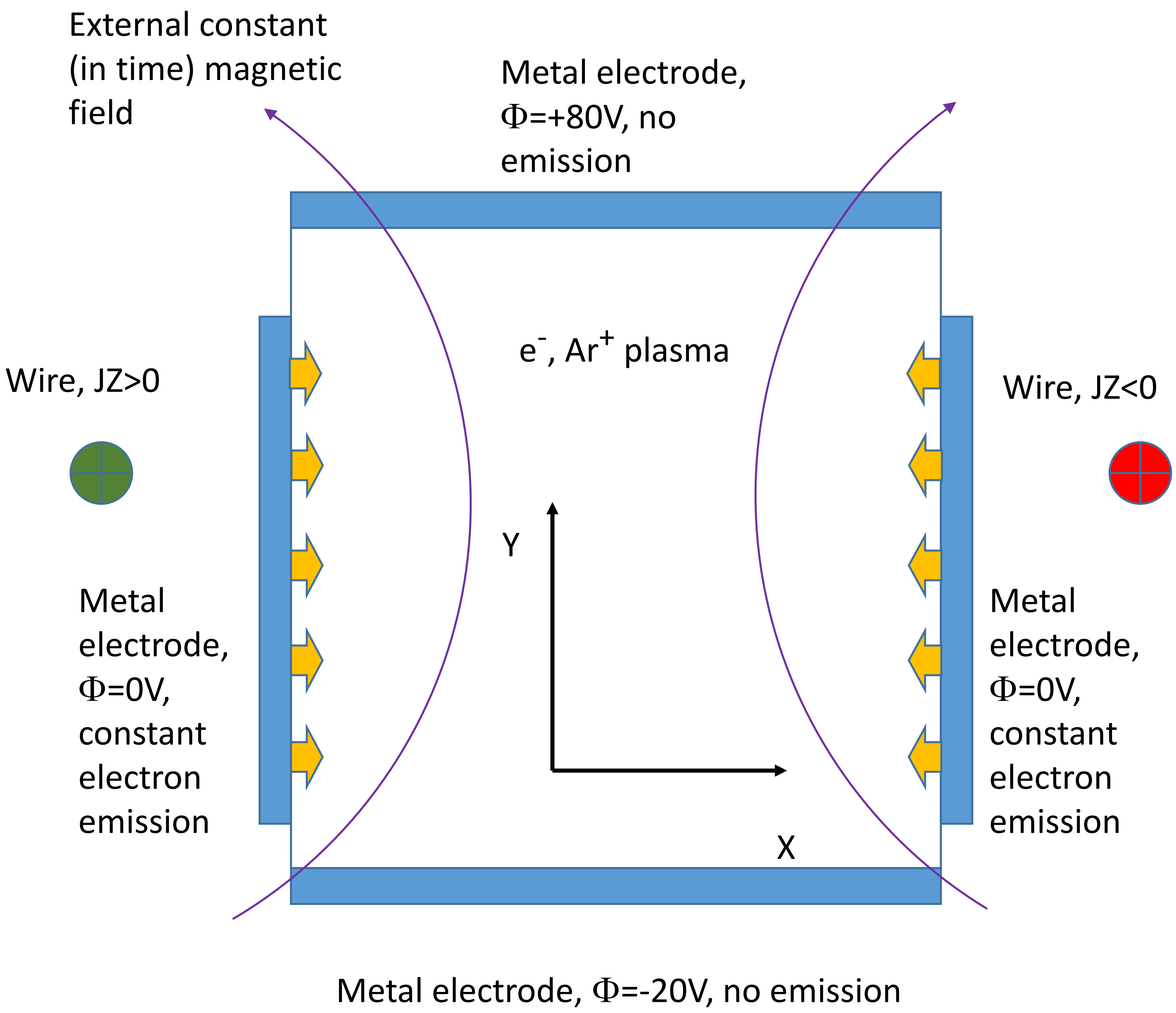
EDIPIC-2D input data description

Dmytro Sydorenko


External constant (in time) magnetic field
Metal electrode, Φ=+80V, no emission
e^-, Ar^+ plasma
Wire, JZ>0
Wire, JZ<0
Y
X
Metal electrode, Φ=0V, constant electron emission
Metal electrode, Φ=0V, constant electron emission
Metal electrode, Φ=-20V, no emission

The set of input data files discussed below was prepared for simulation of this hollow cathode system with 32 MPI processes:

- The whole system is 241x481 nodes or 8mmx16mm;
- start plasma density 1e16 m^{-3}, 20 particles per cell;
- left and right electrodes (0.67mm<y<13.33mm) emit 100 particles per timestep each;
- vacuum gap boundaries between the electrodes;
- magnetic field is created by two opposite JZ currents at x=-10mm,y=8mm and x=18mm,y=8mm;
- Ions and neutrals are Argon;
- neutral density 1e21m^{-3};
- elastic, excitation, ionization collisions are on.

Input data files

- **General configuration:**
  - **init_configuration.dat**
- **Properties of boundary objects:**
  - **init_bo_01.dat**
  - **init_bo_03.dat**
  - **init_bo_05.dat**
  - **init_bo_07.dat**
- **External fields:**
  - **init_extfields.dat**
  - **init_extmagfieldsBxBy.dat**
- **Neutral gas parameters:**
  - **init_neutrals.dat**
  - **init_neutral_Argon0.dat**
- **Electron-neutral collisions cross-sections:**
  - **init_neutral_Argon0_crsect_coll_id_01_type_10.dat**
  - **init_neutral_Argon0_crsect_coll_id_02_type_20.dat**
  - **init_neutral_Argon0_crsect_coll_id_03_type_30.dat**
- **Parameters of initial particle distribution:**
  - **init_particles.dat**
- **Diagnostics:**
  - **init_probes.dat**
  - **init_snapshots.dat**
- **Simulation control:**
  - **init_simcontrol.dat**
- **PETSc field solver configuration:**
  - **petsc.rc**

Strictly necessary files, simulation cannot proceed without these.

If electron-neutral collisions are enabled, simulation requires files with corresponding cross-sections, it cannot proceed without these files.

This file is necessary for all systems requiring a PETSc-based field solver, which includes non-periodic systems, systems periodic in one direction with segmented (nonuniform) boundaries, systems periodic in two directions.

init_configuration.dat, part 1

```
---dddd.ddddddd----- scale electron temperature [eV]
      4.0000000
---+d.dddddddE+dd--- scale electron density [m^-3]
    1.0000000E+17
---ddd---------- number of cells per scale electron Debye length
     2
---ddd---------- maximal expected velocity [units of scale thermal electron velocity]
     6
---ddd---------- number of blocks (processes) along the X (horizontal) direction
     4
---ddd---------- number of blocks (processes) along the Y (vertical) direction
     8
---ddd---------- number of cells along the X-direction in a block
    60
---ddd---------- number of cells along the Y-direction in a block
    60
--dddd---------- number of macroparticles per cell for the scale density
   200
```

Define size of a cell of the numerical grid $\Delta x=\lambda_{D,scale}/N_D$. Initial electron density/temperature do not have to be equal to the scale values.

Define time step via the Courant condition $\Delta t=\Delta x/V_{max}$.

Define splitting of the simulation domain between all MPI processes. Presently this splitting is used in the PETSc field solver. Works for a rectangular simulation domain. See also the next page.

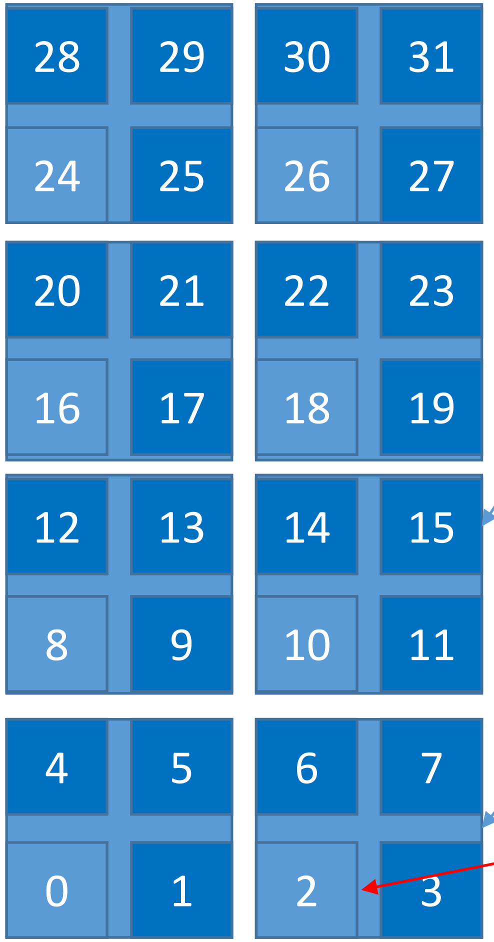
Define size of the subdomain belonging to a single MPI process, used in the PETSc field solver.

Define the number of macroparticles.

init_configuration.dat, part 2

```
•   -----d---------- number of blocks in a cluster along the X-direction
•         2
•   -----d---------- number of blocks in a cluster along the Y-direction
•         2
```

Define grouping of domains belonging to field solvers (blocks) into larger domains (clusters) which are used in particle advancing procedures. In this particular case, the whole domain was split into 32 blocks (4 along x by 8 along y, see the previous page), and 2 by 2 groups of neighbor blocks were combined into 8 clusters (2 clusters along x by 4 clusters along y). This grouping defines (a) which processes receive charge density from a cluster to solve the Poisson equation and (b) the number of processes working on cluster particles when the particle load is even (i.e. the number of particles inside each cluster is the same).



28	29	30	31
24	25	26	27
20	21	22	23
16	17	18	19
12	13	14	15
8	9	10	11
4	5	6	7
0	1	2	3

“Block” (domain assigned to one MPI process for solving the field equation).

“Cluster” (domain where one or more processes advance particles).

“Cluster master” – a process which is always attached to a specific cluster and carries all the necessary information.

In this diagram, numbers in squares are global MPI ranks of processes.

init_configuration.dat, part 3

```
---ddd---ddd---- number of objects along domain boundary // number of material inner
objects (>=0), each inner objects is a rectangle
     8     0
===dd===dd=== object type, number of segments // flat electrode at the left, #1
    1    1
---dddddd---dddddd---dddddd---dddddd---segment start X/Y end X/Y [global node index]
        0       20         0      400
===dd===dd=== object type, number of segments // vacuum gap left top, #2
    0    1
---dddddd---dddddd---dddddd---dddddd---segment start X/Y end X/Y [global node index]
        0      400         0      481
===dd===dd=== object type, number of segments // flat electrode above, #3
    1    1
---dddddd---dddddd---dddddd---dddddd---segment start X/Y end X/Y [global node index]
        0      481       241      481
===dd===dd=== object type, number of segments // vacuum gap right top, #4
    0    1
---dddddd---dddddd---dddddd---dddddd---segment start X/Y end X/Y [global node index]
      241      400       241      481
```

Define number of boundary objects (objects placed along the domain boundary) and inner objects (rectangular objects placed inside the domain). This setup has no inner objects. An example with an inner object is given in slide 8.

This is a full entry for one boundary object which has one segment, that is can be represented by a straight line.

This line defines coordinates of the start and the end of the segment of the boundary object. In general, an object may have more than one segment. In this case, each segment would require an additional line here with the segment’s start and end coordinates.

init_configuration.dat, part 4

```
===dd===dd=== object type, number of segments // flat electrode at the right, #5
    1    1
---dddddd---dddddd---dddddd---dddddd---segment start X/Y end X/Y [global node index]
      241       20      241      400
===dd===dd=== object type, number of segments // vacuum gap right bottom, #6
    0    1
---dddddd---dddddd---dddddd---dddddd---segment start X/Y end X/Y [global node index]
      241        0      241       20
===dd===dd=== object type, number of segments // flat electrode below, #7
    1    1
---dddddd---dddddd---dddddd---dddddd---segment start X/Y end X/Y [global node index]
        0        0      241        0
===dd===dd=== object type, number of segments // vacuum gap left bottom, #8
    0    1
---dddddd---dddddd---dddddd---dddddd---segment start X/Y end X/Y [global node index]
        0        0        0       20


INTEGER, PARAMETER :: VACUUM_GAP = 0
INTEGER, PARAMETER :: METAL_WALL = 1
INTEGER, PARAMETER :: PERIODIC_PIPELINE_X = 2
INTEGER, PARAMETER :: PERIODIC_PIPELINE_Y = 3
INTEGER, PARAMETER :: DIELECTRIC = 4
INTEGER, PARAMETER :: SYMMETRY_PLANE = 5 !!! only at the left edge !!!
```

Presently, 6 types of boundary objects are possible (see the bottom of file init_configuration.dat):

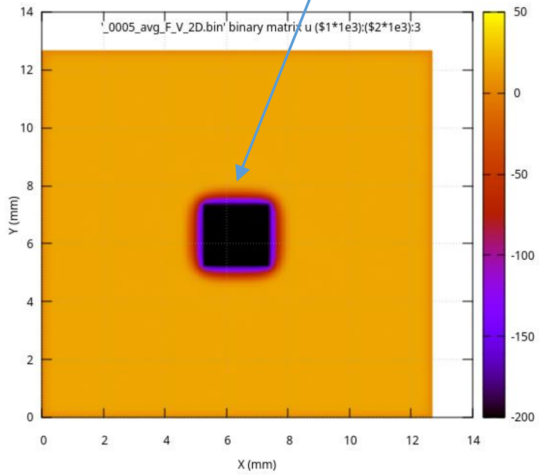
- Vacuum gap = no emission, particles are absorbed, potential along this boundary changes linearly between potentials of endpoints which equal the potentials of neighbor electrodes;
- Metal wall = emission possible, potential given;
- Periodic boundary along X;
- Periodic boundary along Y;
- Dielectric = presently this only works for systems periodic along the X direction with an infinite dielectric half-space(s) at one or both Y boundaries.
- Symmetry plane, can be applied only at the left edge (x=0) of the domain.

These lines are here for convenience only, to remind values for different object types.

init_configuration.dat, part 5, additional parameters (from input_data_dc_inner_object_avg_heat_flows)

```
---ddd---ddd---- number of objects along domain boundary // number of material inner objects (>=0), each inner objects is a
rectangle
     4     1
===dd===dd=== object type, number of segments // vertical metal wall on the left, no emission, #1
    1    1
---dddddd---dddddd---dddddd---dddddd--- segment start X/Y end X/Y [global node index]
        0        0        0      241
===dd===dd=== object type, number of segments // horizontal metal wall at the top, no emission, #2 @@@
    1    1
---dddddd---dddddd---dddddd---dddddd--- segment start X/Y end X/Y [global node index]
        0      241      241      241
===dd===dd=== object type, number of segments // vertical metal wall on the right, no emission, #3
    1    1
---dddddd---dddddd---dddddd---dddddd--- segment start X/Y end X/Y [global node index]
      241        0      241      241
===dd===dd=== object type, number of segments // horizontal metal wall on the bottom, no emission, #4 @@@
    1    1
---dddddd---dddddd---dddddd---dddddd--- segment start X/Y end X/Y [global node index]
        0        0      241        0
===dd=== object type // inner metal object, #5
    1
---dddddd---dddddd---dddddd---dddddd--- coordinates of left bottom X/Y corner and right top X/Y corners [global node index]
      100      100      140      140
---dddd---dddd--- for a system periodic along both X and Y directions, without objects with given potential, specify X/Y
(global node index i/j) of a point with given potential
    120    240
---ddddd.d------- value of the potential at this point [V]
          0.0
```

In this example there is one **inner object**.



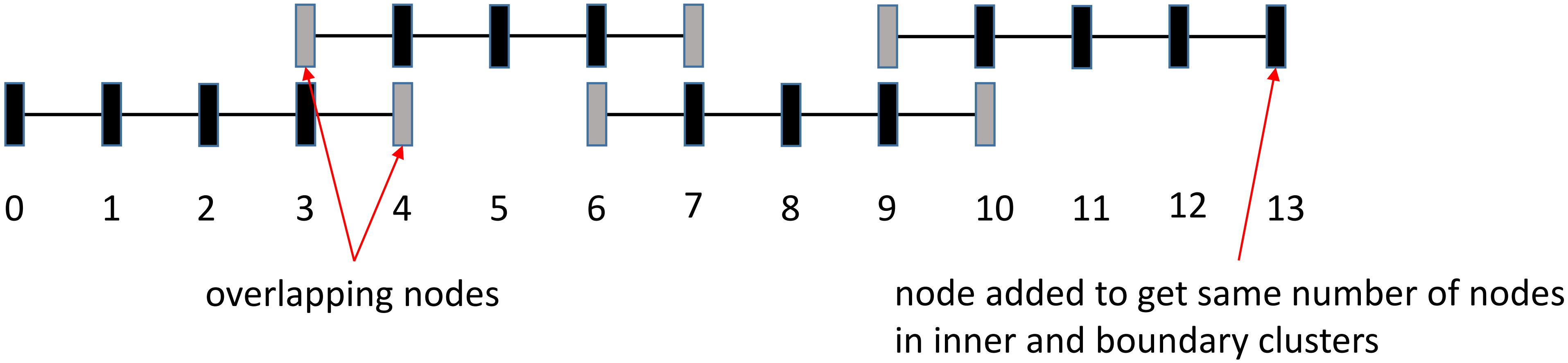
This figure shows time-averaged (t=40-50ns) electrostatic potential vs coordinate, the central electrode (the inner object, #5) has potential -200 V.

These lines define the inner object. There must be a 4-line block like this for each inner object. The inner object description must be after the description of the last boundary object. Inner metal objects can be only of type 1 (metal) or 4 (dielectric). An inner object is always a rectangle, which is why its dimensions are defined by the left bottom/right top corner coordinates.

These lines are necessary only when the system is periodic along both the x and y directions. For all other systems they are ignored.

About specifying coordinates of ends of boundary objects and about the size of the domain

- Coordinates of start/end points of segments of boundary objects are given in indices of nodes. For a domain which has 4 blocks 60 cells each along the X direction, range of node indices in the X direction is from 0 to 4x60+1=241. Similar, in the Y direction, the range of domain node indices along the Y direction is from 0 to 8x60+1=481.
- Indexing begins with zero because it allows to find indices of the left bottom node of the cell containing a particle as i=int(x), j=int(y).
- The “plus 1” in the index of the node with the maximal x and/or y appears because cluster [and block] domains overlap by one cell and we want to have cluster domains to be of the same size. Here is the example of node numbering in 1d with 4 clusters, 3 nodes/cells each:


0
1
2
3
4
5
6
7
8
9
10
11
12
13
overlapping nodes
node added to get same number of nodes in inner and boundary clusters

Domains of each block and cluster are saved in files box_proc_NNNN.dat (blocks) and master_proc_NNNN.dat (cluster) where NNNN is the global MPI rank of the process responsible for the block or cluster. In these files, columns 1,2 are domain corners coordinates in node indices, column 3 is the process rank, columns 4,5 are domain corners coordinates in meters.

init_bo_01.dat

A boundary object may require other parameters in addition to its type and geometry given in init_configuration.dat. Some of these parameters are set here.

```
------AAAAAA--- code/abbreviation of the material, character string
      METAL0
---ddddd.ddd--- constant potential [V]
       0.000
---ddddd.ddd--- amplitude of potential oscillations [V]
       0.000
---ddddd.ddd--- frequency [MHz]
       0.000
---ddddd.ddd--- phase [deg] for sin(omega*t+phase)
       0.000
---ddddd.ddd--- number of electron macroparticles injected each timestep, constant [dim-less]
     100.000
-------d------- emission model (0 = thermal emission, 1 = electron beam)
       0
----dddd.ddd--- temperature of emitted electrons / half-energy-spread of the beam (Tb) normal
to the wall [eV] (>=0)
       0.200
----dddd.ddd--- temperature of emitted electrons parallel to the wall [eV] (>=0)
       0.200
----dddd.ddd--- energy of the electron beam [eV] (>3Tb/2)
       0.000
-------d------- save ions collided with this object in snapshots? (1/0 = Yes/No)
       0
-------d------- save electrons collided with this object in snapshots? (1/0 = Yes/No)
       0
```

If file init_bo_NN.dat is not found, for a boundary object NN default settings are applied: constant zero potential, no emission. Note that these settings are meaningful for a METAL_WALL object only.

Properties of an inner object can be set using a similar data file.

Define material of the boundary. Selecting certain material specifies properties like secondary electron emission (requires a separate input file).

Define wall potential as a given function of time $\Phi(t) = \Phi_{const} + \Phi_{osc}\sin(\omega t + \varphi)$.

Define intensity of constant electron injection and its temperature. Positions of injected particles are distributed randomly/uniformly along the boundary object. The number can have fractional part. The fractional part is treated as the probability of injecting a particle during the timestep. It allows to use small injection currents.

Select emission model, thermal or beam.

Set temperature (energy spread) of injected electrons, in case of injection of a beam set the beam energy.

Turn these flags on if you want to save particles (electrons or ions) collided with this boundary object during time intervals specified in init_snapshots.dat. Post-processing of such a data allows to obtain energy spectra, particle, and energy fluxes of particles collided with the object as functions of position along the object surface.

init_extfields.dat

```
•  ---ddddd.ddd--- X-magnetic field [Gauss]
•         0.000
•  ---ddddd.ddd--- Y-magnetic field [Gauss]
•         0.000
•  --------------- parameters of Z-magnetic field as used by Boeuf and Garrigues
•  ---ddddd.ddd--- Y-coordinate of the BZ maximum, y_Bmax [cm]
•         0.750
•  ---ddddd.ddd--- BZ at y=0 [Gauss]
•         0.000
•  ---ddddd.ddd--- BZ at y_Bmax [Gauss]
•         0.000
•  ---ddddd.ddd--- BZ at y=Lsys [Gauss]
•         0.000
•  ---ddddd.ddd--- characteristic length of decay for y<y_Bmax [cm]
•         0.625  0.600
•  ---ddddd.ddd--- characteristic length of decay for y>y_Bmax [cm]
•         0.625  0.600
•  ---ddddd.ddd--- Z-electric field [V/cm]
•         0.000
•  -------d------- ions sense magnetic field [1=Yes, 0=No]
•         0 1
•  -------d------- ions sense Z-electric field [1=Yes, 0=No]
•         0
```

Set constant (in time) and uniform magnetic field in the X-direction.

Set constant and uniform magnetic field in the Y-direction.

These parameters define the constant magnetic field in the Z-direction which is a function of Y coordinate. The profile was used in axial-azimuthal simulations of a Hall thruster.

Set constant and uniform electric field in the Z-direction.

Turn on/off the magnetic field in the ion motion equations.

Turn on/off the electric field along the Z-direction in the ion motion equations.

init_extmagfieldsBxBy.dat

```
---dd--- number of wires with the electric current along the Z-direction
    2
--- provide below for each wire its X coordinate [mm] Y coordinate [mm] and electric current JZ [A]
---ddddd.ddd---ddddd.ddd---ddddd.ddd---
     -10.000       8.000      30.000
      18.012       8.000     -30.000
```

Here one can specify any [reasonable] number of wires with the electric current along the Z-direction. The currents will create a nonuniform magnetic field in the plane X-Y. The values of magnetic field components are calculated analytically (Ampere’s law), as a superposition of fields from each wire, to ensure that the magnetic field vector has zero divergence. One has to provide coordinates and the current value for each wire. The sign of the current is important. The magnetic field created by these wires is added to the constant uniform magnetic field {Bx,By} defined in file init_extfields.dat . The code saves vector components of the Bx,By magnetic field in files proc_NNNN_BxBy_vs_xy.dat . Here NNNN is the global MPI rank of a process (master of a cluster) which saves the magnetic field within its domain (cluster). In this file, columns 1,2 are the node indices in the x and y directions, columns 3,4 are the dimensional node x and y coordinates in units of meters, columns 5,6 are the magnetic field components Bx and By in units of Tesla.

init_neutrals.dat

```
•   ----------dd--- number of neutral species
•              1
•   ------AAAAAA--- code/abbreviation of the neutral gas, character string
•         Argon0
•   --#d.dddE#dd--- Density [m^-3]
•      1.000E+21
•   -----ddddd.d--- Temperature [K]
•          300.0
```

Define the number of neutral species which will scatter electrons. The code expects to find this number of full data entries below. If it is zero or negative, electron-neutral collisions will be turned off.

These 6 lines form one full data entry for one neutral species selected, in this case Argon. The 6 character string “Argon0” is used to form names of other input data files relevant to this neutral species. The zero in the name is just a filler, no special meaning yet. Here one also specifies the density and the temperature of the neutral species.

Control of electron-neutral collisions is performed via several input data files:

- File **init_neutrals.dat** shown above specifies which neutral species are included in simulation, their density and temperature.
- Files **init_neutral_AAAAAA.dat**, where AAAAAA is a 6 character string representing the neutral gas (e.g. “Argon0”), list and allow to turn on/off possible collisions, as well as provide energy ranges for the cross sections.
- There are also separate data files with cross sections vs energy for each collisional process (for example **init_neutral_Argon0_crsect_coll_id_01_type_10.dat**).

init_neutral_Argon0.dat, part 1

```
•  ---ddd.ddd--- mass [a.m.u.]
•      40.000
•  --------dd--- number of all possible collisional processes
•           3
•  ---dd--d--dd--- collision #1 :: type / activated (1/0 = Yes/No) / ion species produced (ionization collisions only)
•     10   1   0
•  ---dd--d--dd--- collision #2 :: type / activated (1/0 = Yes/No) / ion species produced (ionization collisions only)
•     20   1   0
•  ---dd--d--dd--- collision #3 :: type / activated (1/0 = Yes/No) / ion species produced (ionization collisions only)
•     30   1   1
```

Specify the mass of the neutral in a.m.u.

Specify the number of all available collisional processes.

Define collision type. 10-19 are for elastic collisions (10 implemented, 11-19 reserved); 20-29 are for inelastic collisions (20 implemented, 21-29 implemented); 30 and up are for ionization collisions (only 30 implemented at this time).

Turn this collisional process on (1) or off (0).

For ionization processes, specify here the ordering number of the ion species which is produced in this collision. Has no meaning for elastic or inelastic collisions. In this particular case, there is just one ion species (Ar^+) defined in **init_particles.dat**.

Here collisions # 1,2,3 are elastic, inelastic, and ionization, respectively.

init_neutral_Argon0.dat, part 2

```
--------dd--- number of energy segments for collision probabilities (>0)
        10
--ddddd.ddd------------- minimal energy [eV]
      0.000
--ddddd.ddd---ddd.ddd--- energy segment 1 :: upper boundary [eV] / resolution [eV]
      0.040     0.005
--ddddd.ddd---ddd.ddd--- energy segment 2 :: upper boundary [eV] / resolution [eV]
      0.100     0.010
--ddddd.ddd---ddd.ddd--- energy segment 3 :: upper boundary [eV] / resolution [eV]
      0.400     0.020
--ddddd.ddd---ddd.ddd--- energy segment 4 :: upper boundary [eV] / resolution [eV]
      1.000     0.050
--ddddd.ddd---ddd.ddd--- energy segment 5 :: upper boundary [eV] / resolution [eV]
     10.000     0.200
--ddddd.ddd---ddd.ddd--- energy segment 6 :: upper boundary [eV] / resolution [eV]
     20.000     0.100
--ddddd.ddd---ddd.ddd--- energy segment 7 :: upper boundary [eV] / resolution [eV]
     60.000     0.500
--ddddd.ddd---ddd.ddd--- energy segment 8 :: upper boundary [eV] / resolution [eV]
    200.000     5.000
--ddddd.ddd---ddd.ddd--- energy segment 9 :: upper boundary [eV] / resolution [eV]
   1000.000    10.000
--ddddd.ddd---ddd.ddd--- energy segment 10 :: upper boundary [eV] / resolution [eV]
  10000.000    50.000
```

Here the energy range for collision probability arrays used in simulation is split into several (10) segments of different width with different energy resolution. For example, the first segment is for energies between 0 and 0.04 eV with resolution (step) of 0.005 eV, the second segment is for energies between 0.04 eV and 0.1 eV with resolution of 0.01 eV; and so on. There is 10 segments in total, with the maximal energy of 10000 eV.

The reason for this is to provide sufficient resolution for energies where collision cross-sections are strongly nonuniform and minimize the size of arrays at the same time.

Electron-neutral collision cross-section data files

init_neutral_Argon0_crsect_coll_id_01_type_10.dat

Elastic collisions for Argon, collision id (ordering number) 1, collision type 10 (scattering procedure en_Collision_Elastic_10 does angle scattering depending on particle energy as described in Okhrimovsky et al., Phys.Rev.E, 65, 037402 (2002)).

init_neutral_Argon0_crsect_coll_id_02_type_20.dat

Inelastic collisions for Argon, collision id (ordering number) 2, collision type 20.

init_neutral_Argon0_crsect_coll_id_03_type_30.dat

Ionization collisions for Argon, collision id (ordering number) 3, collision type 30.

Each file contains the number of data points (in the very first line), two columns with energy and cross section, and a brief description with references (the code does not need the description part, it is for convenience only).

Input data files for electron-neutral collisions in Helium

- In sample directory **input_data_dc_semiperiodic_Helium_rcx** (https://github.com/PrincetonUniversity/EDIPIC-2D/tree/main/input_data_dc_semiperiodic_Helium_rcx) there are input files for Helium:
- **init_neutral_Helium.dat**
- **init_neutral_Helium_crsect_coll_id_01_type_11.dat**
  - Elastic collisions for Helium, collision id (ordering number) 1, collision type 11 (scattering procedure en_Colision_Elastic_11 performs isotropic angle scattering independent on the particle energy because the cross-sections table used is the momentum transfer cross-section).
- **init_neutral_Helium_crsect_coll_id_02_type_20.dat**
  - Inelastic collisions for Helium, collision id (ordering number) 2, collision type 20.
- **init_neutral_Helium_crsect_coll_id_03_type_30.dat**
  - Ionization collisions for Helium, collision id (ordering number) 3, collision type 30.

init_particles.dat

Here one defines parameters of initial electron distribution: temperature and density. Those may be different from the scale values set in init_configuration.dat. Presently, the initial density and temperature distribution is uniform.

```
---dddd.ddd----- initial electron temperature [eV]
      1.000
---+d.dddE+dd--- initial electron density [m^-3]
    1.000E+16
------d--------- number of ion species
      1
---ddd.d---+d---dddd.ddd---ddd.dd-- ion mass [amu] / charge [e] / initial temperature [eV] / initial relative concentration [%]
    40.0    1      0.500   100.00
```

Specify the number of ion species

A separate line like this must be provided for each ion species.

Ion mass [amu]

Ion charge [units of e]

Initial ion temperature [eV]

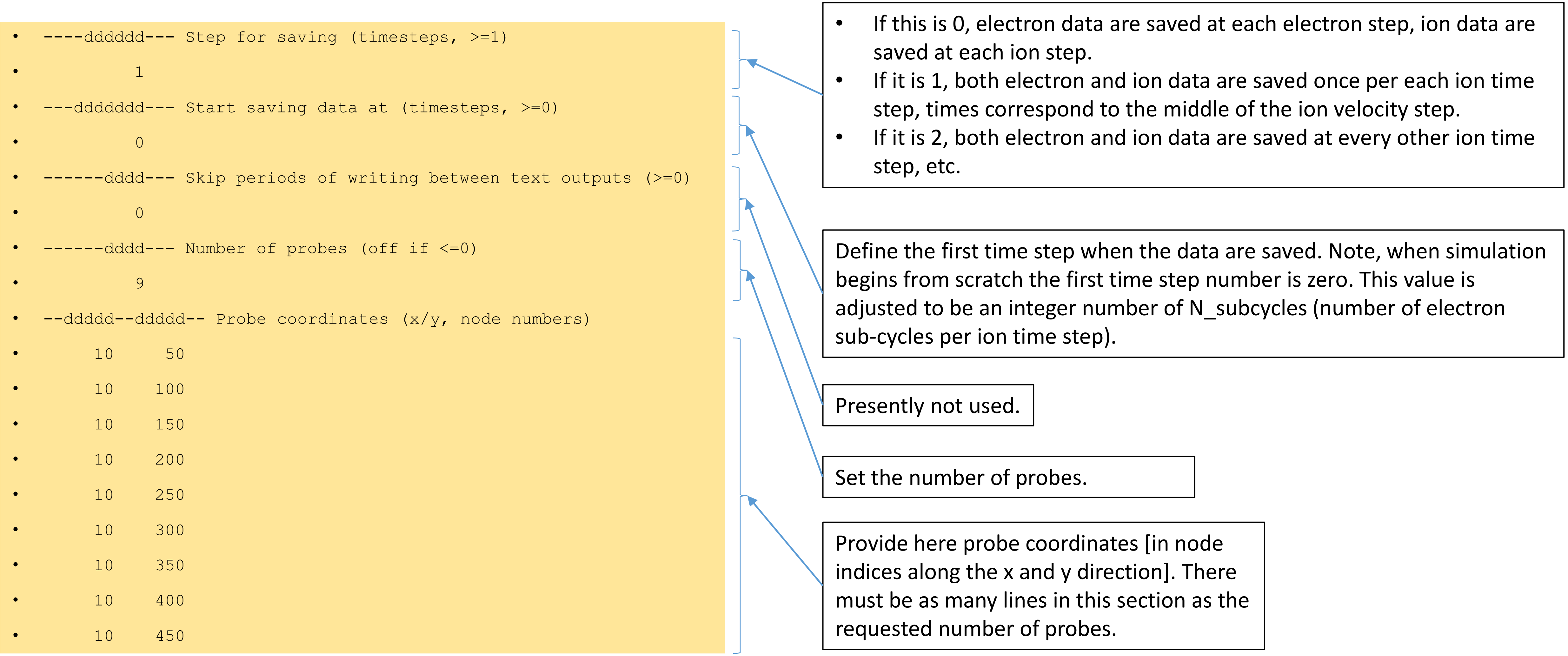
Ratio of initial number densities of the ion species and the electrons [%]. The code will try to have initial number of electron macroparticles equal to the sum of initial numbers of macroparticles of all ion species. So, make sure that the sum of all these initial relative concentrations is 100 [%].

```
---ddddddddd------- seed for random number generator for process with rank zero
   123456789
```

Set the seed for the random number generator.

init_probes.dat

The code can save time dependencies of the following parameters: potential, electric field components Ex and Ey, electron and ion densities, flow velocities (x,y,z), average energies (x,y,z), temperatures (x,y,z), electric currents (x,y,z), heat flow vector components (x,y,z) in certain points called “probes”. This file controls how these data are saved.



```
----dddddd--- Step for saving (timesteps, >=1)
         1
---ddddddd--- Start saving data at (timesteps, >=0)
         0
------dddd--- Skip periods of writing between text outputs (>=0)
         0
------dddd--- Number of probes (off if <=0)
         9
--ddddd--ddddd-- Probe coordinates (x/y, node numbers)
     10     50
     10    100
     10    150
     10    200
     10    250
     10    300
     10    350
     10    400
     10    450
```

- If this is 0, electron data are saved at each electron step, ion data are saved at each ion step.
- If it is 1, both electron and ion data are saved once per each ion time step, times correspond to the middle of the ion velocity step.
- If it is 2, both electron and ion data are saved at every other ion time step, etc.

Define the first time step when the data are saved. Note, when simulation begins from scratch the first time step number is zero. This value is adjusted to be an integer number of N_subcycles (number of electron sub-cycles per ion time step).

Presently not used.

Set the number of probes.

Provide here probe coordinates [in node indices along the x and y direction]. There must be as many lines in this section as the requested number of probes.

init_snapshots.dat, part 1

The code can save 2D snapshots of multiple parameters and electron/ion velocity distribution functions. This file controls how these data are saved.

```
--- save 2D maps of the following parameters? (1=yes, 0=no)
-----F----EX----EY--JXsum--JYsum--JZsum
----dd----dd----dd----dd----dd----dd
     1     1     1     1     1     0
----Ne----JXe---JYe---JZe---VXe---VYe---VZe---WXe---WYe---WZe---TXe---TYe---TZe---QXe---QYe---QZe
----dd----dd----dd----dd----dd----dd----dd----dd----dd----dd----dd----dd----dd----dd----dd----dd
     1     1     1     0     1     1     1     1     1     1     1     1     1     1     1     1
----Ni----JXi---JYi---JZi---VXi---VYi---VZi---WXi---WYi---WZi---TXi---TYi---TZi---QXi---QYi---QZi
----dd----dd----dd----dd----dd----dd----dd----dd----dd----dd----dd----dd----dd----dd----dd----dd
     1     1     1     0     1     1     1     1     1     1     1     1     1     1     1     1
```

Set the flag to 1/0 and turn on/off saving 2D maps of potential, electric fields Ex and Ey, total electric currents Jx, Jy, and Jz. “Total” means “the sum of electric currents due to electrons and all ions”

Turn on/off saving 2D maps of **electron** density, electric currents Jx, Jy, Jz, flow velocities Vx, Vy, Vz, average energies of motion, temperatures, heat flow vector components along the x,y, and z directions.

Same as above but for **ions**.

Default flag values are zero, if the last flags are not specified (for example if the older input file is used) the heat flow vectors are not saved.

The temperatures are defined as $T_{x,y,z} = m\left(\langle v_{x,y,z}^2\rangle - \langle v_{x,y,z}\rangle^2\right)$.

Heat flow vector $\vec{q_s} = \frac{1}{2} n_s m_s \langle c_s^2 \vec{c_s}\rangle$ of species s is defined with respect to the average drift velocity of this species [see Schunk, Reviews of Geophysics and Space Physics, vol.15, p.429, 1977].

Random velocity is $\vec{c_s} = \vec{v_s} - \vec{u_s}$.

Drift velocity is $\vec{u_s} = \langle\vec{v_s}\rangle$.

init_snapshots.dat, part 2

It may be useful to save snapshots with different frequency (i.e. interval between snapshots) at different stages of a simulation.
Snapshots are combined into groups, each group has:

- its start and end time (time when the first and the last snapshot of the group are saved),
- the number of snapshots in the group,
- A set of flags which controls (turns on/off and/or selects content) saving of:
  - the velocity distribution functions (VDFs),
  - phase planes,
  - ionization rates,
  - particles collided with walls.

The group controls are here.
There must be a separate line with start/finish/number-of-snapshots/flags for each requested group. If the number of groups is zero or the number of snapshots in each group is zero, no snapshots will be created.

```
•   ---dd--- Number of groups of snapshots ( >= 0 )

•       3

•   ---ddddddd.ddd---ddddddd.ddd---dddd---d---d---d---d--- group: start (ns) / finish (ns) / number of snapshots / Save requests: VDFs / phase planes /
    ionization rates / particles collided with walls

•             0.000         0.000       1   3   3   1   0

•             1.000         9.000       5   0   1   1   0

•            10.000        10.000       1   3   3   1   0
```

In the bottom of file init_snapshots.dat there is a useful comment explaining values of control flags:

```
### these are just explanations for the above ###
save VDFs (0/1/2/3 = No/1d/2d/1d+2d)
save phase planes (0/1/2/3 = No/e/i/e+i)
save ionization rates (0/1 = No/Yes)
save particles collided with walls (0/1/2/3 = No/e/i/e+i)
```

init_snapshots.dat, part 3

```
•   -------- Parameters for calculation of velocity distribution functions
•   --d-d--- Number of spatial boxes along the X / Y direction in a cluster (>=0)
•     2 2
•   --ddd--- Electrons, maximal velocity [in units of v_Te_ms]
•       6 12
•   --ddd--- Electrons, number of velocity bins per v_Te_ms
•      20
•   --ddd--- Ions, maximal velocity [in units of v_Te_ms*sqrt(me/Ms)]
•       8
•   --ddd--- Number of velocity bins per v_Te_ms*sqrt(me/Ms) for ions
•     100
```

Specify how cluster’s subdomain is split into boxes. If 1 and 1 are chosen here, the whole cluster’s subdomain is treated as one box. If any or both of these numbers is/are zero, VDFs are NOT created.

Define velocity range and velocity bin number/size for electrons.

Define velocity range and velocity bin number/size for ions.

The code calculates velocity distribution functions as follows.

- A cluster subdomain is split into several equal rectangular spatial boxes, for example into 2x2=4 boxes as in the exemplary file above.
- The velocity range is also split into a number of velocity bins. For example the electron velocity range above is from -6$V_{th,e}$ to 6$V_{th,e}$ where $V_{th,e}$ is the electron thermal velocity for the scale temperature, and has 20 bins per $V_{th,e}$. And the ion velocity range is from -8$V_{th,i}$ to 8$V_{th,i}$ where $V_{th,i}$ is the ion thermal velocity for the scale temperature, with 100 bins per $V_{th,i}$.
- Then all particles which are inside one spatial box are distributed into velocity bins according to their velocities along the x, y, and z directions.

The VDF values saved by the code are the numbers of macroparticles from the corresponding spatial box in each velocity bin. The code saves 1d VDFs for electrons and ions and 2d VDFs for electrons.

init_snapshots.dat, part 4

```
•   -------- Parameters for saving phase planes
•   --dd---- Number of rectangular spatial boxes (>=0)
•      2
•   --dddd--dddd----dddd--dddd--- left bottom corner X/Y (node index) / right top
    corner X/Y (node index)
•      119      0      121    481
•        0    240      241    242
```

Phase planes {x,y,vx,vy,vz,tag} are saved for particles inside pre-defined spatial boxes. Here one declares the number of these boxes.

Here one specifies left bottom and right top corners of each spatial box. The number of lines must be no less than the number of boxes declared above, otherwise the code may give an error.

init_simcontrol.dat

```
---ddddddd.ddd----- simulation time [ns]
        10.010
--------dd--------- number of electron sub-cycles per ion cycle (odd)
        11
------dddd--------- number of ion cycles between internal cluster load balancing events
        10
------dddd--------- number of internal cluster load balancing events between global load balancing events
        10
---ddddddd--------- number of ion cycles between checkpoints (no checkpoints if 0, MPIIO if >0, POSIX if
<0)
     10000
---------d--------- use checkpoint (2/1/0 = Yes, to start/Yes, to continue/No)
         0
--dddddddd--------- time step when the checkpoint to be used was saved (dim-less)
         0
```

Set simulation time [ns].

Set the number of electron time steps between ion updates, must be an odd number. If ions move at each electron step, set 1.

Define interval between **internal** PLB events.

Define interval between **global** PLB events.

Define interval between creating checkpoints.

- Zero means no checkpoints.
- A positive value means checkpoints will be saved using MPIIO procedures. In this case, a single file Tcntr_TTTTTTTT.check is created which contains data for all MPI processes.
- A negative value means checkpoints will be saved using standard write Fortran commands. One file per process Tcntr_TTTTTTTT_proc_PPP.check is created and placed in folder checkdir_TTTTTTTT.
- In the above, TTTTTTTT is the time step of checkpoint, PPP is the MPI rank of process.

Specify whether the simulation starts from the scratch (0) or continues from a checkpoint (1) or uses a checkpoint as an initial state (2) (in this case the step counter starts at zero).

When a checkpoint is created, the time step becomes part of the name of the checkpoint file. Here we specify which checkpoint will be used. In this particular case, the filename would be Tcntr_00000000.check for an MPIIO checkpoint and checkdir_00000000/T_cntr_00000000_proc_*.check for a POSIX checkpoint. The code tries to read from an MPIIO file first. If such a file does not exist, the code tries to read from POSIX checkpoint files.

An **internal** particle load balancing (PLB) event is when processes working on the same cluster exchange particles between each other to achieve approximately even particle load within the cluster.
An **external** PLB compares the particle load in different clusters and tries to make it even. The external PLB event is more expensive because it may reassign some processes to other clusters, and therefore it should be applied less frequently than the internal one.

petsc.rc

```
# -help
#-vec_view ascii:vector.dat
#-mat_view ascii:matrix.dat
-pc_type hypre
-ksp_type gmres
-ksp_rtol 1.e-9
-pc_monitor
-ksp_monitor
#-log_exclude snes
-log_view
# -on_error_attach_debugger gdb
```

This file is required by the PETSc based field solver. The file was provided by Salomon Janhunen.

Preconditioner type

Solver type

Relative convergence tolerance

Additional input files

- init_physconstants.dat
  - see https://github.com/PrincetonUniversity/EDIPIC-2D/tree/main/input_data_dc_semiperiodic_Helium_rcx
- init_neutral_AAAAAA_rcx_param.dat
  - see https://github.com/PrincetonUniversity/EDIPIC-2D/tree/main/input_data_dc_semiperiodic_Helium_rcx
- init_bo_NN_waveform.dat
  - see https://github.com/PrincetonUniversity/EDIPIC-2D/tree/main/input_data_files_waveform
- init_material_AAAAAA.dat
  - see https://github.com/PrincetonUniversity/EDIPIC-2D/tree/main/input_data_dc_inner_object_ion_ind_EE
- init_avgsnapshots.dat
  - see https://github.com/PrincetonUniversity/EDIPIC-2D/tree/main/input_data_dc_inner_object_avg_heat_flows
- init_ext_circuit.dat
  - see https://github.com/PrincetonUniversity/EDIPIC-2D/tree/main/input_data_rf_external_circuit

These files are optional. If the code does not find any of these files, either default values will be used or certain features will not be enabled.

init_physconstants.dat

```
•  the vacuum permittivity constant is in the line below (the true value is 8.854188e-12 F/m)
•      8.854188E-12
```

This file allows to change the vacuum permittivity constant ε_0. Increasing the constant reduces the plasma frequency and increases the simulation time step. This may be useful for simulation of quasi-stationary state of large systems during large time intervals if phenomena like turbulence or plasma waves are not the primary objective of interest.

If this file is not present, the default (true) value of the vacuum permittivity constant is taken.

Data processing program **dataproc_extract_bo_particle_fluxes_from_history_new.f90** calculates time step used in simulation the same way as the code does. It either takes the required value of ε_0 from file init_physconstants.dat or uses the true default value if the file is not available.

init_neutral_Helium_rcx_param.dat

```
• corresponsing ion species index, xsec in m2 at 1 eV, alpha (non-dim. const.)
•   1    2.79e-19  0.0557
```

- The general format for the name of this file is init_neutral_AAAAAA_rcx_param.dat, where AAAAAA is replaced with the name of the ion species (in this case Helium).
- This file contains parameters required by the model of resonance charge exchange collisions.
- If the file is not found, resonance charge exchange collisions are turned off.

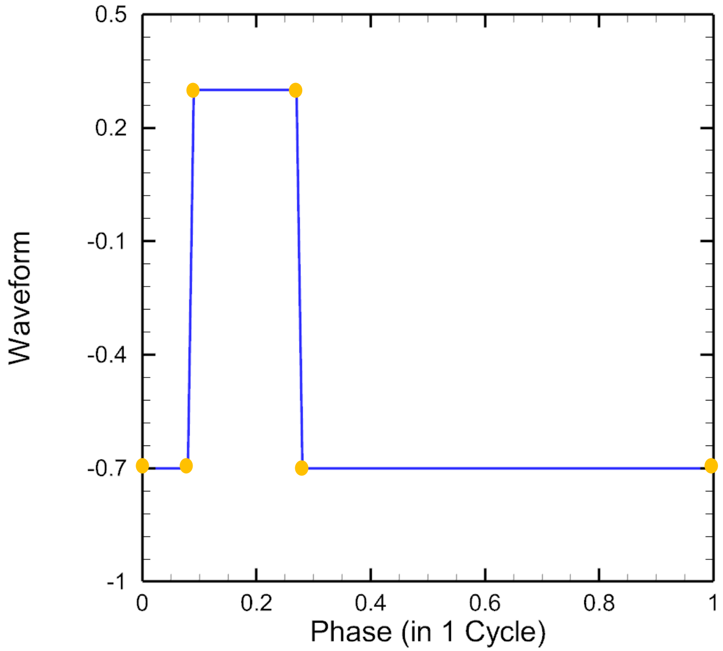
init_bo_NN_waveform.dat In the file name, NN is replaced with the number of the corresponding boundary object.

```
•   50.0  waveform potential amplitude V
•    1.0e8 waveform frequency Hz
•  # column 1 is time in units of the waveform period
•  # column 2 is value in units of the waveform amplitude
•    0.0   -0.7
•    0.08  -0.7
•    0.09  0.3
•    0.27  0.3
•    0.28  -0.7
•    1.0   -0.7
```

Define amplitude and frequency of the additional signal.

Define shape of the additional signal.

- This file allows to add a second signal on top of the existing electric boundary condition:
  $\phi(t) = \phi_{const} + \phi_{osc}\sin(\omega_{osc}t + \Phi) + \phi_{custom} * f(\omega_{custom}t)$
- In the example above, $f(\omega_{custom}t)$ is →
- Free format is used.



init_material_METAL2.dat (part 1)

```
------dd.ddd- dielectric constant
       1.000
=======d===== ELASTIC ELECTRON REFLECTION MODEL (0/1/2 = turned off/model 1/model 2)
       0
-------d----- ELASTIC ELECTRON REFLECTION TYPE (0/1 = specular/random)
       1
              ELASTIC ELECTRON REFLECTION, MODEL 1, PARAMETERS:
-------d.ddd- emission coefficient (from 0 to 1), constant for energies between E_min and E_max
       1.000
--dddddd.ddd- lower energy boundary E_min, [eV]
       0.000
--dddddd.ddd- upper energy boundary E_max, [eV]
   10000.000
              ELASTIC ELECTRON REFLECTION, MODEL 2, PARAMETERS:
--dddddd.ddd- threshold energy, [eV]
       2.000
--dddddd.ddd- energy of maximum of the emission coefficient [eV]
      10.000
-------d.ddd- maximum of the emission coefficient (from 0 to 1)
       0.550
--dddddd.ddd- energy scale of decaying part for energy above the coefficient maximum energy, [eV]
      14.000
-------d.ddd- additional fraction of the classic emission coefficient, (>=0, <<1)
       0.030
```

The general template for the name of this file is init_material_AAAAAA.dat . AAAAAA must be replaced with the name of material (METAL2 here). It corresponds to the material name used in file init_bo_NN.dat (line 2 from the top of the file).

This file specifies the dielectric constant (relevant only if the object is dielectric), secondary electron emission properties, ion-induced secondary electron emission properties.

If for a boundary object NN the corresponding material AAAAAA file is not found, default values are used: dielectric constant 1, all particles are absorbed, no secondary electron emission, no ion-induced secondary electron emission. The object still may do constant electron injection if it is requested in init_bo_NN.dat.

init_material_METAL2.dat (part 2)

```
=======d===== INELASTIC ELECTRON REFLECTION MODEL (0/1/2 = turned off/model 1/model 2)
       0 2
              INELASTIC ELECTRON REFLECTION, MODEL 1, PARAMETERS:
-------d.ddd- emission coefficient (from 0 to 1), constant for energies between E_min and E_max
       1.000
--dddddd.ddd- lower energy boundary E_min, [eV]
       0.000
--dddddd.ddd- upper energy boundary E_max, [eV]
   10000.000
              INELASTIC ELECTRON REFLECTION, MODEL 2, PARAMETERS:
-------d.ddd- fraction of the classic emission coefficient (>=0, <<1)
       0.070
=======d===== TRUE SECONDARY EMISSION MODEL (0/1/2 = turned off/model 1/model 2)
       0 2
--dddddd.ddd- temperature of true secondary electrons, [eV]
       2.000
              TRUE SECONDARY EMISSION, MODEL 1, PARAMETERS:
-------d.ddd- emission coefficient (>=0), constant for energies between E_min and E_max
       0.300
--dddddd.ddd- lower energy boundary E_min, [eV]
       4.000
--dddddd.ddd- upper energy boundary E_max, [eV]
   10000.000
              TRUE SECONDARY EMISSION, MODEL 2 (AND CLASSIC), PARAMETERS
--dddddd.ddd- threshold energy, [eV]
      13.000
--dddddd.ddd- energy of maximum of the emission coefficient, [eV]
     500.000
------dd.ddd- maximum of the emission coefficient (>0)
       3.000
-------d.ddd- Smoothness factor (0 = very rough, 2 = polished)
       1.000
```

Secondary electrons emitted due to bombardment of material surface by energetic electrons have several components: elastically reflected electrons, inelastically reflected electrons, and true secondary electrons. Each component can be turned on/off independently. The model of secondary electron emission is largely the same as used in EDIPIC-1D, see Dmytro Sydorenko’s PhD Thesis in https://github.com/PrincetonUniversity/EDIPIC-2D/tree/main/Doc .

init_material_METAL2.dat (part 3)

```
•   =======d===== ION-MATERIAL INTERACTION MODEL (0/1/2 = 100% ion adsorption/100% specular reflection/ion-induced electron emission)
•          2
•   ------------- ION-INDUCED ELECTRON EMISSION, ION SPECIES 1, PARAMETERS:
•   -------d.ddd- emission coefficient (>=0), constant for energies between E_min and E_max
•          0.200
•   --dddddd.ddd- lower energy boundary E_min, [eV]
•        100.000
•   --dddddd.ddd- upper energy boundary E_max, [eV]
•      10000.000
•   --dddddd.ddd- temperature of emitted electrons, [eV]
•          2.000
```

This section defines properties of ion interaction with the material surface. One can select between complete absorption of ions, specular reflection of ions, or absorption of ions combined with emission of secondary electrons.

Note, if one needs a boundary where both electrons and ions are reflected, one has to set this flag to 1, disable inelastically reflected and true secondary electrons (set on/off flags in the previous slide to 0, as they are now), turn on elastic electron reflection model 1, and select specular electron reflection (see slide 30).

init_avgsnapshots.dat

```
•  --- save 2D maps of the following TIME-AVERAGED parameters? (1=yes, 0=no)
•  -----F----EX----EY--JXsum--JYsum--JZsum   (type flag values below)
•       1    1     1   0      0      0
•  ----Ne----JXe---JYe---JZe---VXe---VYe---VZe---WXe---WYe---WZe---TXe---TYe---TZe---QXe---QYe---QZe  (type flag values below)
•      1     1     1     0     1     1     0     1     1     0     1     1     0     1     1     0
•  ----Ni----JXi---JYi---JZi---VXi---VYi---VZi---WXi---WYi---WZi---TXi---TYi---TZi---QXi---QYi---QZi  (type flag values below)
•      1     1     1     0     1     1     0     1     1     0     1     1     0     1     1     0
•  --- number of groups of snapshots (>0, no averaged snapshots if <=0), type below
•     1
•  --- For each group, type below start (ns), finish (ns), number of snapshots (>=0)
•    10.0    50.0    4
```

Select parameters to be saved.

Specify when snapshots must be created.

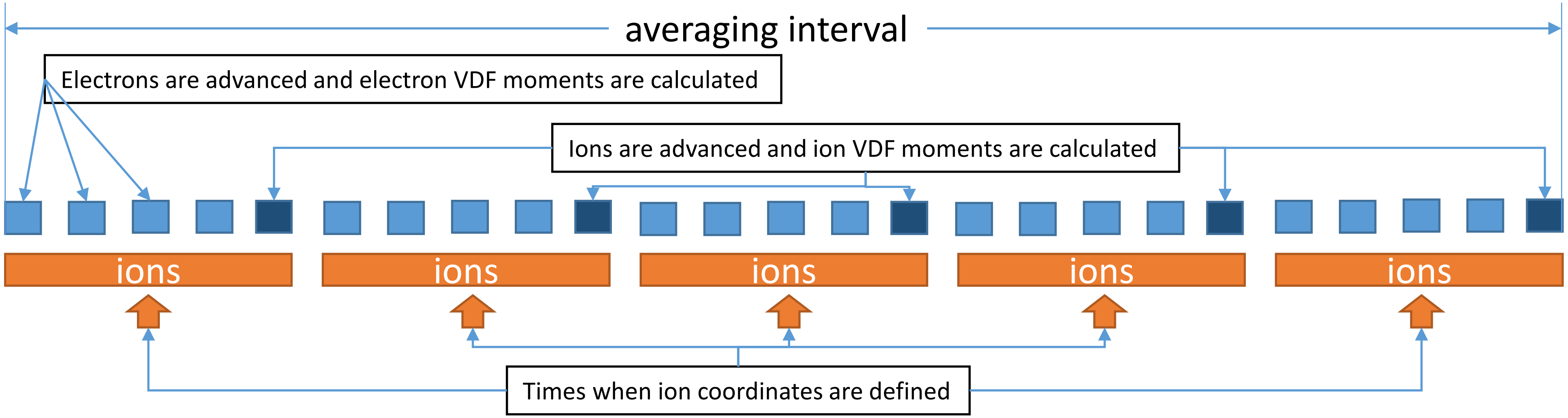
In this particular case, 4 time-averaged snapshots will be created:

- snapshot 1 at time about 20 ns with values averaged over interval 10 ns to 20 ns,
- snapshot 2 at time about 30 ns with values averaged over interval 20 ns to 30 ns,
- snapshot 3 at time about 40 ns with values averaged over interval 30 ns to 40 ns,
- snapshot 4 at time about 50 ns with values averaged over interval 40 ns to 50 ns.

Beginning and end of averaging interval are adjusted in order to ensure that electron and ion moments are averaged over the same interval. The averaging begins at the time step that immediately follows ion advance (ion velocity are updated then ion coordinates are updated). The averaging ends at the time step of ion advance.

Time-averaged snapshots reduce noise!

Time-averaged snapshots are created independently on the ordinary snapshots.



averaging interval

Electrons are advanced and electron VDF moments are calculated

Ions are advanced and ion VDF moments are calculated

Times when ion coordinates are defined

init_ext_circuit.dat

```
• total number of electrodes whose potential must be solved (>0, if <=0 then no external circuit)
•    1
• below list in one column numbers of electrodes whose potential must be solved
•    1
• below is amplitude of potential oscillations in the driving rf voltage source [V]
•    100.0
• below is frequency of potential oscillations in the driving rf voltage source [Hz]
•    2.5E+07
• below is phase of potential oscillations in the driving rf voltage source [deg]
•    0.0
• below is capacitance of capacitor [F]
•    1.0E-09
```

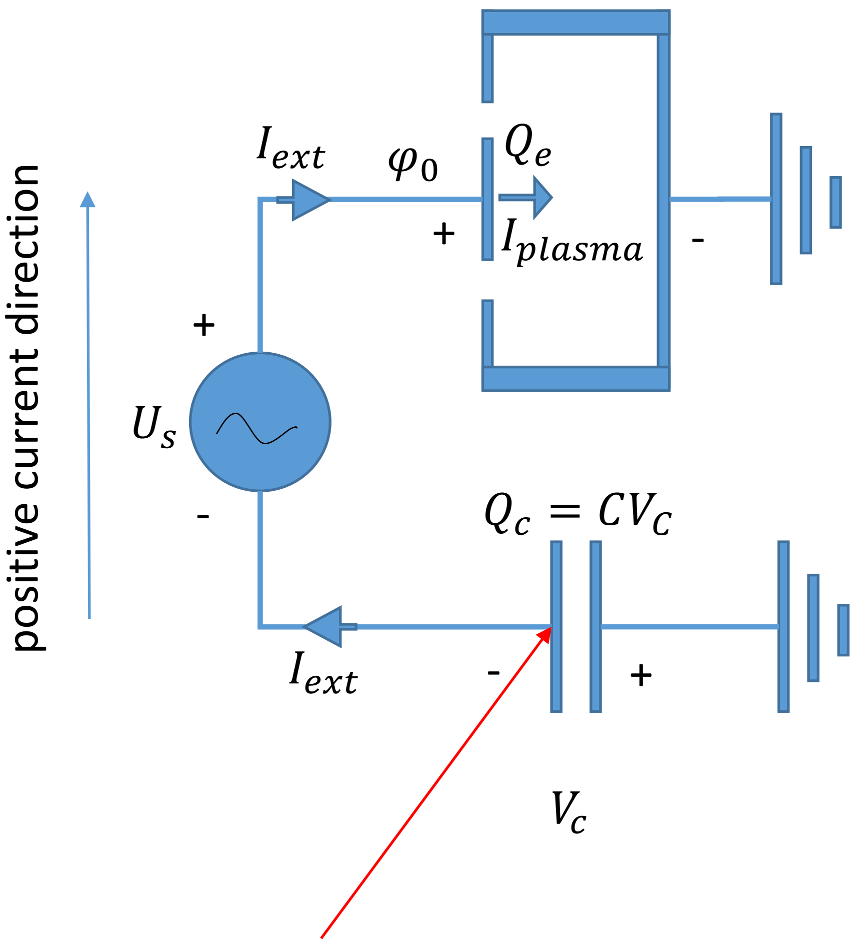
Give the number of electrodes connected to the external circuit whose potential must be solved. Do not include electrodes whose potentials are known (e.g. grounded or a given function of time).

List here boundary object numbers for all electrodes whose potential must be solved.

Give here parameters of the contour shown in the next slide. For a different contour, the list of parameters will be different as well.

This file is necessary to specify parameters of the external circuit. If the file is not found, the code proceeds as if no external circuit exists.

Rf discharge with a capacitor



Potential of this point is $-V_c$

- External circuit equation connects potential of an electrode and the variation of charge on this electrode during the time step which is due to
  - plasma particles hitting the electrode,
  - emission from electrode,
  - electric current in the external circuit.

$$\left.\begin{aligned} U_s &= V_C + \varphi_0 \\ I_{ext} &= \frac{dQ_C}{dt} = C\frac{dV_C}{dt} \\ \frac{dQ_e}{dt} &= I_{ext} - I_{plasma} \end{aligned}\right\}$$

Total charge on the electrode

$$\frac{d}{dt}\varphi_0 = \frac{d}{dt}U_s - \frac{1}{C}\left(\frac{dQ_e}{dt} + I_{plasma}\right)$$

$$I_{plasma} = \frac{|\Delta Q^-_{coll}|}{\Delta t} - \frac{|\Delta Q^+_{coll}|}{\Delta t} - \frac{|\Delta Q^-_{emit}|}{\Delta t}$$

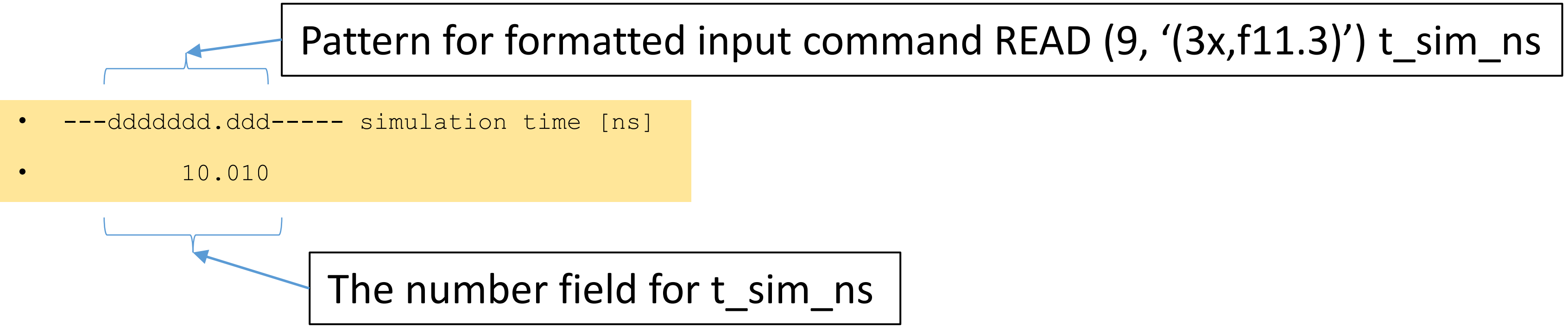
Absolute values of charge of:

- electrons collided with the electrode,
- ions collided with the electrode,
- electrons emitted by the electrode

during one time step.

Some general comments

- This is not a final version, the code is in the development process, and the data files may change.
- Presently the main simulation domain is a rectangle. Inner metal or dielectric rectangular objects can be placed inside the main domain.
- The code uses formatted input for many input files, so it is important to follow patterns given above the number fields (if there is one), otherwise an error may occur.



Pattern for formatted input command READ (9, ‘(3x,f11.3)’) t_sim_ns

```
•   ---ddddddd.ddd----- simulation time [ns]
•           10.010
```

The number field for t_sim_ns

Files which use free format (as of 19-03-2022): init_probes.dat, init_physconstants.dat, init_neutral_Helium_rcx_param.dat, init_avgsnapshots.dat, init_ext_circuit.dat, init_bo_NN_waveform.dat .