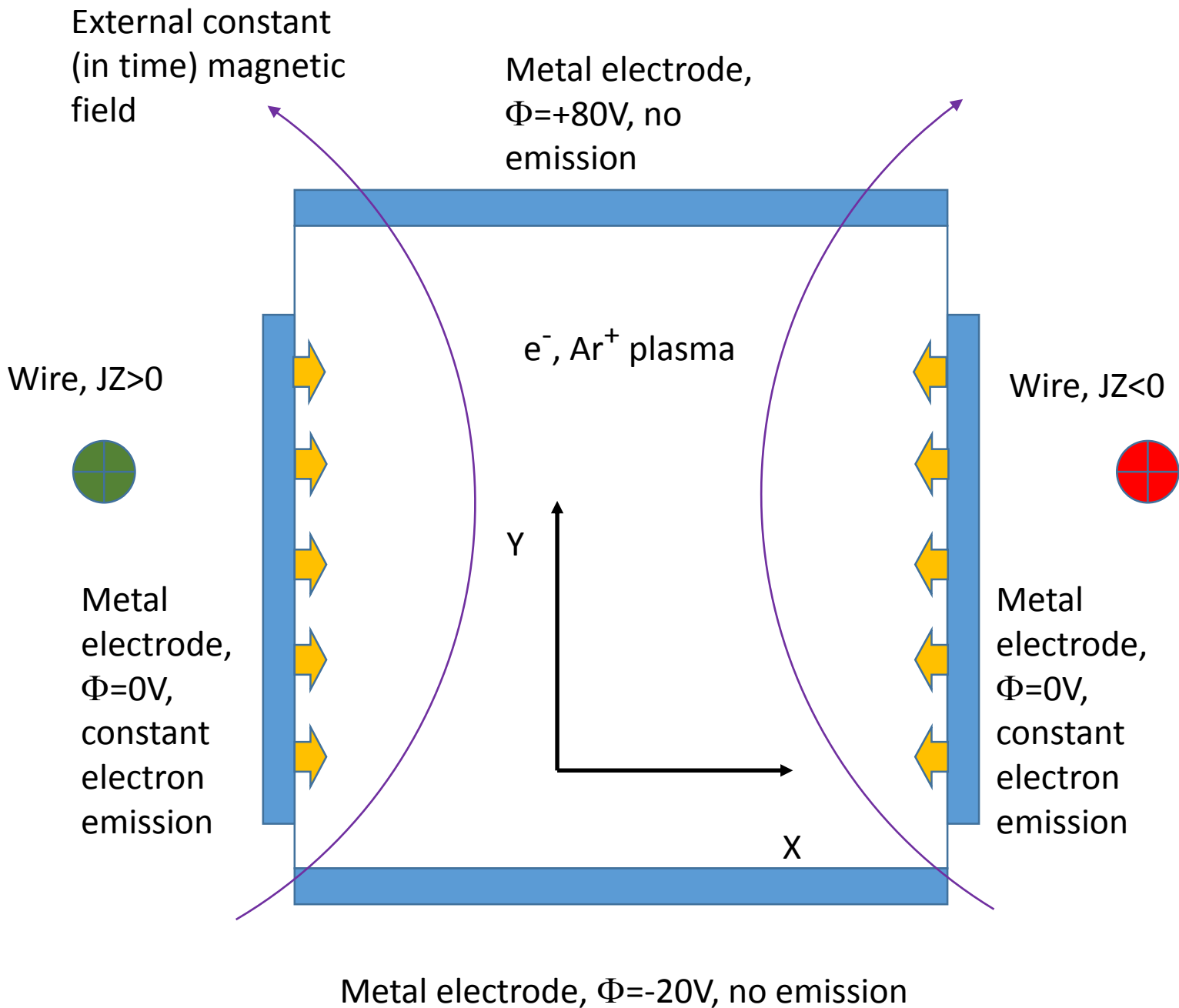


EDIPIC-2D input data description

Dmytro Sydorenko



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 $1e16 \text{ m}^{-3}$, 20 particles per cell;
- left and right electrodes ($0.67\text{mm}<y<13.33\text{mm}$) emit 100 particles per timestep each;
- vacuum gap boundaries between the electrodes;
- magnetic field is created by two opposite JZ currents at $x=-10\text{mm}, y=8\text{mm}$ and $x=18\text{mm}, y=8\text{mm}$;
- Ions and neutrals are Argon;
- neutral density $1e21\text{m}^{-3}$;
- 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`

init_configuration.dat, part 1

```
• ---dddd.dddddddd----- scale electron temperature [eV]
•      4.0000000
• ---+d.ddddddddE+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 $\Delta x = \lambda_{D, \text{scale}} / N_D$. Initial electron density/temperature do not have to be equal to the scale values.

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

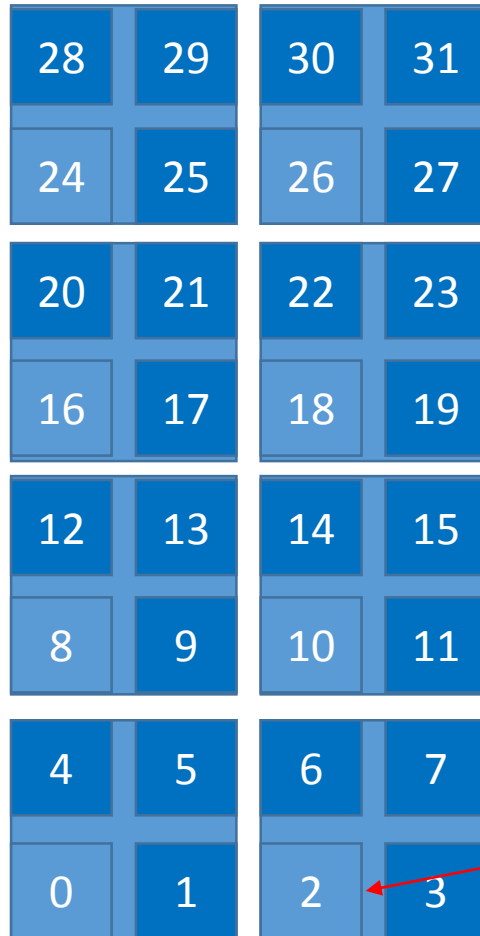
Define splitting of the simulation domain between all MPI processes. Presently this splitting is used in the PETSc field solver. Works for a rectangular simulation domain. See also the next page.

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



“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.

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).

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

init_configuration.dat, part 3

```
• ---ddd----- number of boundary objects
•      8
• ===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. The code expects to find this number of full entries as described below, otherwise it gives an error.

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---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---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---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---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
```

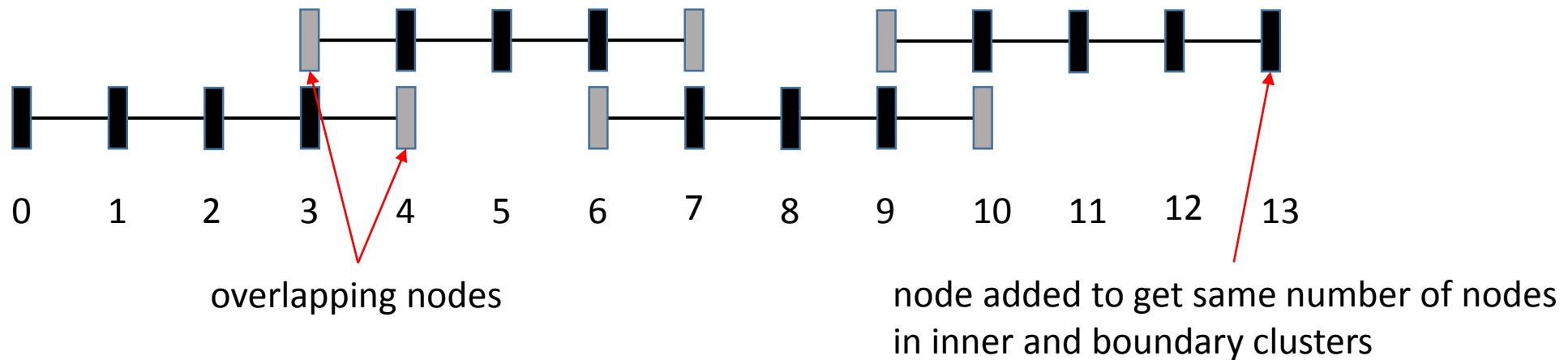
Presently, 5 types of boundary objects are possible (see the bottom of file `init_configuration.dat`):

- 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.

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

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 $4 \times 60 + 1 = 241$. Similar, in the Y direction, the range of domain node indices along the Y direction is from 0 to $8 \times 60 + 1 = 481$.
- Indexing begins with zero because it allows to find indices of the left bottom node of the cell containing a particle as $i = \text{int}(x)$, $j = \text{int}(y)$.
- 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:



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

```
• -----AAAAA--- code/abbreviation of the material, character string
•      METAL0
• ---dddd.ddd--- constant potential [V]
•      0.000
• ---dddd.ddd--- amplitude of potential oscillations [V]
•      0.000
• ---dddd.ddd--- frequency [MHz]
•      0.000
• ---dddd.ddd--- phase [deg] for sin(omega*t+phase)
•      0.000
• ----ddd----- number of electron macroparticles injected each timestep, constant [dim-less]
•      100
• -----d.ddd--- temperature of emitted electrons [eV]
•      0.200
```

Define material of the boundary. Reserved for future use. When implemented, selecting certain material will automatically specify properties like secondary electron emission.

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.

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.

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.

init_extfields.dat

```
• ---dddddd.ddd--- X-magnetic field [Gauss]
•
• 0.000
• ---dddddd.ddd--- Y-magnetic field [Gauss]
•
• 0.000
• ----- parameters of Z-magnetic field as used by Boeuf and Garrigues
• ---dddddd.ddd--- Y-coordinate of the BZ maximum, y_Bmax [cm]
•
• 0.750
• ---dddddd.ddd--- BZ at y=0 [Gauss]
•
• 0.000
• ---dddddd.ddd--- BZ at y_Bmax [Gauss]
•
• 0.000
• ---dddddd.ddd--- BZ at y=Lsys [Gauss]
•
• 0.000
• ---dddddd.ddd--- characteristic length of decay for y<y_Bmax [cm]
•
• 0.625 0.600
• ---dddddd.ddd--- characteristic length of decay for y>y_Bmax [cm]
•
• 0.625 0.600
• ---dddddd.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]
• ---dddddd.ddd---dddddd.ddd---dddddd.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
• -----dddd.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 reserved); 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
•
• --dddddd.ddd----- minimal energy [eV]
•
•      0.000
•
• --dddddd.ddd---ddd.ddd--- energy segment 1 :: upper boundary [eV] / resolution [eV]
•
•      0.040      0.005
•
• --dddddd.ddd---ddd.ddd--- energy segment 2 :: upper boundary [eV] / resolution [eV]
•
•      0.100      0.010
•
• --dddddd.ddd---ddd.ddd--- energy segment 3 :: upper boundary [eV] / resolution [eV]
•
•      0.400      0.020
•
• --dddddd.ddd---ddd.ddd--- energy segment 4 :: upper boundary [eV] / resolution [eV]
•
•      1.000      0.050
•
• --dddddd.ddd---ddd.ddd--- energy segment 5 :: upper boundary [eV] / resolution [eV]
•
•     10.000      0.200
•
• --dddddd.ddd---ddd.ddd--- energy segment 6 :: upper boundary [eV] / resolution [eV]
•
•     20.000      0.100
•
• --dddddd.ddd---ddd.ddd--- energy segment 7 :: upper boundary [eV] / resolution [eV]
•
•     60.000      0.500
•
• --dddddd.ddd---ddd.ddd--- energy segment 8 :: upper boundary [eV] / resolution [eV]
•
•    200.000      5.000
•
• --dddddd.ddd---ddd.ddd--- energy segment 9 :: upper boundary [eV] / resolution [eV]
•
•   1000.000     10.000
•
• --dddddd.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.

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).

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 in certain points called “probes”. This file controls how these data are saved.

```
• ----dddddd--- Step for saving (timesteps, >=1)
•
•           1
• ---ddddddddd--- 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
• --dddddd--dddddd-- 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 step is 1, data are saved at each time step, 2 – every other time step, 3 – every 3rd 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.

Presently not used, reserved for future use.

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
• -----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
• -----Ni-----JXi---JYi---JZi---VXi---VYi---VZi---WXi---WYi---WZi---TXi---TYi---TZi
• -----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
```

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 along the x, y, and z directions, temperatures along the x, y, and z directions.

Same as above but for **ions**.

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

init_snapshots.dat, part 2

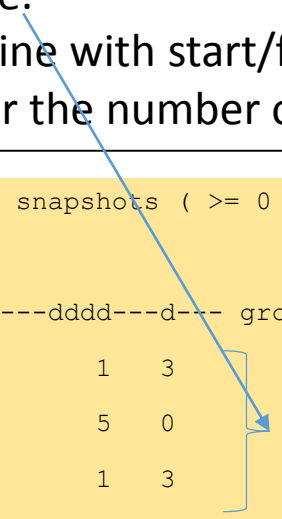
It may be useful to save snapshots with different frequency (i.e. interval between snapshots) at different stages of a simulation. Snapshots are combined into groups, each group has:

- 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,
- and a flag which controls saving of the velocity distribution functions (VDFs).

The group controls are here.

There must be a separate line with start/finish/number-of-snapshots/evdf-flag 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--- group: start (ns) / finish (ns) / number of snapshots / save VDFs (0/1/2/3 = No/1d only/2d only/1d and 2d)
•      0.000      0.000      1  3
•      1.000      9.000      5  0
•      10.000     10.000      1  3
```



The flag which controls saving the electron and ion VDFs in a snapshot group has been added for the following reason. The VDF files may be quite large, especially for distributions over two velocity components. To reduce the size of the data produced in a simulation, one can choose which VDFs (if any) will be saved in each group of snapshots. In the example above, both 1d and 2d VDFs are created for the first and the last groups, while the second group of snapshots is created without any VDFs.

- ----- 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 $v_{Te_ms} \cdot \sqrt{m_e/m_i}$]
- 8
- --ddd--- Number of velocity bins per $v_{Te_ms} \cdot \sqrt{m_e/m_i}$ for ions
- 100

init_snapshots.dat, part 3

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 $2 \times 2 = 4$ 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 $-6V_{th,e}$ to $6V_{th,e}$ 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 $-8V_{th,i}$ to $8V_{th,i}$ 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_simcontrol.dat

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.

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 .

```
• ---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)
•          10000
• -----d----- use checkpoint (2/1/0 = Yes, to start/Yes, to continue/No)
•          0
• --ddddddd----- time step when the checkpoint to be used was saved (dim-less)
•          0
```

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.

Some general comments

- This is not a final version, the code is in the development process, and the data files may change.
- Presently the simulation domain is a rectangle only.
- File `init_configuration.dat` is prepared for a certain number of MPI processes (32 in this example). If the number of MPI processes is about to change (even if the size of the whole domain remains the same), one has to modify this file accordingly. In addition, this may affect the size of spatial boxes used for calculation of particle velocity distribution functions, so editing of `init_snapshots.dat` may also be necessary.
- The code uses formatted input, so it is important to follow patterns given above the number fields, otherwise an error may (most likely) occur.

