MCBC

Monte Carlo Charge Breeding Code

Version 1.0

FAR-TECH, Inc.
3550 General Atomics Ct, MS 15-155
San Diego CA 92121
Phone 858-455-6655
Email support@far-tech.com
URL http//far-tech.com

Introduction

MCBC is a Monte Carlo particle-tracking code, written in Fortran, which runs under Unix/Linux systems. It simulates the creation of higher charge-state ions by injecting a +1 ion beam into an ECRIS plasma. MCBC can follow the trajectories of the ions and simulate Coulomb collisions, ionizations, and charge exchanges with the background plasma. There are two operational modes in MCBC. First, MCBC can predict the profile of captured ions in the given ECRIS plasma by setting *Steady_state* as false. For this code, we define “capture” as follows: when the energy of a test ion is less than the temperature of the background plasma ions, we call it "captured". Also MCBC can track the injected ions until they are lost or exit from the device if *Steady_state* is true.

Getting Started

Unix/Linux Installation

Place the `MCBC2010.zip` file in a directory of your choosing and type the following commands at the prompt:

```
unzip MCBC2010.zip
cd MCBC2010
chmod +x mcbc mcbc32 mcbc_mpi mcbc_mpi32
```

You can delete the `MCBC2010.zip` file at this point, if desired.

In order to take advantage of all the features of MCBC, the following packages should be installed on your system:

Package	***Purpose***
NetCDF	required; saving data
OpenGL	optional; plotting results
MATLAB *	optional; plotting results

* The plotting scripts work using MATLAB R2007b with the external MEXNC (NetCDF toolbox) package installed. They may also work on other versions of MATLAB, but MEXNC is needed.

Program Execution

To run MCBC on a single processor on a 64-bit machine, type:

```
./mcbc
```

Running MCBC on multiple processors is possible, but still in the testing stage. Future versions of the code will include a more thoroughly tested multi-processor capability; for now, it is not recommended that you use this capability. To run MCBC on multiple

processors, you must have the MPI environment set up. Then edit the file `linux.machines`. This file lists the node names which will be used in the run. They should be modified to correspond to the node names for your system. For example, to start the simulation on 3 processors, issue the command:

```
mpirun -machinefile linux.machines –np 3 mcbc_mpi
```

Units Used in MCBC

SI units are used unless specified otherwise:

time……second
length……..meter
current……Ampere
energy……..eV
potential……...Volts
magnetic field……..Tesla
charge……..Coulomb

MCBC Input

The general input parameters for MCBC are specified in the file `mcbc.in`. In addition, the 3D background plasma parameters, magnetic field, and electrical field must be provided.

The following tables list the input parameters in the file `mcbc.in`. For many runs, only a few of the default values for these parameters may need to be modified.

General Parameters

Variable Name	Type	Description
number_ions_injected	integer	Number of injected beam ions
number_ions_per_cpu	integer	Number of ions per cpu for each job sent when parallel computation is used
timestep_max	integer	Maximum number of time steps allowed
beam_energy	real	Beam kinetic energy
ngrid_x	integer	Number of x grid points
ngrid_y	integer	Number of y grid points
ngrid_z	integer	Number of z grid points
plasma_data_flag	flag	Flag for plasma data -1 = Internally created (see Plasma

		Parameters, below) 0 = No plasma 1 = Read plasma data from *.data files 2 = Read plasma data from the NetCDF file GEM_data (see below)
use_maxwellian_edf	logic	Flag for electron distribution function (EDF) .T. = Use Maxwellian EDF .F. = Use GEM EDF (non-Maxwellian)
ipush	flag	Flag for ODE solver 1 = Boris Leap Frog 2 = 2nd order Runge Kutta (RK) 3 = 4th order Runge Kutta (RK)
Steady_state	flag	Flag for MCBC operation mode .T. = ions are tracked until they are lost or exit the device .F.= ions capture mode
GEM_data	text	Filename for plasma background parameters calculated by GEM, in netCDF format
MCBC_output_FILE	text	MCBC output file for GEM, in netCDF format
nbins_r	integer	Number of bins in r direction for captured ions
nbins_z	integer	Number of bins in z direction for captured ions
Nsave	integer	Save results (during the run) after this number of ions have been injected
Nprint	integer	Write screen output after this number of ions have been injected
timestep_fraction	real	Used to adjust the timestep

ECR Device Parameters

Variable Name	Type	Description
domain_x	real	Computational x-domain
domain_y	real	Computational y-domain
domain_z	real	Computational z-domain
device_position_z	real	Starting z-position of ECRIS chamber
device_length	real	Length of ECRIS device
device_radius	real	Radius of ECRIS device
injection_aperture_radius	real	Injection aperture radius
beam_center_x	real	Beam center position
beam_center_y	real	Beam center position
beam_center_z	real	Beam center position
b_field_flag *	flag	Flag for generating the magnetic field -1 Generate field using experimental field data and save it to magnetic.nc (in NetCDF format) 0 No magnetic field 1 Read field from a *.data file 2 Read field from the file magnetic.nc (in

		NetCDF format) 3 Use polynomial fitting parameters to generate magnetic field
scale_factor_b_field	real	Multiplier/scaling factor for the magnetic field
axial_field	text	Filename for axial field data (see Note, below)
poly_deg	integer	0 No mirror field [positive integer] Polynomial fitting degree for mirror field
poly_f **	array	Coefficients for polynomial fitting
radial_field	text	Filename for radial field data (see Note, below)
multipole_order	flag	Order of the multipole field. 0 No multipole field 2 Quadrupole 3 Hexapole
e_field_flag	flag	Flag for generating electric field profile 0 No E field 1 Read E field from a *.data file 2 E Field read from the NetCDF file GEM_data (see above)
scale_factor_e_field	real	Multiplier/scaling factor for the electric field

* If `b_field_flag = 0`, `1`, or `2`, then the parameters `axial_field`, `poly_deg`, `poly_f`, `radial_field`, and `multipole_order` are not used.
If `b_field_flag = -1`, the parameters `axial_field`, `poly_deg`, `radial_field`, and `multipole_order` are used; `poly_f` is not.
If `b_field_flag = 3`, then `axial_field`, `poly_deg`, `poly_f`, `radial_field`, and `multipole_order` are all used.

** You can generate these polynomial fitting coefficients using MATLAB, which may provide a better fitting than MCBC does (with `b_field_flag = -1`). To do so, use the built-in MATLAB function:

```
[P, S, mu] = polyfit(z, Bz, N)
```

`z` and `Bz` are the measured values. `N` is the polynomial degree (`poly_deg` in MCBC). `P` contains the polynomial coefficients (`poly_f` in MCBC).

Note: The format for the `axial_field` file is as follows:

n
z_1 Bz_1
z_2 Bz_2
⋮
z_n Bz_n

The format for the `radial_field` file is identical, using `r` and `Br` instead of `z` and `Bz`.

Plasma Parameters

The following are used only if plasma_data_flag = -1:

Variable Name	Type	Description

temp_e	real	Electron temperature
den_e	real	Electron density
temp_i	real	Ion temperature
charge_eff *	real	Effective charge of background plasma
den_n	real	Neutral density

* Effective charge state of the background plasma. During Coulomb collisions, the test beam ion should collide with every ion charge state of the background plasma but to simplify the problem, we only let it collide with one effective ion which has effective charge `zeff = sum(qj*qj*nij)/sum(qj*nij) , j=1,2,.....A`.

Other plasma parameters:

Variable Name	Type	Description
charge_i	integer	Ion charge number (Not used)
mass_i	real	Ion mass number of background plasma
temp_n	real	Neutral temperature (Not used)
mass_n	integer	Neutral mass number
ion_atomic_number	integer	Atomic number of background plasma ions

Beam Parameters

Variable Name	Type	Description
temp_s	real	Default value for source temperature
mass_s	real	Source ion mass
charge_s	real	Source ion charge state at the beginning
beam_atomic_number	integer	Atomic Number of beam ions
beam_angular_divergence	real	Beam angular spread in mrad
beam_current	real	Ion beam current

Miscellaneous Parameters

Variable Name	Type	Description
output_ion_positions	logic	Flag for saving ion positions at the given times .F. do not record ion positions .T. record final ion positions
output_trace	logic	Flag for saving the ion trajectories .F. don't trace ions .T. trace ion trajectories
number_of_trace_ions	integer	Number of ion trajectories that are saved
trace_timestep_maximum	integer	Maximum number of time steps allowed for tracing trajectories

snapshot_time_number	integer	Number of elements in snapshot_time_array
snapshot_time_array	array	The times to take snapshots

To specify the 3D background plasma parameters, magnetic field, and electrical field, you have three options. You select which option to use by setting the `plasma_data_flag`, `b_field_flag`, and `e_field_flag` parameters in `mcbc.in` (see the above descriptions in the *General Parameters* and *ECR Device Parameters* tables).

1. Import these values from GEM.
   GEM.sav is an example NetCDF output file created using GEM1D.

2. Have MCBC create a uniform plasma.
   For this option, you must specify the appropriate parameters in `mcbc.in`.

3. Read the values from *.data files. The following table lists the required file names for the 3D background plasma parameters, 3D magnetic field, and 3D electrical field.

Parameter	***Variable Name***	***File Name***
electron density	ne(i,j,k)	density_e.data
ion density	ni(i,j,k)	density_i.data
electron temperature	Te(i,j,k)	temperature_e.data
ion temperature	Ti(i,j,k)	temperature_i.data
effective charge *	zeff(i,j,k)	zeff.data
magnetic field	bx(i,j,k), by(i,j,k), bz(i,j,k)	m_field.data
electric field	ex(i,j,k), ey(i,j,k), ez(i,j,k)	e_field.data

* See the note for `charge_eff` in the *Plasma Parameters* table, above.

Sample *.data files are included in the distribution. Your *.data files must follow the same format as these example files. The size of the data matrixes is (ngrid_x+1,ngrid_y+1,ngrid_z+1).

The following code describes how the magnetic field and electron density are read into the program. The electric field is read in the same way as the magnetic field; all other variables are read in the same way as the electron density.

Magnetic Field

```
DO i=1,ngrid_x+1
    DO j=1,ngrid_y+1
        DO k=1,ngrid_z+1
            read(80,*) bx(i,j,k),by(i,j,k),bz(i,j,k)
        ENDDO
    ENDDO
ENDDO
```

Electron Density

```
DO i=1,ngrid_x+1
    DO j=1,ngrid_y+1
        DO k=1,ngrid_z+1
            read(80,*) n_e(i,j,k)
        ENDDO
    ENDDO
ENDDO
```

MCBC Output

As the simulation runs, MCBC displays updates on the screen. The following is a sample print out:

```
-------------------------------------------------------------
  Nion =   2000,   Time = 0.145E-03    I_time=     123

  Lost = 13.20%,   Exit = 13.10%,   Back =  0.00%
  Trap = 73.15%,   Move =  0.00%,    Neu =  0.55%
  1+ = 33.55%,     2+ = 34.40%,     3+ =  4.95%
  4+ =  0.15%,     5+ =  0.00%,     6+ =  0.00%

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

On the first line, the number of injected ions, moving (tracing) time of the last ion, and number of time steps traced for the last ion are displayed. The second and third lines consist of the percentages of ions that are lost to the wall, exit to the extraction end, come back to the injection end, are captured (trapped) in the plasma, are still moving in the plasma when the maximum time step is reached, and are neutralized. Lines four and five show the percentages of the ions that are in these charge states when they are captured.

The output from MCBC consists of the profile of captured test ions. MCBC generates the following output files:

MCBC.log
This text file provides a simple summary of the MCBC run.

MCBC_output.sav
This NetCDF file is used to save beam capture profiles and diagnostic data for plotting and for future calculations using GEM. `MCBC_output.sav` is the main output file for MCBC.

To view the contents of the NetCDF file as text, type:

```
ncdump MCBC_output.sav > MCBC_output.txt
```

Then use a text editor to view `MCBC_output.txt`.

magnetic.nc

This NetCDF file contains the magnetic field from the last run. The file `MCBC_output.sav` also includes the magnetic field data, but `magnetic.nc` is provided for using the magnetic field separately.

In addition, MCBC includes two tools for plotting the results: an OpenGL tool and a MATLAB script. The OpenGL plotting tool is meant for quickly checking the simulation results; the MATLAB script provides more extensive plotting capabilities. If OpenGL is installed on your system, you can view basic plots of MCBC's input and output by issuing the following command:

```
./plot_tool.dat
```

This tool plots the values contained in the file `plot_input.txt`: plasma profiles (input) and charge state distribution on axis (output).

If MATLAB is available on your system, you can use the provided MATLAB script (`mcbcs.m`) to read and plot the data in `MCBC_output.sav`. To generate the MATLAB plots, issue the following command at the MATLAB prompt:

```
mcbcs
```

Note that all the other MATLAB scripts which are included with the distribution (`find_seg.m`, `pathdef.m`, `readnc.m`, and `FundConst.m`) are auxiliary functions which are used by `mcbcs.m`.

The MATLAB mcbcs function creates plots of the input and output for MCBC. This includes plots of the background plasma and magnetic field profiles (input) as well as the profiles of the captured ions (output). Additional output plots can be generated through the parameters listed in the *Miscellaneous Parameters* table (above). Sample MATLAB output plots are provided in the following section.

An Example Run Using MCBC

The MCBC package also includes an example simulation of the charge breeding of a 1+ rubidium ion beam in an oxygen plasma. (Simply issuing the command `./mcbc` without altering any of the distribution files will run this example.) In this section we describe the setup and MCBC results for the example. All input and output plots shown here were generated using the MATLAB `mcbcs` function.

Simulation Geometry and Background ECRIS Plasma

The simulated ECRIS device is 0.29m in length and 0.08m in diameter. The MCBC simulation domain ranges from 0 to 0.35m in the z direction and 0 to 0.08m in the x and y directions (see Figure 1). The ECRIS device (the shaded area in Figure 1) is placed at z=0.03m. The background oxygen plasma is calculated by GEM1D using the parameters listed in Table 1 (below). The 3D magnetic field is fitted from given radial

and axial measurements. In this example, the polynomial fitting is done in MATLAB. The polynomial coefficients are transferred to MCBC through the input parameter *poly_f*.

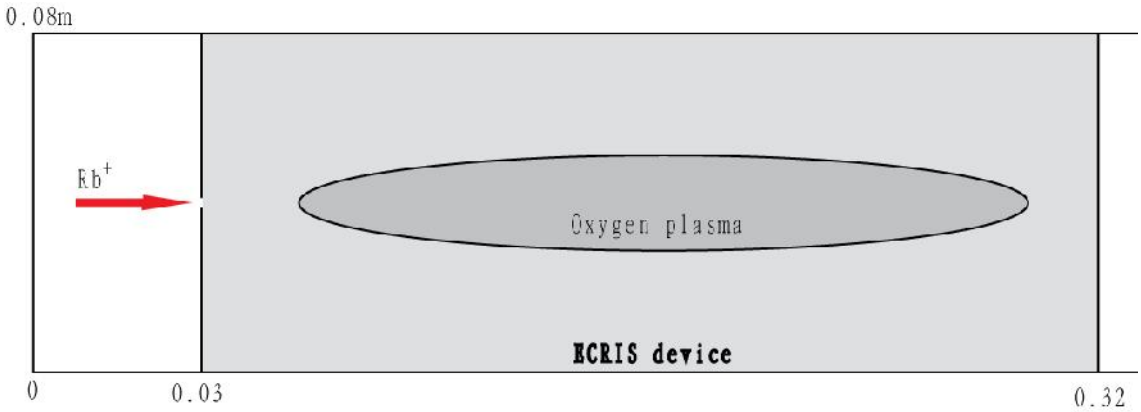

Figure 1. MCBC simulation domain and ECRIS device.

TABLE 1. Main simulation parameters of GEM 1D.

Parameters	Values
rf power (W)	450
rf frequency (GHz)	10.00
Supporting plasma	oxygen
Gas pressure (Torr)	1.2×10^{-7}
Length (cm)	29
Radius (cm)	4
$B_{injection}/B_{min}$	4.5
$B_{extraction}/B_{min}$	3.0

The plasma profile calculated by GEM1D is plotted in Figure 2. Since it provides only a 1D profile, MCBC will automatically extend it onto the 3D simulation grid space.

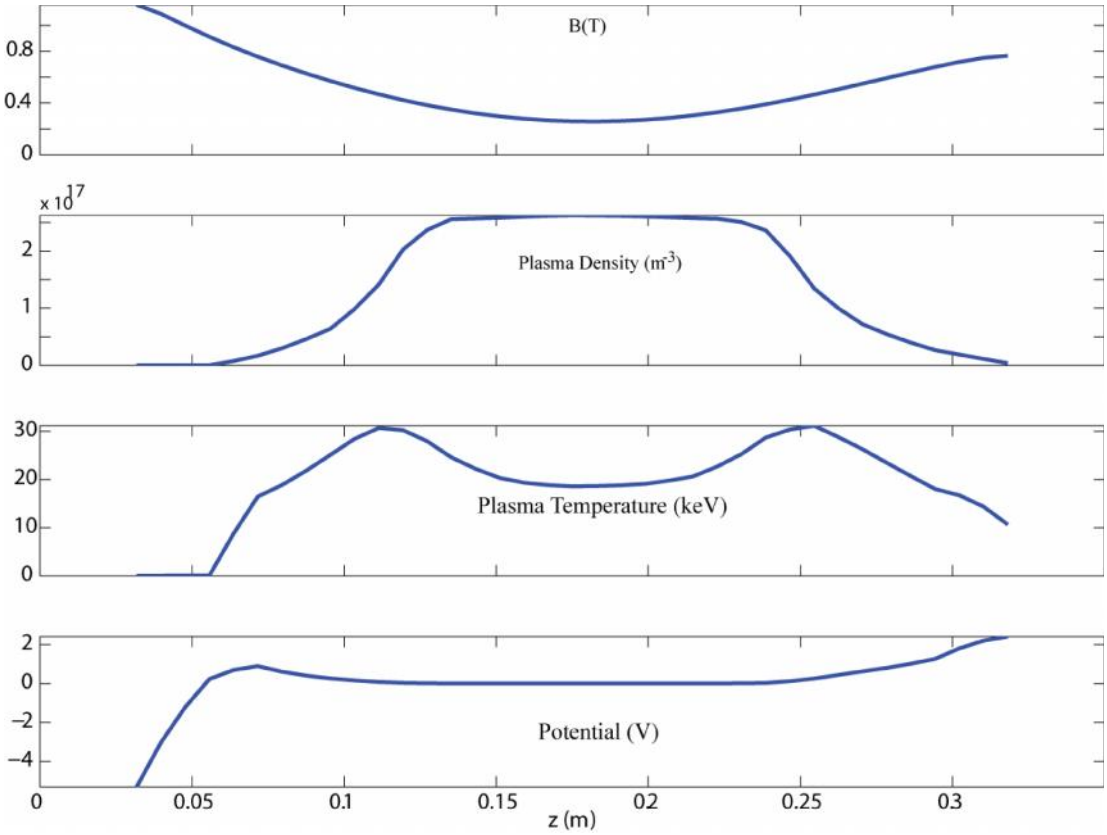

Figure 2: Background oxygen plasma and axial magnetic field profiles calculated by GEM1D.

MCBC Results

Once the plasma profiles are calculated, MCBC is used to simulate the injection and trapping of the rubidium beam. All Rb+ particles are started at the z = 0.03m axial location (see Figure 1), where the axial magnetic field is at a maximum (Figure 2). The beam is assumed to be mono-energetic, and distributed uniformly on the 1mm beam radius.

Each particle is tracked until it is “captured” or hits the wall, with “capture” being defined as having energy less than the temperature of the oxygen ions, which is assumed to be 1eV. The locations at which the beam ions are captured are then typically grouped into 2D bins. However, for this example case, only 1D axial bins are used. The profile of the captured rubidium ions is shown in Figure 3. The efficiency is calculated by dividing the number of captured ions with each charge state by the total number of injected rubidium ions, which is 5,000 in this example.

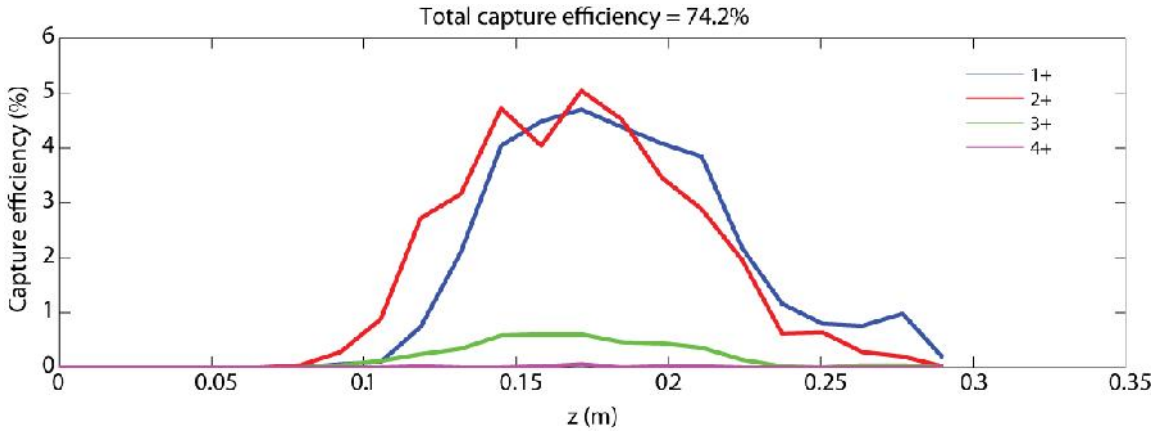

Figure 3: Profiles of captured Rb ions.

Diagnostics

If `output_trace` is turned on, MCBC can record the trajectories of the ions. Figure 4 is an example trajectory plot. It shows the trajectories of the first 10 injected rubidium ions, with different colors representing different charge states.

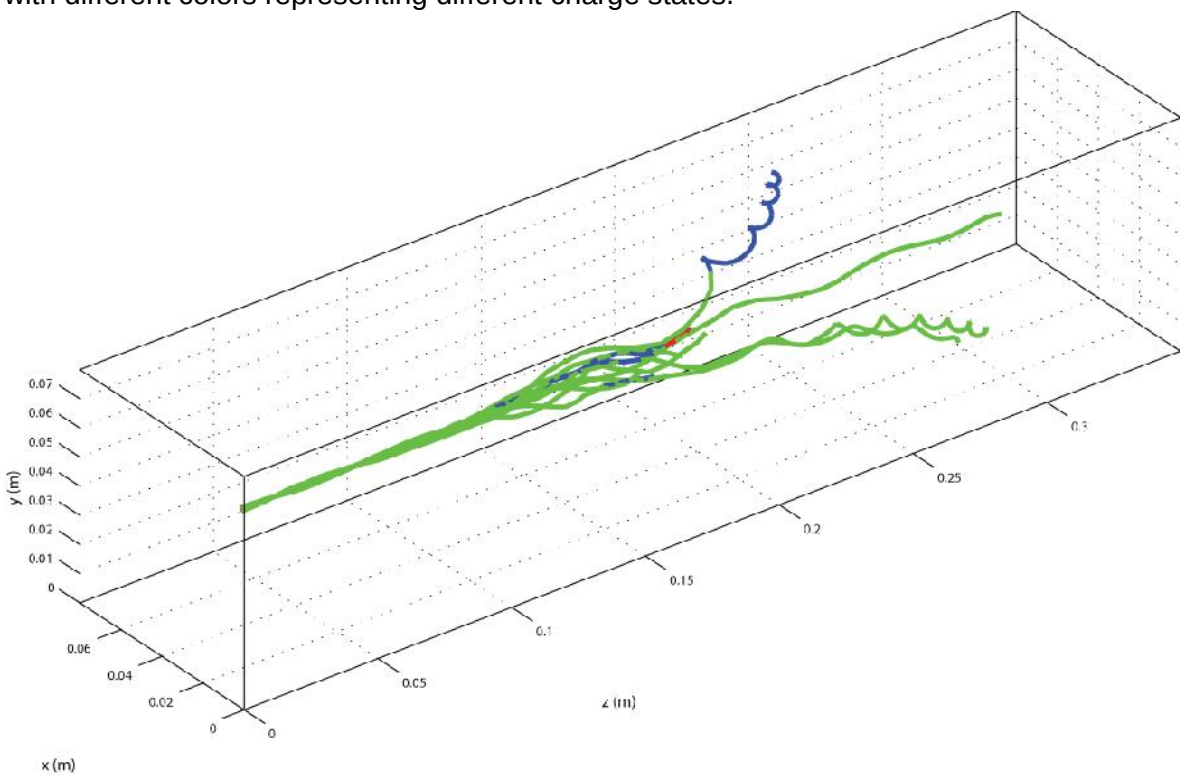

Figure 4: Ion trajectory plot. When q=1, the trajectory is plotted in green; when q=2, the trajectory is blue; when q=3, the trajectory is red.

If `output_ion_positions` is turned on, the initial and final positions of the ions are recorded (see Figure 5). Snapshots of the ion positions are also saved at the user-defined times in `snapshot_time_array` (Figure 6).

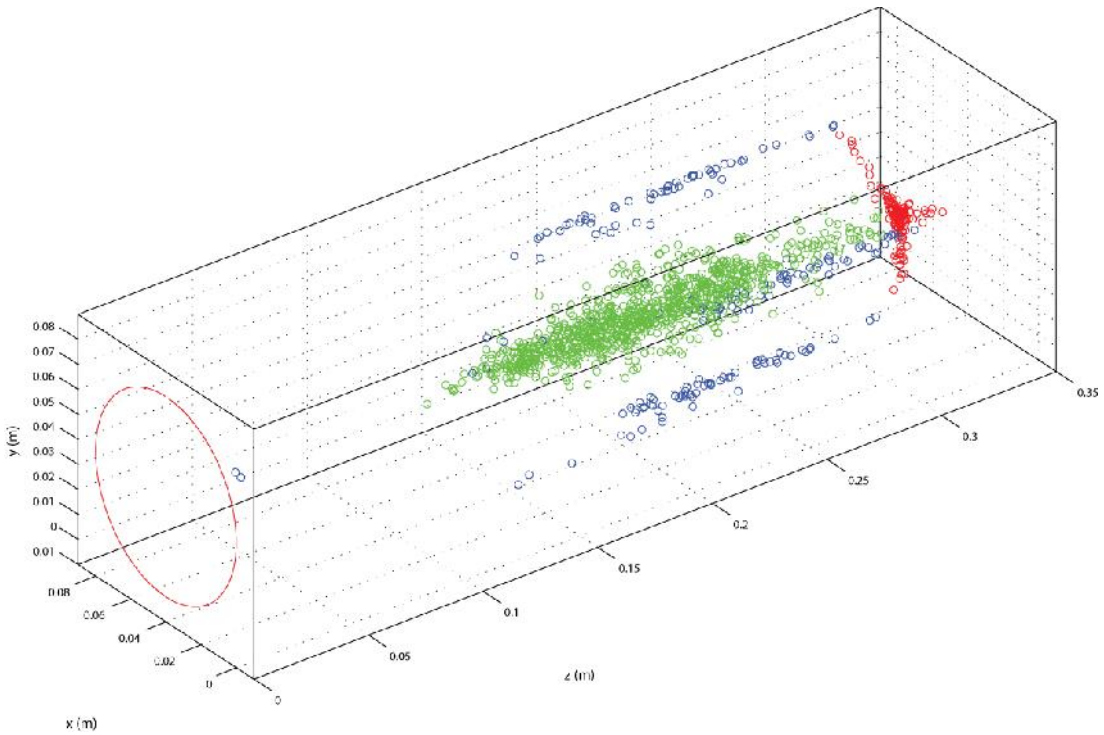

Figure 5: Recorded final ion positions. Red: ions exiting the device; green: captured ions; blue: ions that are lost to the wall or injection end.

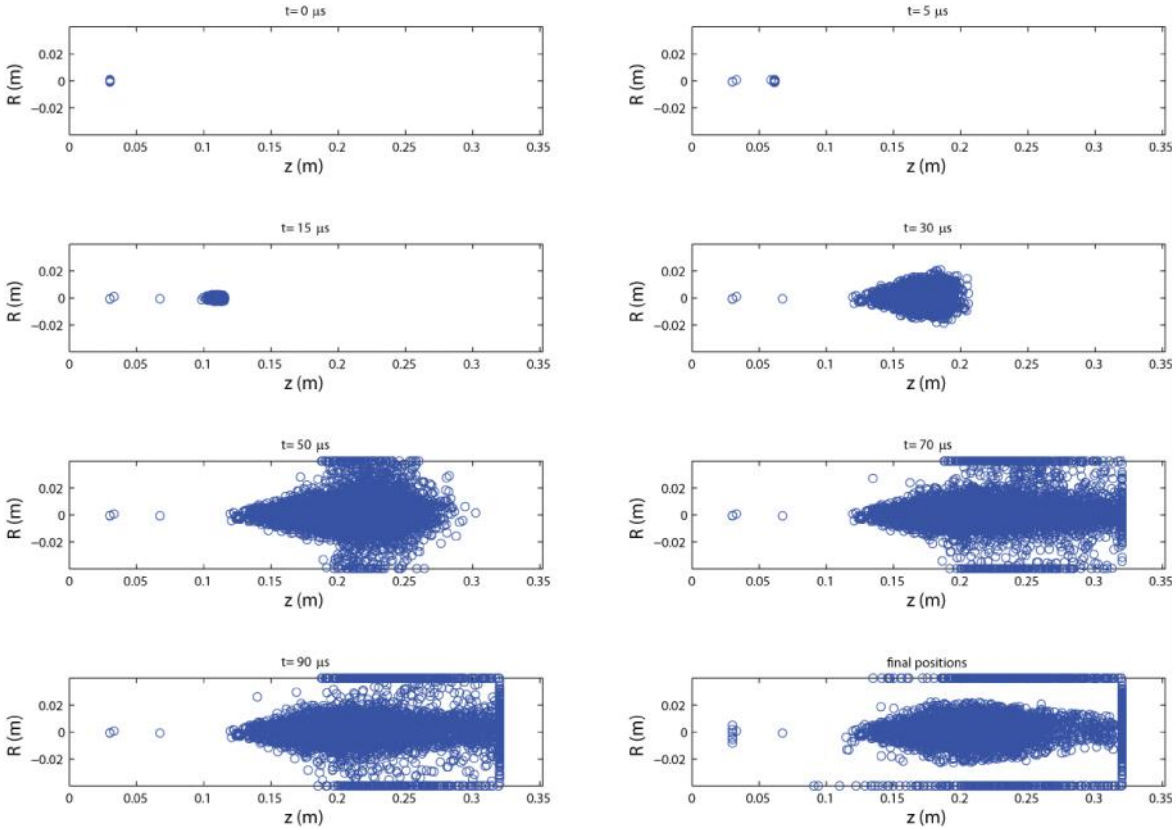

Figure 6: Snapshots of the ions at different times.