# New simulation code for ionization profile monitors

*D. Vilsmeier*[*1,2], *P. Forck*[1], *and M. Sapinski*[1]
[1]GSI, Darmstadt, Germany; [2]Universität Regensburg, Germany

Ionization Profile Monitors (IPMs) are devices for non-invasive beam profile measurement. They make use of phenomena of rest gas ionization by the beam. Electrons or ions produced in the ionization process are extracted from the region of interaction between the beam and the rest gas, and transported, using external electric and magnetic fields (called guiding fields), towards a detection device. The detection device is usually based on a phosphor screen or a segmented anode. The distribution of the positions of electrons or ions on the detector allow for reconstruction of the beam profile.

Although the principle of the IPM operation is simple, the reconstructed beam profile can be distorted due to various effects. Those effects include the nonuniformities of the guiding fields, initial kick given to electrons during the ionization (initial velocities) or the influence of the bunch fields on the trajectories of particles. Most of those effects cannot be treated analytically and specialized simulation program is needed. Such a program typically incorporates particle generation in the ionization process, calculation of the beam fields and tracking of charged particles in a superposition of transient bunch fields and constant guiding fields.

Surprisingly, none of the investigated available codes (CST and Geant4) has an ability to perform this kind of simulations therefore, in almost every laboratory which uses IPMs, researchers wrote their own codes [1,2]. Those codes are tuned to particular needs of the laboratories and are usually designed for specific cases only. Nevertheless, they contain a lot of common aspects.

In this situation we decided to create a modern, universal simulation code [3], which is easy to set up and which contains algorithms adequate for most of the investigated IPMs. The program, written in Python, has a modular structure and is easily extendable (it is being extended to Beam Induced Fluorescence monitor case). The main components are: ionization, guiding fields, bunch fields and tracking. Currently those components contain the following methods:

- Ionization: zero velocities, Voitkiv analytical formula [4] and phenomenological parametrization of double differential cross-section. This component is an independent package, so it can be used by other projects.
- Guiding fields: uniform fields and CST field map.
- Bunch field: analytical expression for special cases (Basetti-Eriskine, etc.) and numerical Poisson solvers (2D, 3D).
- Tracking: Runge-Kutta method, Boris algorithm and analytical expression (for parallel electric and magnetic fields only).

Additional components describe the detector part, the rest gas properties and manage the simulation output. The Graphical User Interface to the simulation program is written using Qt libraries and the simulation configuration, as well as the simulation output, can be handled using XML format. The code is open source and accessible via Python Package Index (virtual-ipm package).

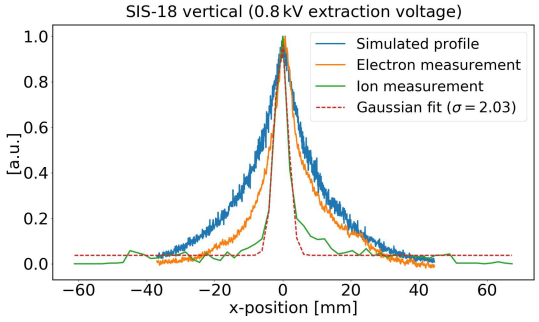


Figure 1: Measurement and simulation results for SIS18 IPM: measured beam profile (green), simulated beam profile distorted by the initial velocities of electrons and bunch field (blue) and the corresponding measurement (orange).

Figure 1 presents an example of results of the simulation program. The data were measured on June 28th, 2016 during $^{124}Xe^{43+}$ beam time using vertical IPM in SIS-18. The device was working in electron collection mode and the extraction voltage was decreased from 4 kV to 800 V in order to investigate the profile distortion. The beam had asymmetric shape, what has not been taken into account by the simulation, therefore a good agreement is seen only for half of the profile.

## References

[1] M. Sapinski et al.,"Ionization Profile Monitor Simulations - Status and Future Plans", IBIC 2016, Barcelona, September 2016, TUPG71

[2] https://twiki.cern.ch/twiki/bin/view/IPMSim

[3] https://gitlab.com/IPMsim/Virtual-IPM/

[4] A. Voitkiv et al., J.Phys.B: At.Mol.Opt.Phys.**32**(1999) p. 3923

* d.vilsmeier@gsi.de