Theory & Simulation Efforts for the Gamma Factory

Alexey Petrenko (Budker INP)

Gamma Factory meeting at CERN, March 26, 2019.

Gamma Factory Theory and Simulation efforts:

Evgeny Bessonov's Note:
Latest MS Word version is available at our OneNote Page at CERN.
Latex version:
https://www.overleaf.com/read/gpjksbkrxqcx
Another simplified version to be inserted into the Yellow Report:
https://www.overleaf.com/read/ztdwjrhqpyby

Simulations:

1) My python-based code: https://anaconda.org/petrenko/psi_beam_vs_laser.
2) Camilla Curatolo's code.
3) Wiesław Płaczek's modification of CAIN code.
4) RH code.
5) Semi-Analytical calculations by Aurelien Martens.

Mattermost channel on GF simulations:
https://mattermost.web.cern.ch/gammafactory/channels/simulation-tools

GF PoP parameter table: https://espace.cern.ch/PBC-acc-GammaFactory/default.aspx

Vittoria's GF FEL option studies.

My python-based simulations:

Earlier work:
Longitudinal dynamics of a single ion and an ion bunch in the LHC/SPS + laser:
https://indico.cern.ch/event/668097/contributions/2796070/
http://www.inp.nsk.su/~petrenko/misc/ion_cooling/animations/
https://apetrenko.blob.core.windows.net/misc/Li_like_Pb_cooling.html

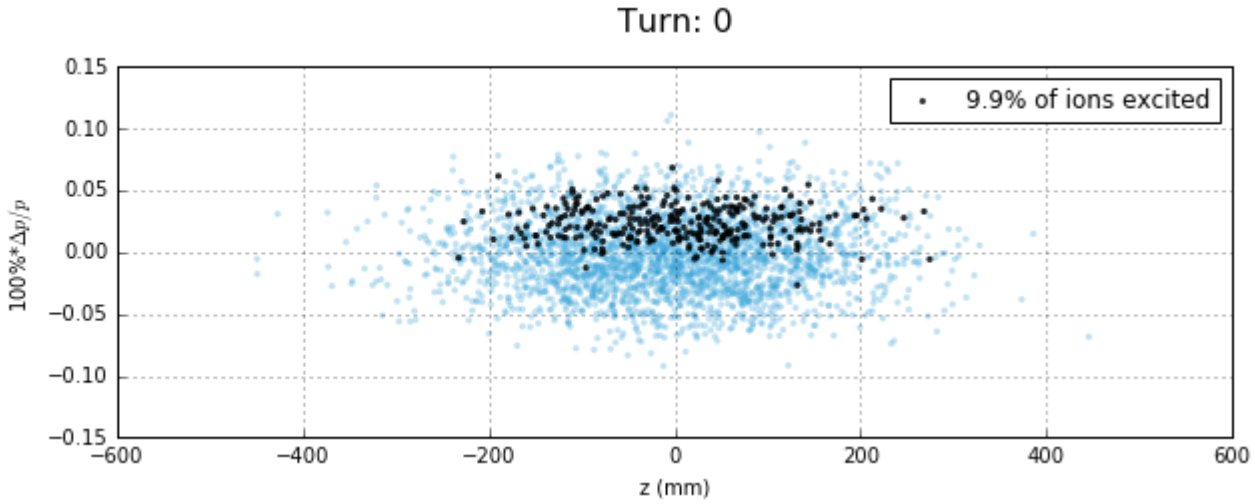
GF PoP Experiment:
Laser intensity estimates and Monte Carlo simulations for laser-ion bunch interaction region:
https://anaconda.org/petrenko/psi_beam_vs_laser

LHC-based Gamma Factory simulations:
https://anaconda.org/petrenko/lhc_psi_beam_vs_laser

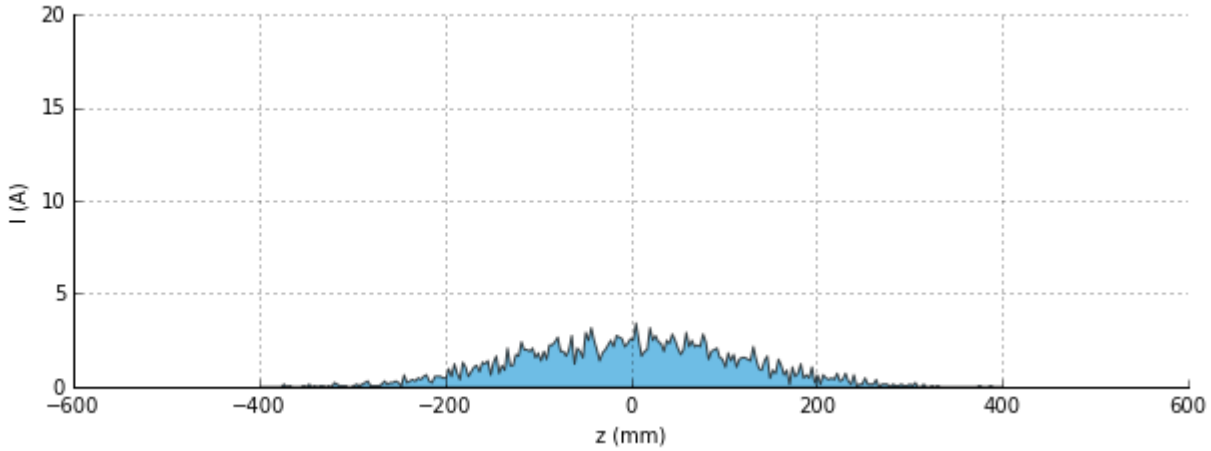
Next steps:
+ multiple photon absorption/emission (for LHC Gamma-Factory),
+ realistic laser parameters,
+ parallelization
+ release code as a python library
+ collective effects via longitudinal and transverse impedance
+ intra-beam scattering/stripping
+ …?

SPS PoP Experiment Cooling over 90 sec (2 degrees crossing angle):

A



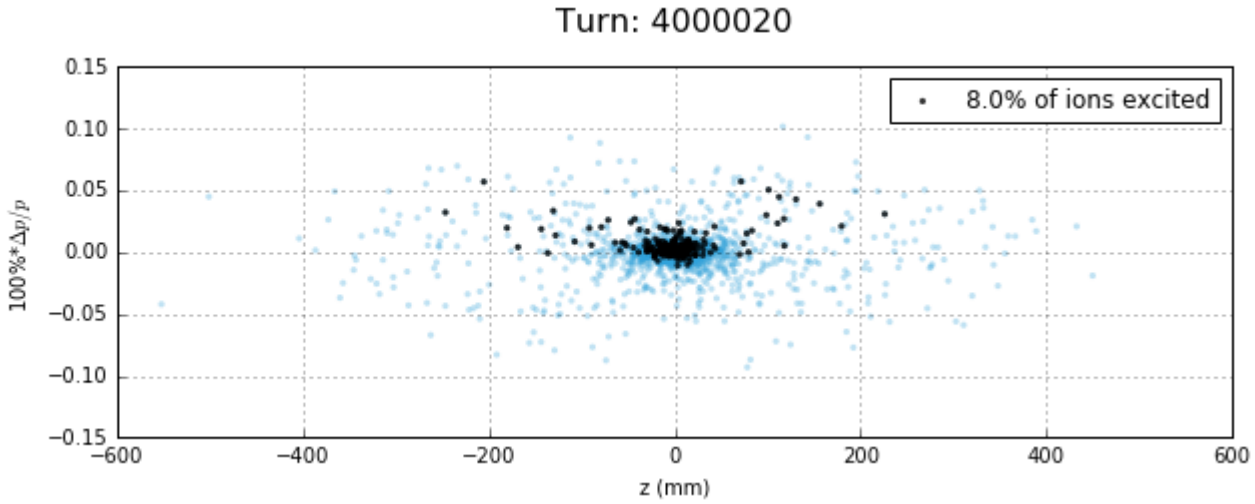
B



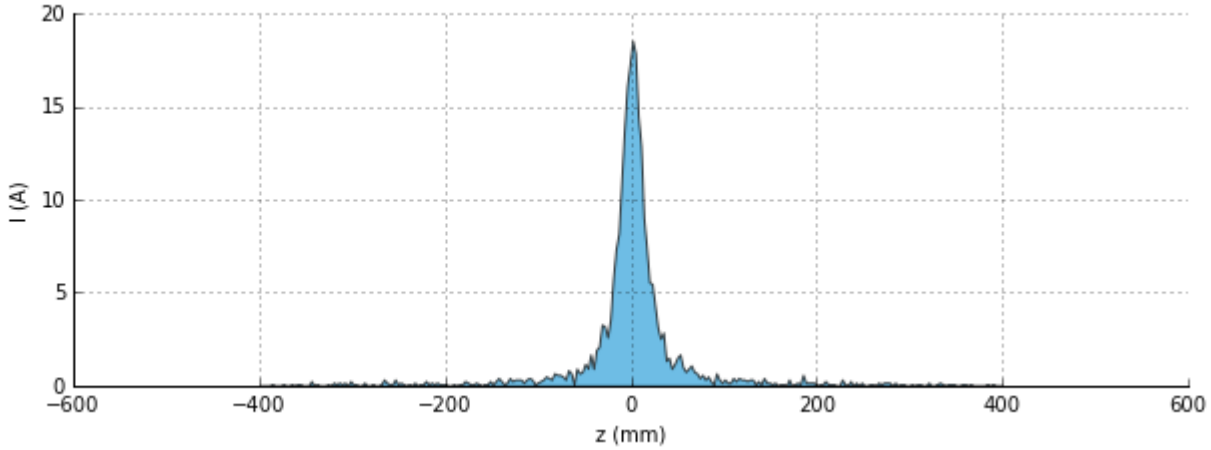
With 6 degrees crossing angle the number of excited ions is 2x less.

SPS PoP Experiment Cooling over 90 sec (2 degrees crossing angle):

A



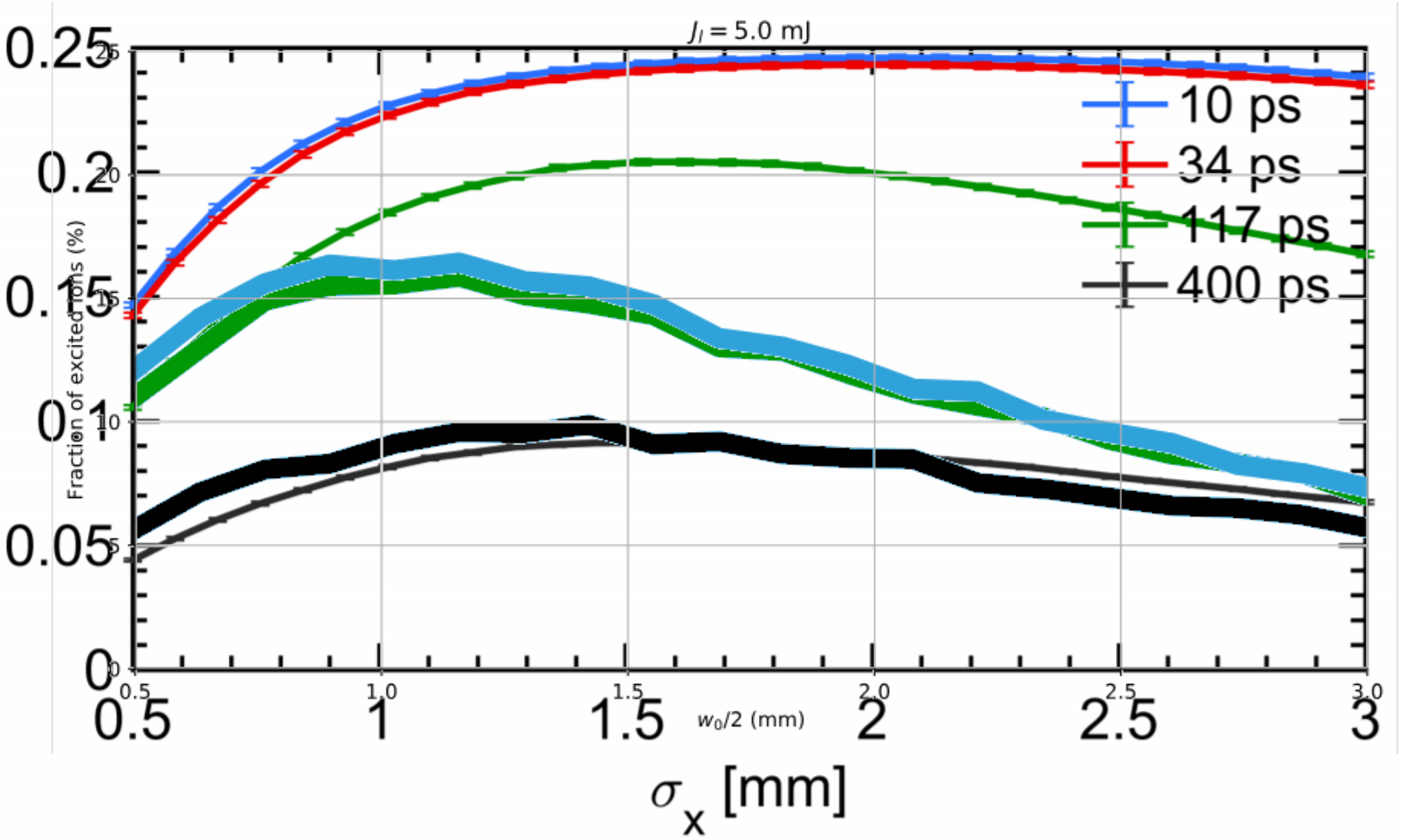
B



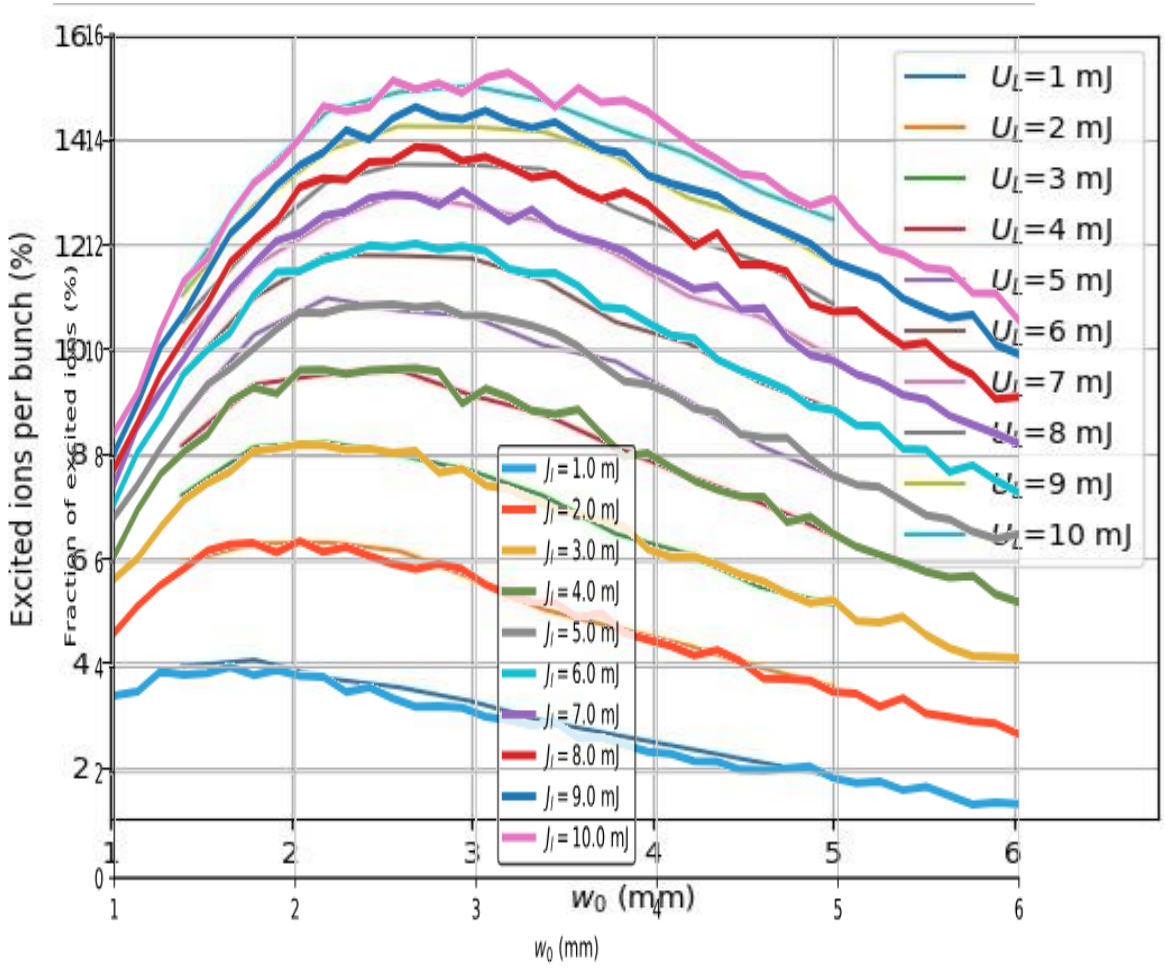
Calculation time: 1.5 hours on my laptop.

Code benchmarking

My python code vs Aurelien's code:



Camilla's plot:

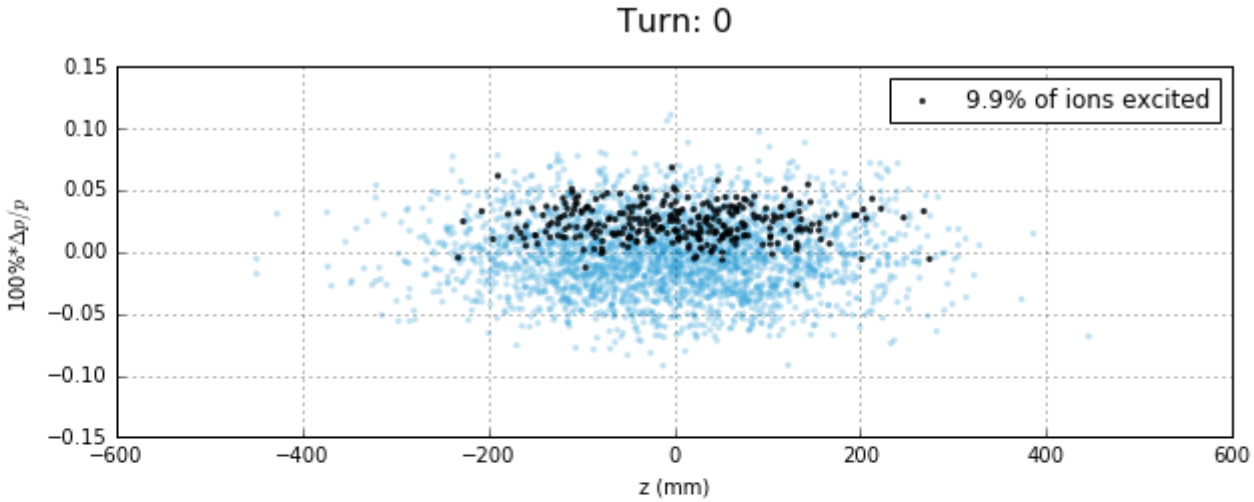



```
100000          !nions ----- num of macroparticles
193.687D+9       !mion ----- ion mass in eV
18.68908D+12     !eionmed  ----- mean ion energy
0.0003           !relenspread ------rel energy spread
0.001051         !sigx ---- in m
0.001171         !sigy ---- in m
0.12             !sigz ----- in m
2.D-6           !emitt_n
2.D+8            !n_ion ----num ion per bunch
230.76           !rismed ---- resonance energy in eV
0.001           !U_L energy laser in J
0.00015          !delas relative energy spread laser
0.5D-3            !sigl ---- rms transverse size laser in m
3.7D-12          !sigt ----- laser length in s
0                !ncmcut 1=selection in angle in CM/ 0=no sel
74.D-12          !tau0 ---- mean lifetime spont emission in s
6.               !dscreen ---- screen distance in m
1.               !rep -----repetition rate collisione
2.               !g1
2.               !g2
2.               !angcoll in deg
```

Perfect agreement!

A

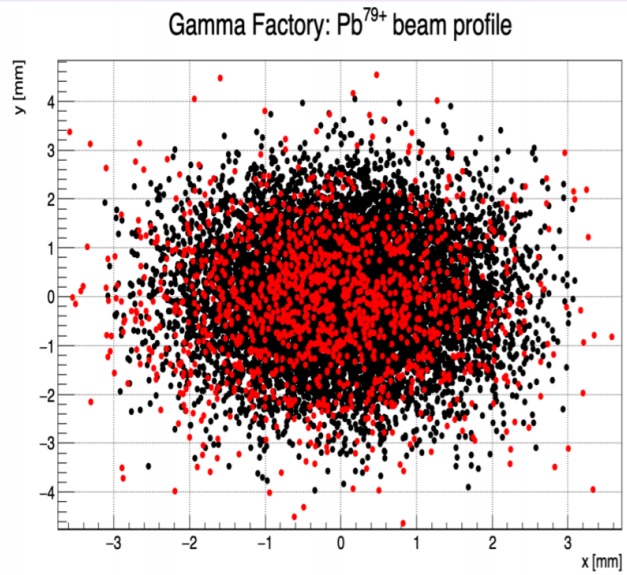
My python MC code:

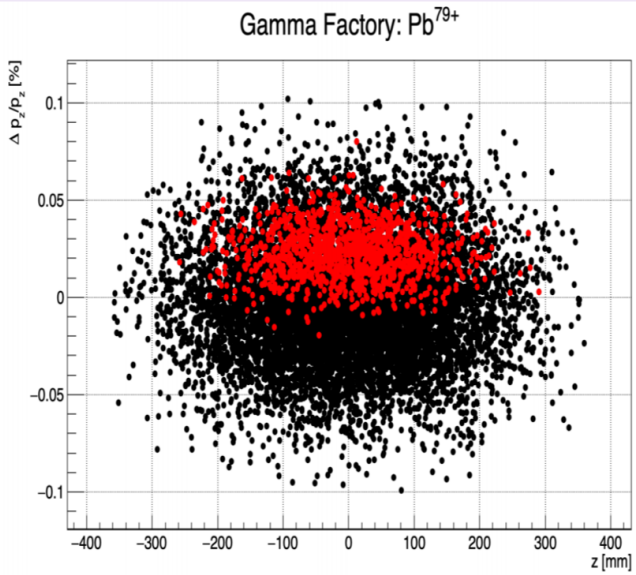


B

Doppler cooling of PSI beam

- Laser energy lowered by $2\sigma_\omega$ w.r.t. resonance energy
  - excited ions
  - other ions





▷ Fraction of excited ions: $\mathbf{N_\gamma/N_i = 9.7\%}$
(with spontaneous emission delay and stimulated emission)

[Wiesław Płaczek's modification of CAIN code]:

Conclusions & Plans

Full Monte Carlo Simulation of the Gamma Factory (SPS & LHC) beam dynamics is now possible.

Several SPS PoP experiment simulations have been benchmarked already – Good agreement observed!

Plans:

- Writing the Yellow Report,
- Applying the Monte Carlo code to different GF regimes.
- Integrating the code into other programs
- …

Monte Carlo scheme:

1) Define the ion beam

```
X = np.matrix([
  x ,
  xp,
  y ,
  yp,
  z ,
  dp
])
```

2) Define the 1-turn transformation (matrix + RF-cavity as a function)

```
M = np.matrix([
    [ R11,  R12, 0  , 0  ,  0  ,  R16],
    [ R21,  R22, 0  , 0  ,  0  ,  R26],
    [ 0  ,  0  , R33, R34,  0  ,  0  ],
    [ 0  ,  0  , R43, R44,  0  ,  0  ],
    [-R51, -R52, 0  , 0  ,  1  , -R56+L/(gamma_0*gamma_0)],
    [ 0  ,  0  , 0  , 0  ,  0  ,  1  ]
])
```

3) Define laser beam

Then turn-by turn simulation is defined in a simple loop:

```
for turn in range(0,100000):
    Excited = ExciteIons(X)          # ion excitations
    X = EmitPhotons(X, Excited)      # photon emissions
    X = RFcavity(X, h, eVrf, phi0) # beam goes through RF-cavity
    X = M*X                          # 1-turn matrix
```

Maybe it already makes sense to publish these functions as a python library.