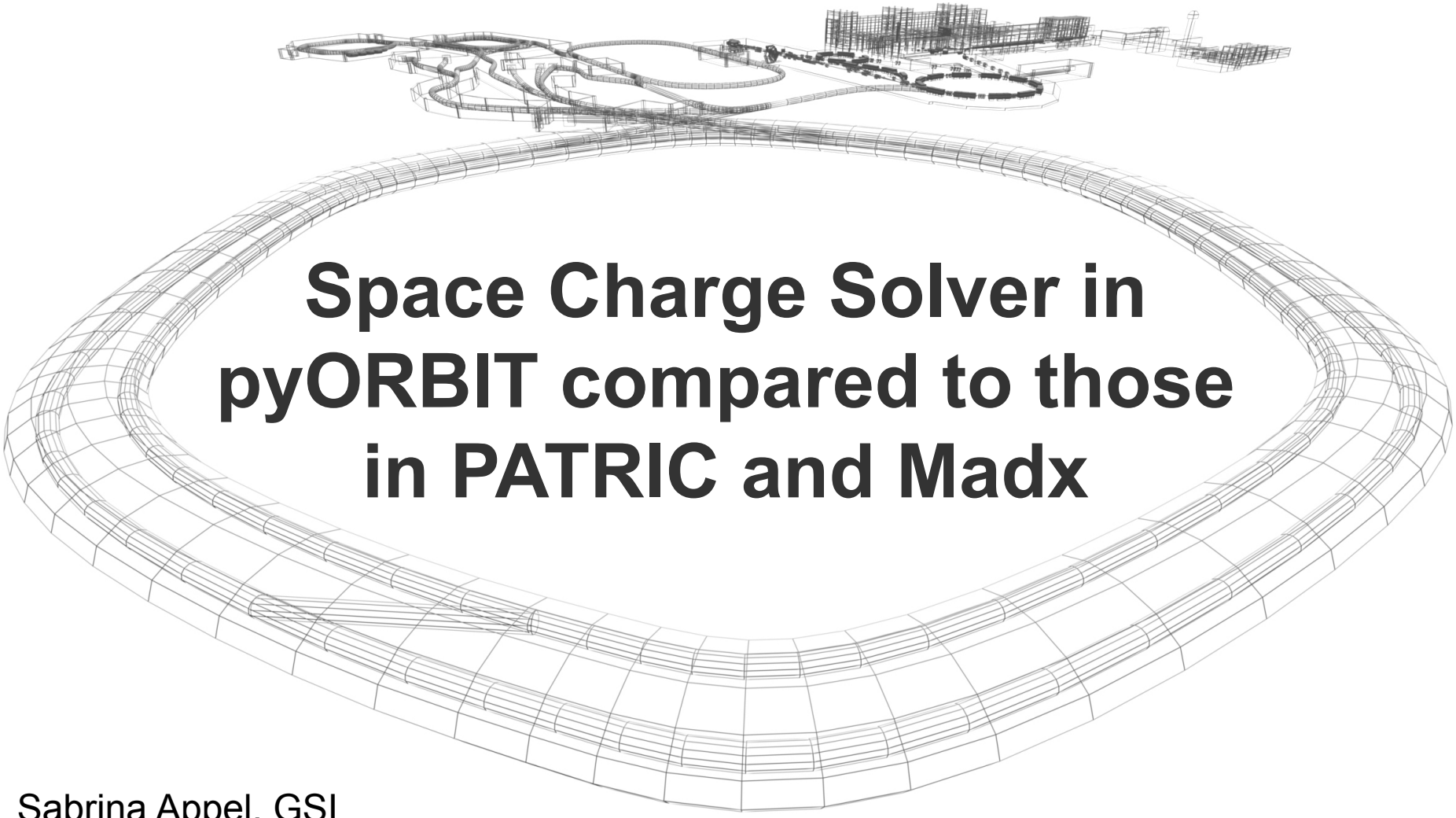
GSI Helmholtzzentrum für Schwerionenforschung GmbH



Space Charge Solver in pyORBIT compared to those in PATRIC and Madx

Sabrina Appel, GSI

Outline

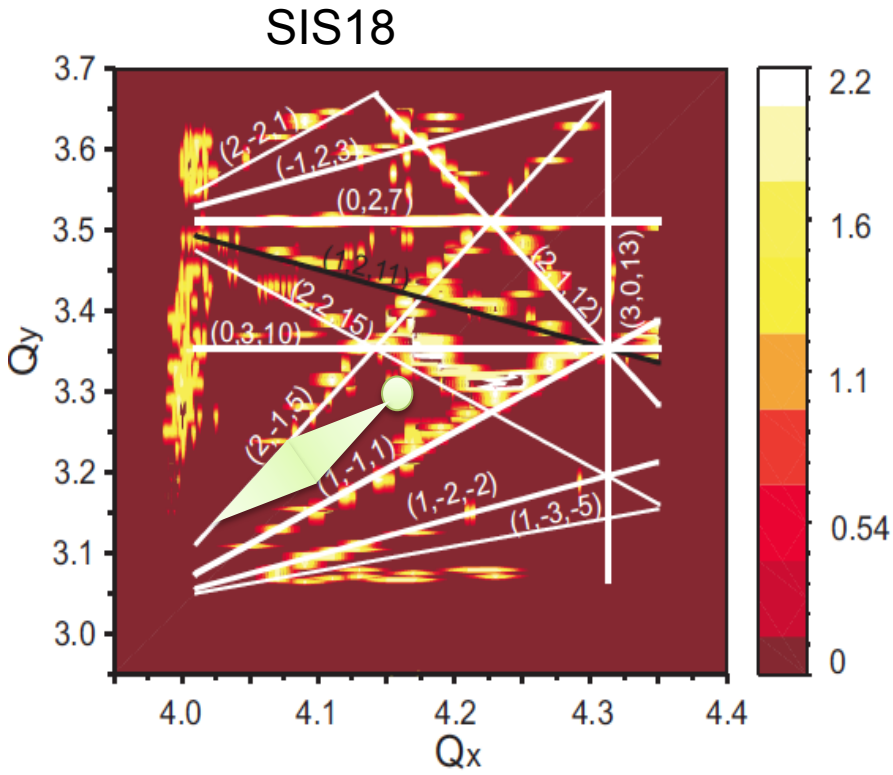
- Space charge (SC) effects in SIS18/100
- SC Solver
  - Frozen SC Solver (Madx)
  - PIC Solver
    - Longitudinal SC (pyorbit)
    - Transversal SC
      - PATRIC
      - pyORBIT
- SC matching
  - Transversal and longitudinal
- Tune footprint for Madx, pyORBIT and PATRIC
- Summery and Outlook

SIS18 Space charge effects

- Intensity goal: The space charge limit of ΔQ_y=0.5
- Injection energy: 11.4 MeV/u ($ß_0$=0.155), Emittances: $\epsilon_{x,y} = 150 / 50$ mm mrad

$$\Delta Q_y^{sc} = -\frac{2NZ^2 g_f}{\pi A \beta_0^2 \gamma_0^3 B_f \left(\varepsilon_y + \sqrt{\varepsilon_y \varepsilon_x}\right)}$$

- Space charge induced voltage reduction by 40%
- Dual rf bucket with flattened bunch profile
- Codes demands:
  - Resolution of synchrotron motion (long bunches) -> 2.5 D SC solver
  - Dual rf
  - Matching routines for longitudinal and transversal space charge
- Studies:
  - Long term simulation after MTI
  - Comparison to frozen space charge simulation results

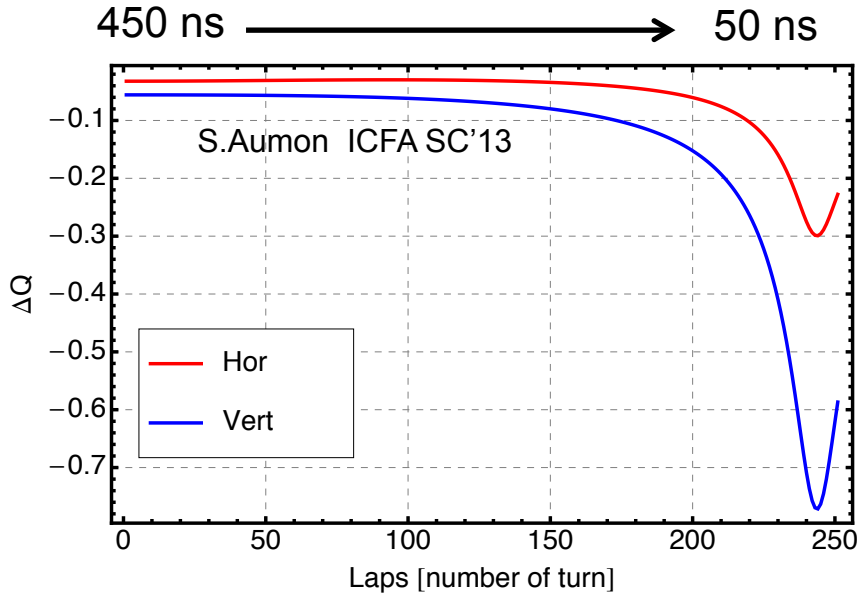


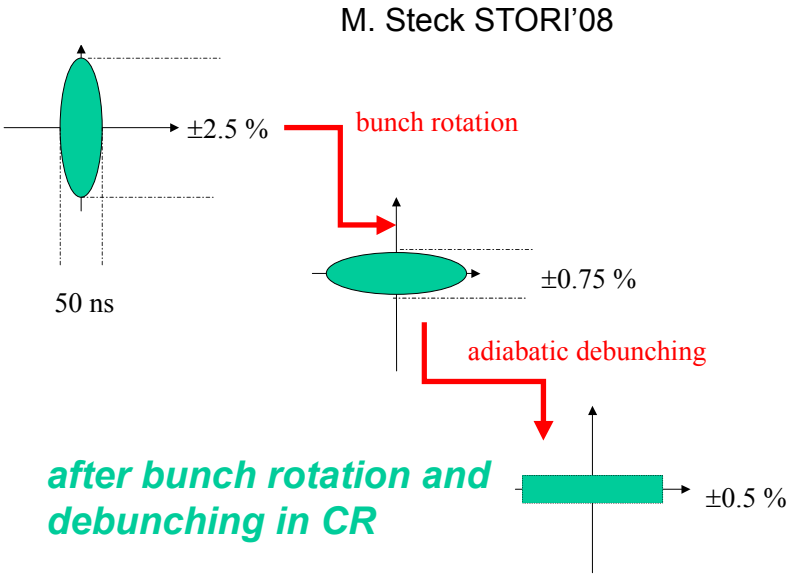
SIS100 Bunch compression

- Intense short ions beam required by experiments for **plasma physics** and **exotic elements productions**

  -> 50 ns short U^{28+} beam with $5x10^{11}$ particles per bunch at 1.5 GeV/u

- Space charge affects the optics
- Large transverse space charge tune shift



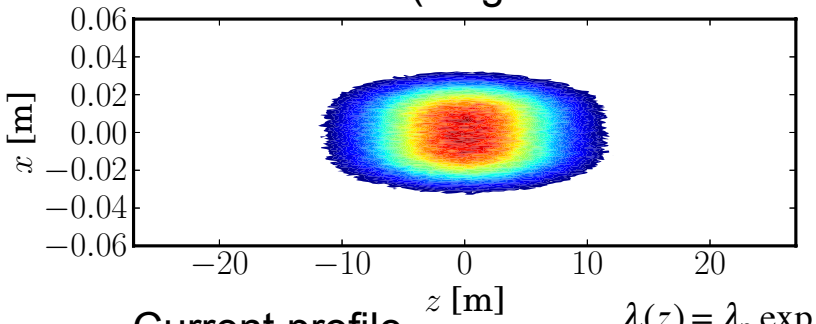


- Studies:
- Space charge effect on the transverse beam distribution (2D)
- Effects during bunch compression (magnets error, resonances etc..)

PATRIC: Space charge tune shift

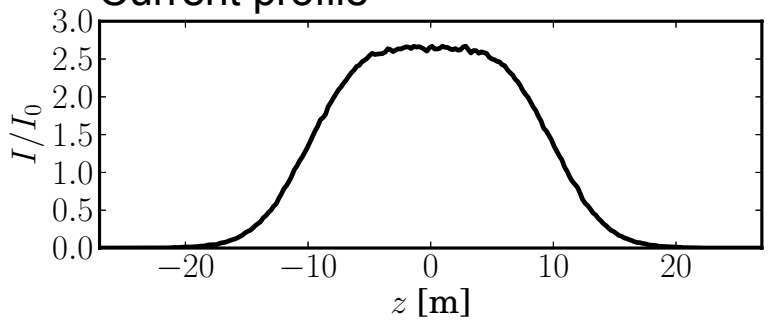
✓ Matched 3D bunch distribution (in a dual rf bucket)

Bunch distribution (longitudinal-horizontal)

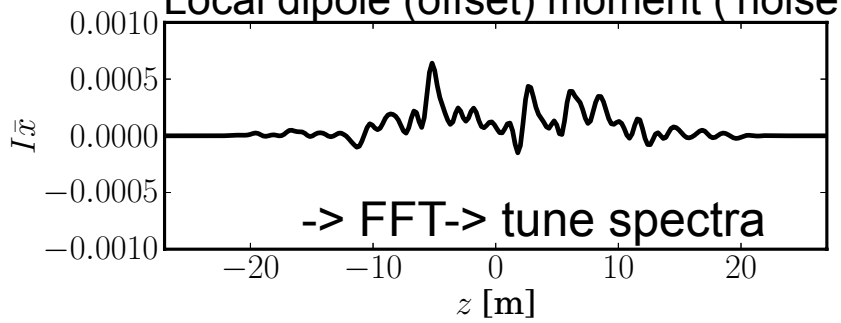


Current profile

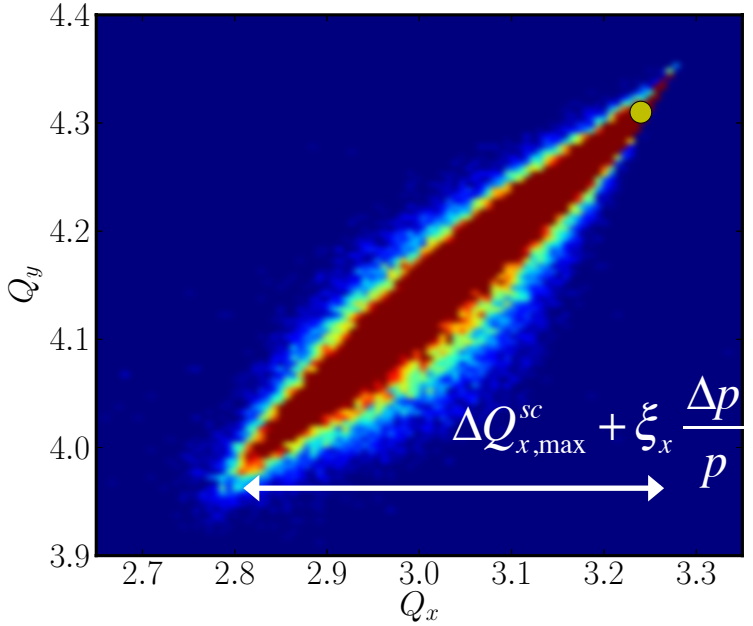
$$\lambda(z) = \lambda_0 \exp\left(-\frac{z^4}{2\sigma_l^4}\right)$$



Local dipole (offset) moment ('noise')



Tune footprint with space charge (SIS18)

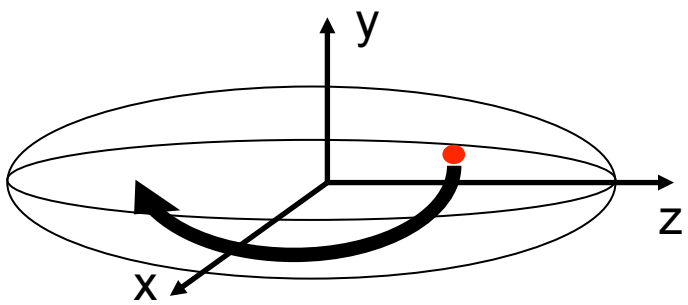


O. Boine-Frankenheim, ICFA SC 2013

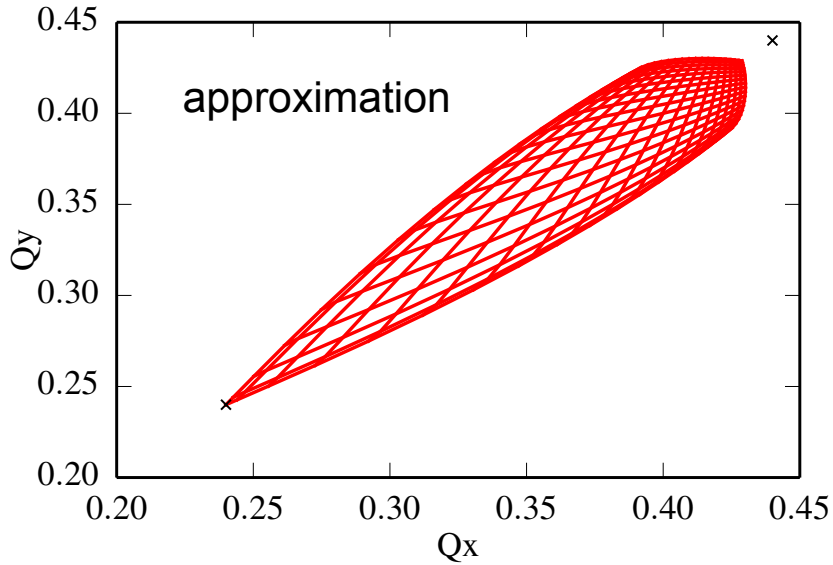
Frozen space charge

- The solution of Poisson equation for a 2D Gaussian can be calculated analytical

$$\nabla^2\Phi = \frac{\rho}{\epsilon_0\gamma^2} \qquad \rho \sim e^{-\left(\frac{x^2}{2\sigma_x^2}+\frac{x^2}{2\sigma_x^2}\right)}$$



- Also the space charge tune spread as a function of the particle amplitude

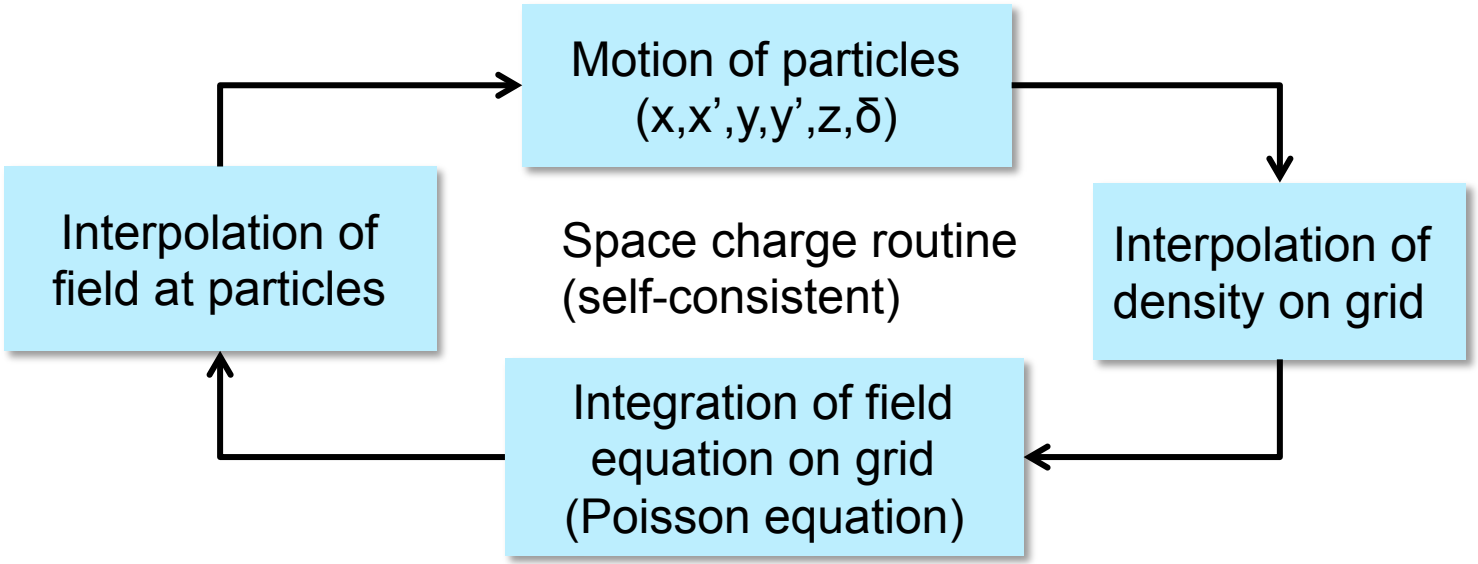


- Tracking: The kick acting on the particle is computed from the electric field (analytical)

- Sometimes during tracking simulations the beam intensity and Gaussian beam size are adapted

See e.g. A. Burov, et. al., Transverse instabilities of coasting beams with space charge, Phys. Rev. ST-AB (2009)
M. Bassetti, et. al., Closed expression for the electrical field of a two-dimensional Gaussian charge, CERN-ISR-TH/80-06

PIC algorithms: Space charge routine



- The particle-to-particle and particle-to-environment interactions are calculated with PIC (Particle In Cell) algorithms

$$\nabla^2\Phi = \frac{\rho}{\epsilon_0\gamma^2}$$

- Space charge forces or potential are obtained by solving the Poisson equation (2D: Solve the electric field equation)

$$\frac{\partial E_x}{\partial x} + \frac{\partial E_y}{\partial y} = \frac{\rho}{\varepsilon_0}$$

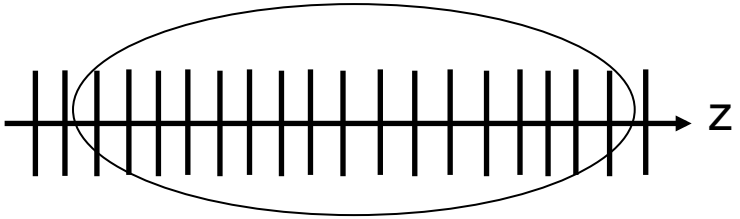
- Solving the integral from on grid with FFT and boundary condition for arbitrary beam distribution

$$\Phi(\vec{r}) = \int \frac{\rho(\vec{r}-\vec{r})}{|\vec{r}-\vec{r}'|^2} d\vec{r}'$$

PyOrbit: Longitudinal impedance + space charge

- The longitudinal impedance is represented by its harmonic content in terms of the fundamental ring frequency

Particle are binned longitudinally (1D)



FFT of binned distribution
$Z_n^{sc}+Z_n^{extern}$

Calculate and add kicks for each particles

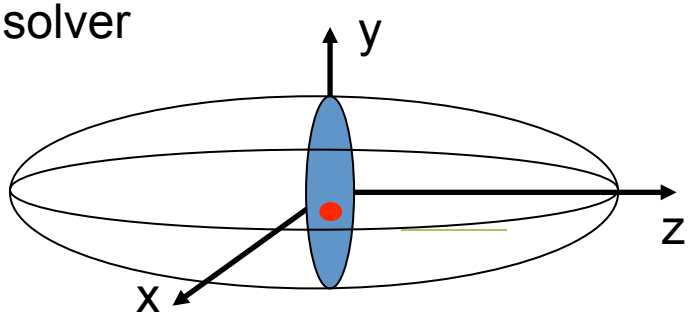
```
void LSpaceChargeCalc::trackBunch(Bunch* bunch){
…
for(int n = 1; n < nBins / 2; n++){
  realPart = std::real(_zImped_n[n]);
  imagPart = std::imag(_zImped_n[n]) + zSpaceCharge_n;
  _z[n] = n * sqrt(realPart * realPart + imagPart * imagPart);
  _chi[n] = atan2(imagPart, realPart);}
…
 for(int n = 1; n < nBins / 2; n++){
  cosArg = n * (angle + OrbitConst::PI) + _fftphase[n] + _chi[n];
  kick += 2 * _fftmagnitude[n] * _z[n] * cos(cosArg);}
  return kick;
}
```

- Similar approach in PATRIC

Source: Talks by S. Cousineau and J.A. Holmes, ORNL

PyOrbit: Transverse Space Charge Solver

- Solving Poisson equation on 2D grid with fast FFT solver
  - Particle are binned in 2D rectangular grid (momentum-conserving interpolation scheme)
- Grid is adapt to beam size
- Kicks are weighted by local longitudinal density (non-uniform distribution)




```
void SpaceChargeForceCalc2p5D::trackBunch(Bunch* bunch, double length){
…

forceSolver->findForce(rhoGrid, forceGridX, forceGridY);
forceGridX->interpolateBilinear(x,y,fx);
forceGridY->interpolateBilinear(x,y,fy);
…

for(i=0, bunchSize,i++){
Lfactor =  zGrid->getValue(z) * factor;
bunch->xp(i) += fx * Lfactor;
bunch->yp(i) += fy * Lfactor;}
}
```

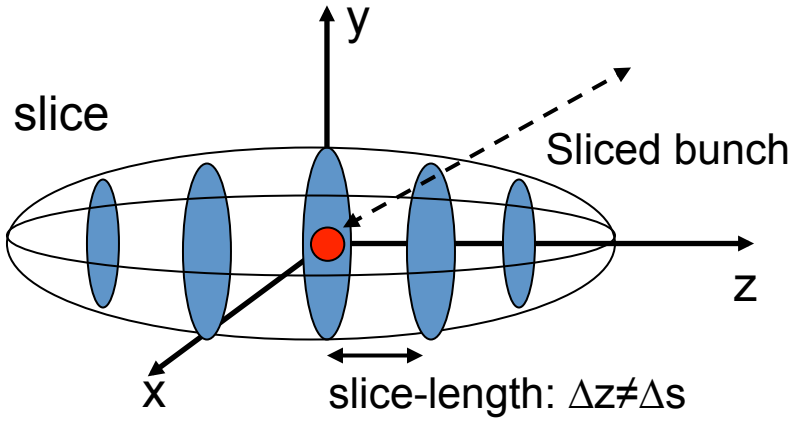
Source: Talks by S. Cousineau and J.A. Holmes, ORNL and pyORBIT source code

PATRIC: Transverse Space Charge Solver

- Bunch is sliced n times
- The 2D space charge field is computed for each slice
- Grid is static

$$\frac{\partial E_x}{\partial x}+\frac{\partial E_y}{\partial y}=\frac{\rho(x,y,\{z,s_m\})}{\varepsilon_0}$$

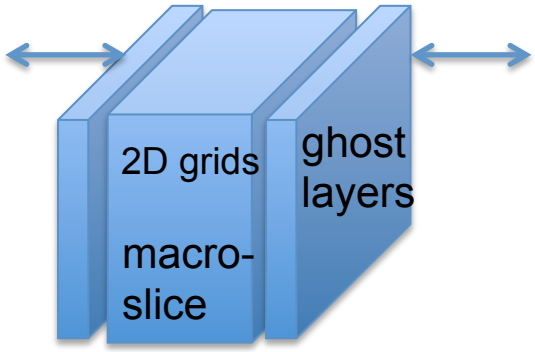
(fast 2.5D Poisson solver)




```
Pics.gatherXYZ(charge*qe/rho_xyz.get_dz(),rho_xyz); # 3D Grid

 // send and receive density ghost grids to neighbor
MPI_Isend(rho_xyz.get_ghostl(),….)
MPI_Recv(rho_xy_tmp.get_grid(),,…)
…

poisson_xyz(Ex3,Ey3,rho_xyz,gf1);  # 2D Poisson solver for slice

// send and receive efield ghost grids to neighbor slices
MPI_Isend(Ex3[0].get_grid(),…)
MPI_Recv(Ex3.get_ghostr(),…)
…
Pics.kick(Ex3,Ey3,ds)
```

- Corrected interpolation

MPI send/receive to neighbor slices



➢ Grid slices “feel” all other slices

Oliver’s talk, ICAP 2012

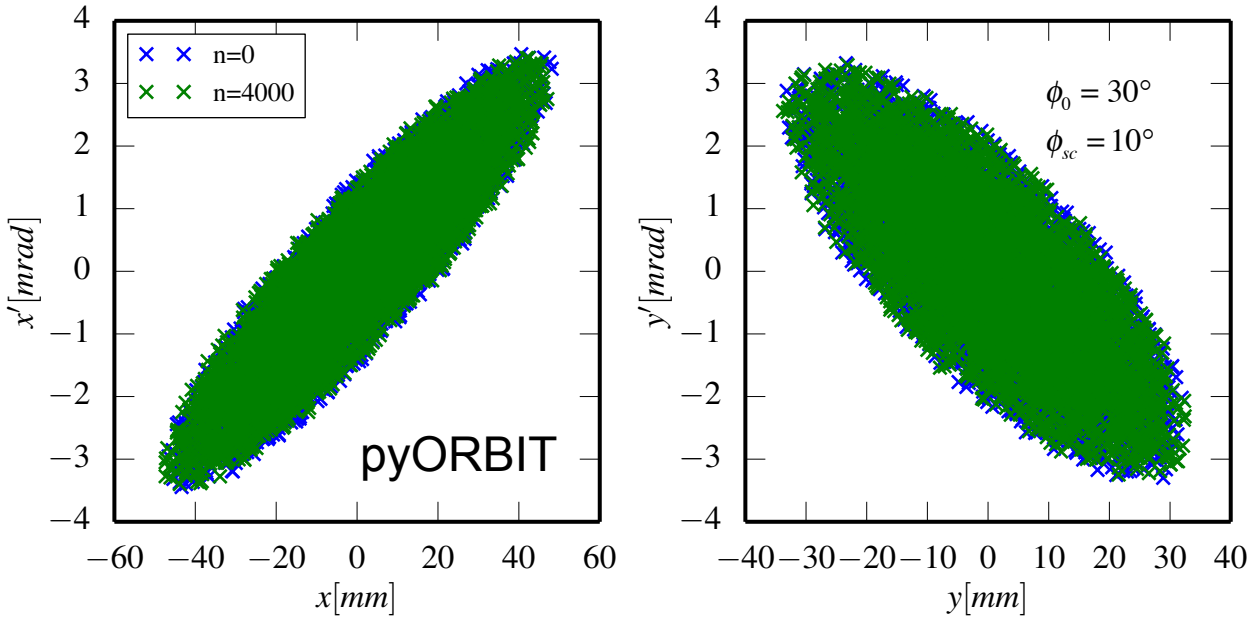
Trans. SC matching in pyORBIT and PATRIC

See e.g. Transverse matching with space chage (Venturini, Reiser, PRL 1998)

- Solving of the two coupled beam envelopes equation with space charge

$$X'' + k_x X - \frac{2K}{X+Y} - \frac{\epsilon_x^2}{X^3} = 0 \qquad Y'' + k_y Y - \frac{2K}{X+Y} - \frac{\epsilon_x^2}{Y^3} = 0$$

```
envx,envxs,envy,envys,Dx,Dxs,s = match_root(lattice,emitx,emity,sigma_p,Ksc)
phasex,phasey = phase_advance(envx,envy,Dx,emitx,emity,dp,s) # check
```



Shown only 10% of computed macro-particles

See e.g.: Oliver's presentation tomorrow

Long. SC matching for arbitrary rf wave forms

See e.g. Hofmann-Pedersen, The bunch distribution can be matched to arbitrary rf bucket forms (1979)
O. Boine-F., *rf barrier compression with space charge*, Phys. Rev. ST-AB (2010

'Hamiltonian':

$$H = \frac{v^2}{2} - \frac{qV_0}{m^* L} Y(z)$$

Potential function:

$$Y(z) = \int_0^z V(z)dz \qquad V(z) = V_{rf}(z) + V_{sc}(z)$$

Single rf wave:

$$V_{rf}(z) = V_0 \sin(hz/R)$$

Dual rf wave:

$$V_{rf}(z) = V_0 \left(\sin(hz/R) + \frac{1}{2}\sin(2hz/R) \right)$$

Elliptic bunch distribution:

$$f(z,v) = f_0 \sqrt{1 - \frac{Y_{rf}}{Y_m} - \frac{v^2}{v_m^2}}$$

Potential at the bunch end: $Y_m = Y(z_m)$

Maximum velocity in the bunch center:

$$\frac{v_m^2}{2} = \frac{qV_0}{m^* L} Y_m$$

Long. SC matching in pyORBIT and PATRIC

1. Set rf field and potential on 2D grid ($N_y=N_x=2N_{bin}$)

```
machting.set_rf_dual(rf_field,Yrf_field,
RFHNum,RatioVoltage)
```

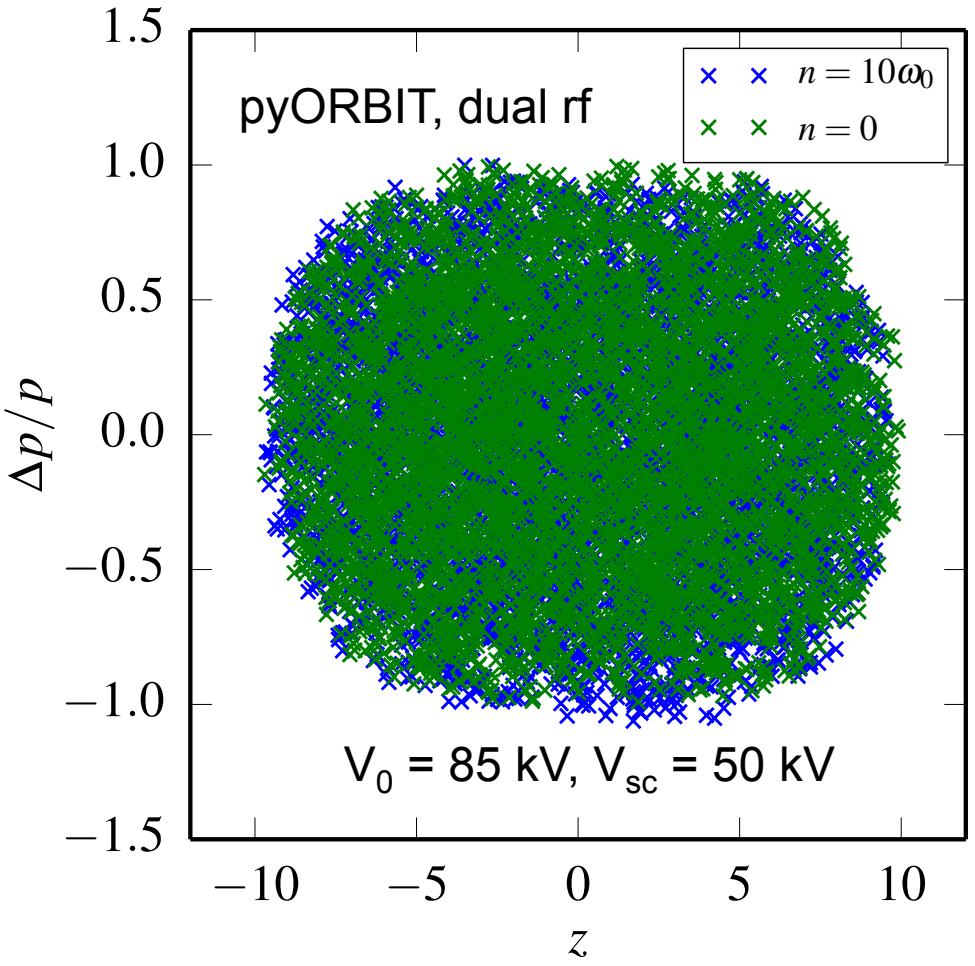
2. Determine matched voltage and beam distribution f(z,v) on grid

```
V0, distf =
machting.match_elliptic(Yrf_field, distf,
zm, dpBunch, Zs, intensity)
```

3. Generate random numbers sets x_i ,v_i from $f(z_j,v_k)$

```
x,dp = machting.matchPIC(NPIC, distf)
```

Shown only 10% of computed macro-particles



-> also Gaussian distribution possible

Betatron tune calculation

- Average Phase advance (APA) method
  - Non instantaneous
  - Evaluation of phase difference for each particle after each turn (normalized coordinate system)
  - Lattice function must be known

$$\theta_{n,j} = \arctan \frac{x'_{n,j}}{x_{n,j}}$$

$$\tilde{Q}_j = \frac{\theta_{n-1} - \theta_n}{2\pi}$$

$$\Delta = \frac{1}{n}$$

- FFT method
  - Non instantaneous
  - FFT analysis of the average transvers oscillations
  - Resolution given by the number of average turns/cells

$$FFT(x, y)$$

$$\Delta = \frac{1}{n}$$

- One-Turn-Matrix (OTM) method
  - Quasi instantaneous
  - Reconstruction of OTM for each particle
  - Trace or eigenvalues of OTM gives fraction turn

$$x_j^{n+1} = M_j x_j^n$$

$$\tilde{Q}_j = \frac{1}{2\pi} \arccos \frac{Tr(M_j)}{2}$$

See e.g: Bartolini, R, Bazzani, A, Tune evaluation in simulations and experiments, CERN SL/95-84 (AP)
Luccio, A, D'Imperio N., Eigenvalues of the One-turn Matrix, C-A/AP'126

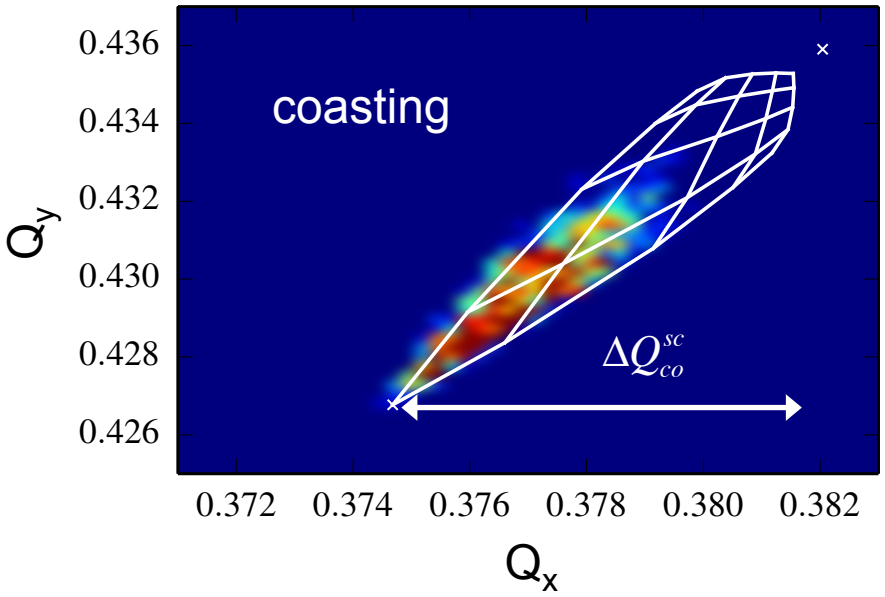
Madx: Tune footprint

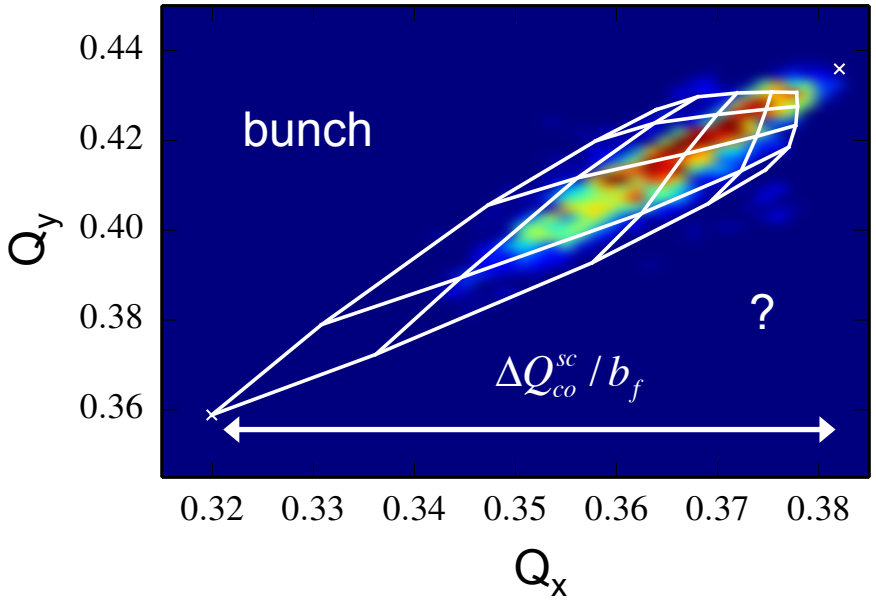
- Frozen space charge
- Simple FODO lattice with SC

```
bb_i: beambeam, sigx:=sigxi, sigy:=sigyi, bbdir=1;
fodo0: line=(bb_1,dr,bb_2,qf,bb_3,dr,bb_4,dr,bb_5,qd,bb_6,dr);
fodo_mult: line=(rfcav,12*fodo0);
```

✓ Coasting beam

○ Bunch beam
- SC tune spread is a factor of two too small
- Meaning of the options in the input files are unclear



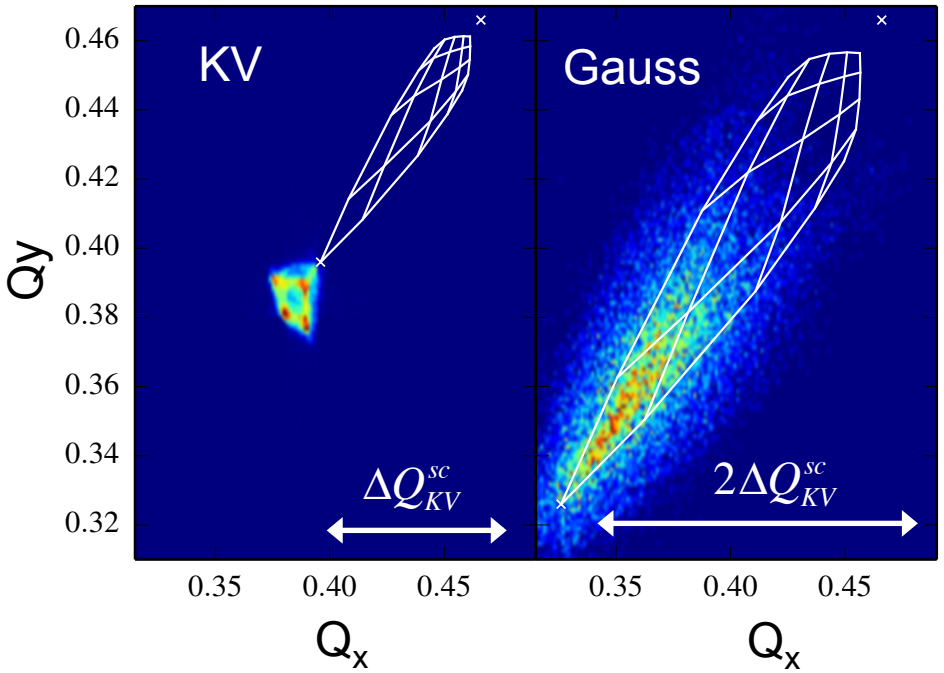


Result from Stefan Sorge

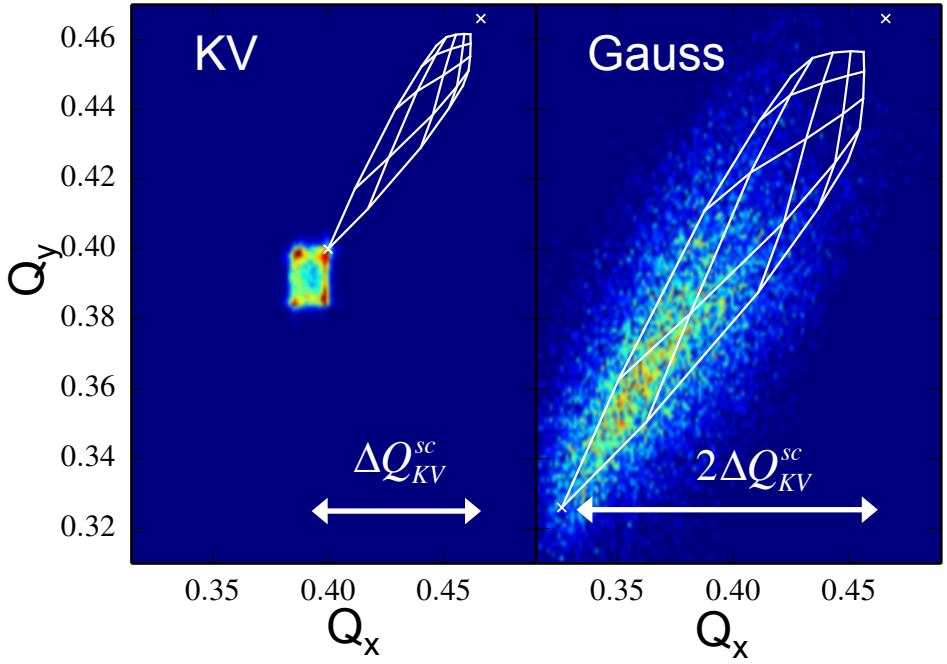
PATRIC versus pyORBIT

- Tune footprint for 2D space charge solver
- In both codes constant focusing and coasting beam has been used
- Good agreement between the two codes

PATRIC



pyORBIT



- In the early version of the presentation the tune footprint of pyORBIT simulation results with a Gauss distribution was distributed over the whole white diamond -> probable an effects of the lattice, constant focusing gives the expect tune footprint for all cases

Summery and Outlook

- Space charge (SC) effects in SIS18/100
- SC Solver
  - Frozen SC Solver
    - Only Gaussian distribution
    - Solution of Poisson equation is solved analytical
  - PIC Solver
    - Arbitrary beam distribution and self-consistent
    - Poisson equation solved on Grid with FFT (different approaches)
- SC matching pyORBIT and PATRIC for trans. + long.
- Tune footprint for Madx, pyORBIT and PATRIC
  - More or less in good agreement
- SIS18/100: Bunch compression (“2D”)
- Long term “3D” simulation in SIS18 after MTI