Incoherent effects from e-cloud with SixTrackLib

Konstantinos Paraschou[1,2], Giovanni Iadarola[1]

[1]CERN, Switzerland
[2]Aristotle University of Thessaloniki, Greece

Acknowledgements: H. Bartosik, R. De Maria, L. Giacomel, Y. Papaphilippou, L. Sabato, M. Schwinzerl, G. Skripka

185th HiLumi WP2 Meeting
CERN, Tuesday, 24th November 2020

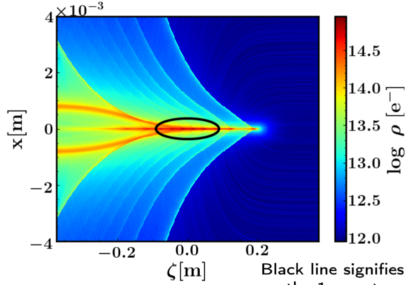
Overview

1. Recap
   - Motivation
   - Electron cloud kick
   - Tricubic interpolation
   - Macroparticle noise
   - Interpolation artifacts
2. Configuration of simulations
   - SixTrackLib implementation
   - Tracking sequence configuration
   - Collimators
3. Tests
   - RF-Multipole test
   - PyHEADTAIL footprint test
4. Tracking Results
   - Frequency Map Analysis
   - Dynamic Aperture
   - Long-term tracking (losses)

Motivation

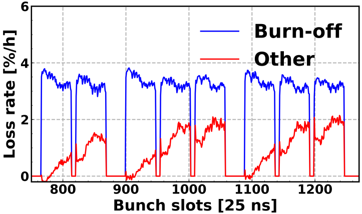
Electrons trapped in beam chamber (Electron Cloud) can introduce non-linearities in single-particle beam dynamics.

PyECLOUD simulation:



Black line signifies the 1σ contour

LHC experimental observations:



Bunch-by-bunch pattern[1] on (slow) losses resembles typical E-cloud buildup behaviour.

[1] More details in G. Iadarola, LBOC meeting 112

Electron cloud kick

It is possible to prove[2,3] that e-cloud kick can be written as the gradient of a scalar potential:

$$
\begin{aligned}
x, y, \tau &\mapsto x, y, \tau \\
p_x &\mapsto p_x - \frac{qL}{\beta_0 P_0 c}\frac{\partial \phi}{\partial x}(x, y, \tau) \\
p_y &\mapsto p_y - \frac{qL}{\beta_0 P_0 c}\frac{\partial \phi}{\partial y}(x, y, \tau) \\
p_\tau &\mapsto p_\tau - \frac{qL}{\beta_0 P_0 c}\frac{\partial \phi}{\partial \tau}(x, y, \tau)
\end{aligned}
$$

This map can be generated from the Hamiltonian:

$$
H(x, y, \tau; s) = \frac{qL}{\beta_0 P_0 c}\phi\,(x, y, \tau)\,\delta(s)
$$

---

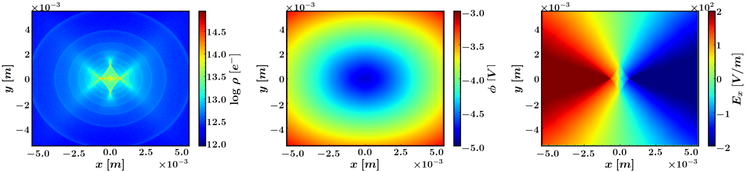
[2] Under usual thin-lens approximations.

[3] see G. Iadarola, CERN-ACC-NOTE-2019-0033.

The electron cloud simulation

$$H(x, y, \tau; s) = \frac{qL}{\beta_0 P_0 c} \phi(x, y, \tau)\, \delta(s)$$

- The potential ϕ can be calculated by PyECLOUD simulations over a discrete grid.



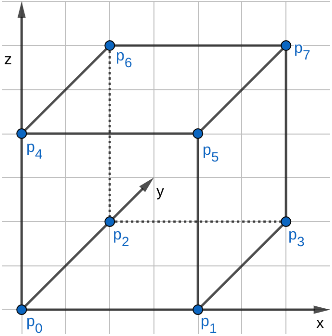
- To study slow effects, we should interpolate ϕ in a way that kicks are symplectic.
- Linear interpolation would not suffice[4]

[4] More details in K. Paraschou, Electron Cloud Meeting #67

How to interpolate

Objective

Given a regular 3D grid of any function $f^{i,j,k}$, we need to interpolate **locally** in a way that $\left\{ f, \frac{\partial f}{\partial x}, \frac{\partial f}{\partial y}, \frac{\partial f}{\partial z}, \frac{\partial^2 f}{\partial x \partial y}, \frac{\partial^2 f}{\partial x \partial z}, \frac{\partial^2 f}{\partial y \partial z} \right\}$ are continuous globally.



- Lekien and Marsden[5] proved that it is possible to meet this condition by using a tricubic interpolation scheme:

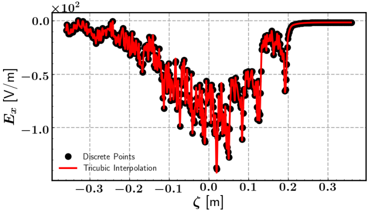
$$f(x, y, z) = \sum_{i=0}^{3} \sum_{j=0}^{3} \sum_{k=0}^{3} a_{ijk} x^i y^j z^k$$

- The 64 coefficients a_{ijk} change from cell to cell but required quantities stay continuous across cells.

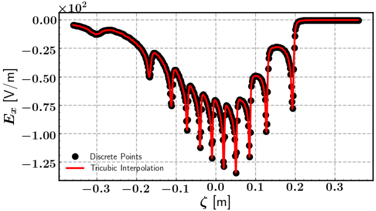
[5] Lekien, F & J. E., Marsden. (2005). Tricubic Interpolation in Three Dimensions. International Journal for Numerical Methods in Engineering. 63. 10.1002/nme.1296.

Interpolation of a PyECLOUD simulation

Individual simulation

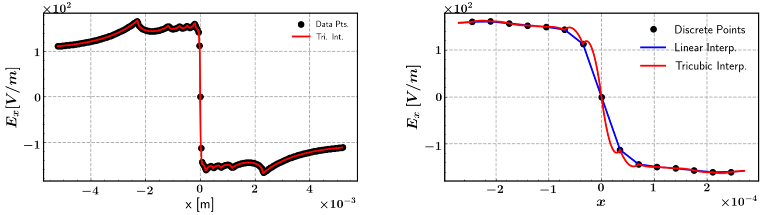


Average of 2000



- Simulation suffers from macroparticle noise.
- Solution: Reduce noise by averaging many simulations.
  - Averaging 2000 simulations reveals clear structure.

Interpolation artifacts



[zoom of left figure]

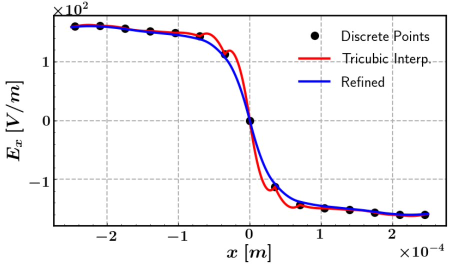
- Close look at interpolation reveals **irregularities**.

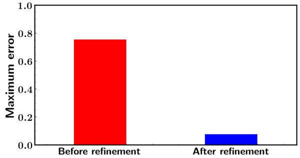
- Tricubic interpolation is symplectic but not accurate enough.
- Linear interpolation is more accurate (but not symplectic).
- Investigation with analytical potential pointed to the evaluation of derivatives (Finite Differences) as the problem.

Interpolation artifacts

To solve:

- Resolve Poisson's equation on an auxilliary finer grid
- to get better approximation of derivatives
- but keep information[6] only on original grid. (to limit memory consumption)





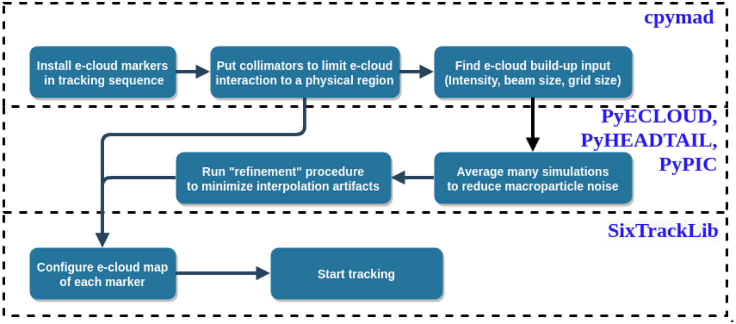
Maximum error* reduced by an order of magnitude.

*w.r.t high order interpolation of electric fields

[6] $\left\{\phi, \frac{\partial\phi}{\partial x}, \frac{\partial\phi}{\partial y}, \frac{\partial\phi}{\partial\tau}, \frac{\partial^2\phi}{\partial x\partial y}, \frac{\partial^2\phi}{\partial x\partial\tau}, \frac{\partial^2\phi}{\partial y\partial\tau}, \frac{\partial^3\phi}{\partial x\partial y\partial\tau}\right\}$, more details in K. Paraschou, Electron Cloud Meeting #72 and K. Paraschou, 165$^{\text{th}}$ HL-LHC WP2 Meeting

Overview

To set up a realistic simulation:



SixTrackLib Implementation

$$\frac{\partial \phi}{\partial \tilde{x}}(\tilde{x}, \tilde{y}, \tilde{\tau}) = \sum_{i=1}^{3}\sum_{j=0}^{3}\sum_{k=0}^{3} i a_{ijk} \tilde{x}^{i-1} \tilde{y}^{j} \tilde{\tau}^{k}, \qquad \mathrm{a} = \mathrm{B}^{-1}\,\mathrm{b}$$

We chose to implement the e-cloud map in SixTrackLib to take advantage of GPU tracking. The algorithm boils down to:

1. Store in memory the discrete potential ϕ
(example size: $500 \times 500 \times 500 \times 8 \times 8$ bytes $= 8$ **GB**).

SixTrackLib Implementation

$$\frac{\partial \phi}{\partial \tilde{x}}(\tilde{x}, \tilde{y}, \tilde{\tau}) = \sum_{i=1}^{3} \sum_{j=0}^{3} \sum_{k=0}^{3} i a_{ijk} \tilde{x}^{i-1} \tilde{y}^{j} \tilde{\tau}^{k}, \qquad \mathrm{a} = \mathrm{B}^{-1}\,\mathrm{b}$$

We chose to implement the e-cloud map in SixTrackLib to take advantage of GPU tracking. The algorithm boils down to:

1. Store in memory the discrete potential ϕ
   (example size: $500 \times 500 \times 500 \times 8 \times 8$ bytes $= 8$ **GB**).
2. Use particle's x, y, τ to find cell ($\tilde{x} = \frac{x - x_0}{dx}$).

SixTrackLib Implementation

$$\frac{\partial \phi}{\partial \tilde{x}}(\tilde{x}, \tilde{y}, \tilde{\tau}) = \sum_{i=1}^{3} \sum_{j=0}^{3} \sum_{k=0}^{3} i a_{ijk} \tilde{x}^{i-1} \tilde{y}^{j} \tilde{\tau}^{k}, \qquad \mathrm{a} = \mathrm{B}^{-1}\,\mathrm{b}$$

We chose to implement the e-cloud map in SixTrackLib to take advantage of GPU tracking. The algorithm boils down to:

1. Store in memory the discrete potential ϕ (example size: $500 \times 500 \times 500 \times 8 \times 8$ bytes $= 8$ **GB**).
2. Use particle's x, y, τ to find cell ($\tilde{x} = \frac{x - x_0}{dx}$).
3. From cell's 8 corners p_i, assemble vector $\mathrm{b} = \left(\phi|_{p_i}, \frac{\partial \phi}{\partial x}|_{p_i}, \frac{\partial \phi}{\partial y}|_{p_i}, \frac{\partial \phi}{\partial \tau}|_{p_i}, \frac{\partial^2 \phi}{\partial x \partial y}|_{p_i}, \frac{\partial^2 \phi}{\partial x \partial \tau}|_{p_i}, \frac{\partial^2 \phi}{\partial y \partial \tau}|_{p_i}, \frac{\partial^3 \phi}{\partial x \partial y \partial \tau}|_{p_i}, \ldots \right)$

SixTrackLib Implementation

$$\frac{\partial \phi}{\partial \tilde{x}}(\tilde{x}, \tilde{y}, \tilde{\tau}) = \sum_{i=1}^{3} \sum_{j=0}^{3} \sum_{k=0}^{3} i a_{ijk} \tilde{x}^{i-1} \tilde{y}^{j} \tilde{\tau}^{k}, \qquad \mathrm{a} = \mathrm{B}^{-1}\,\mathrm{b}$$

We chose to implement the e-cloud map in SixTrackLib to take advantage of GPU tracking. The algorithm boils down to:

1. Store in memory the discrete potential ϕ (example size: $500 \times 500 \times 500 \times 8 \times 8$ bytes $= 8$ **GB**).
2. Use particle's x, y, τ to find cell ($\tilde{x} = \frac{x - x_0}{dx}$).
3. From cell's 8 corners p_i, assemble vector $\mathrm{b} = \left(\phi|_{p_i}, \frac{\partial \phi}{\partial x}|_{p_i}, \frac{\partial \phi}{\partial y}|_{p_i}, \frac{\partial \phi}{\partial \tau}|_{p_i}, \frac{\partial^2 \phi}{\partial x \partial y}|_{p_i}, \frac{\partial^2 \phi}{\partial x \partial \tau}|_{p_i}, \frac{\partial^2 \phi}{\partial y \partial \tau}|_{p_i}, \frac{\partial^3 \phi}{\partial x \partial y \partial \tau}|_{p_i}, \ldots\right)$
4. Perform matrix multiplication $\mathrm{a} = \mathrm{B}^{-1}\,\mathrm{b}$ to find a_{ijk}.

SixTrackLib Implementation

$$\frac{\partial \phi}{\partial \tilde{x}}(\tilde{x}, \tilde{y}, \tilde{\tau}) = \sum_{i=1}^{3} \sum_{j=0}^{3} \sum_{k=0}^{3} i a_{ijk} \tilde{x}^{i-1} \tilde{y}^{j} \tilde{\tau}^{k}, \qquad \mathrm{a} = \mathrm{B}^{-1}\,\mathrm{b}$$

We chose to implement the e-cloud map in SixTrackLib to take advantage of GPU tracking. The algorithm boils down to:

1. Store in memory the discrete potential ϕ (example size: $500 \times 500 \times 500 \times 8 \times 8$ bytes $= 8$ **GB**).
2. Use particle's x, y, τ to find cell ($\tilde{x} = \frac{x - x_0}{dx}$).
3. From cell's 8 corners p_i, assemble vector $\mathrm{b} = \left(\phi|_{p_i}, \frac{\partial \phi}{\partial x}|_{p_i}, \frac{\partial \phi}{\partial y}|_{p_i}, \frac{\partial \phi}{\partial \tau}|_{p_i}, \frac{\partial^2 \phi}{\partial x \partial y}|_{p_i}, \frac{\partial^2 \phi}{\partial x \partial \tau}|_{p_i}, \frac{\partial^2 \phi}{\partial y \partial \tau}|_{p_i}, \frac{\partial^3 \phi}{\partial x \partial y \partial \tau}|_{p_i}, \ldots\right)$
4. Perform matrix multiplication $\mathrm{a} = \mathrm{B}^{-1}\,\mathrm{b}$ to find a_{ijk}.
5. Evaluate triple sum and apply kicks.

SixTrackLib Implementation

$$\frac{\partial \phi}{\partial \tilde{x}}(\tilde{x}, \tilde{y}, \tilde{\tau}) = \sum_{i=1}^{3} \sum_{j=0}^{3} \sum_{k=0}^{3} i a_{ijk} \tilde{x}^{i-1} \tilde{y}^{j} \tilde{\tau}^{k}, \qquad \mathrm{a} = \mathrm{B}^{-1}\,\mathrm{b}$$

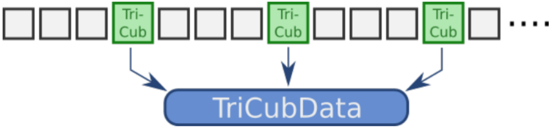
We chose to implement the e-cloud map in SixTrackLib to take advantage of GPU tracking. The algorithm boils down to:

1. Store in memory the discrete potential ϕ (example size: $500 \times 500 \times 500 \times 8 \times 8$ bytes $= 8$ **GB**).
2. Use particle's x, y, τ to find cell ($\tilde{x} = \frac{x - x_0}{dx}$).
3. From cell's 8 corners p_i, assemble vector $\mathrm{b} = \left(\phi|_{p_i}, \frac{\partial \phi}{\partial x}|_{p_i}, \frac{\partial \phi}{\partial y}|_{p_i}, \frac{\partial \phi}{\partial \tau}|_{p_i}, \frac{\partial^2 \phi}{\partial x \partial y}|_{p_i}, \frac{\partial^2 \phi}{\partial x \partial \tau}|_{p_i}, \frac{\partial^2 \phi}{\partial y \partial \tau}|_{p_i}, \frac{\partial^3 \phi}{\partial x \partial y \partial \tau}|_{p_i}, \dots \right)$
4. Perform matrix multiplication $\mathrm{a} = \mathrm{B}^{-1}\,\mathrm{b}$ to find a_{ijk}.
5. Evaluate triple sum and apply kicks.

SixTrackLib Implementation (2)

1. Store in memory the discrete potential ϕ (example size: $500 \times 500 \times 500 \times 8 \times 8$ bytes = **8 GB**).

- If each map needs multiple GBs of memory, it is prohibitive to include more interactions.
- Thanks to SixTrackLib's flexibility and the work of **Martin Schwinzerl**[7], each map(TriCub) can point to the same stored e-cloud potential(TriCubData).
- Possible to include as many e-cloud maps as we want.



[7] More details in M. Schwinzerl, BE Seminar on SixTrackLib

SixTrackLib Implementation (3)

4. Perform matrix multiplication $\mathbf{a} = \mathbf{B}^{-1}\mathbf{b}$ to find a_{ijk}.

```
for( i = 0; i < 64; i++ )
    for( j = 0; j < 64; j++ )
        a[i] += B[i][j] * b[j];
```

- B^{-1} is a constant, integer, sparse matrix of size 64×64 and a, b are vectors of size 64.
- Typical algorithm requires $64 \times 64 = 4096$ multiplications and additions, and $\sim 32\,KB$ memory.
- In GPUs, each "parallel processor" has its own local memory. If it runs out, it will occupy the memory of other processors.
- **Very important to minimize memory consumption!**

SixTrackLib Implementation (3)

4. Perform matrix multiplication $\mathbf{a} = \mathbf{B}^{-1}\,\mathbf{b}$ to find a_{ijk}.

- **Very important to minimize memory consumption!**
- Inspection of matrix B^{-1} reveals that only the numbers $\{1, 2, 3, 4, 6, 8, 9, 12, 18, 27\}$ appear.
- Developed code to write code that computes this **specific** matrix multiplication explicitly.
- $\rightarrow$ reduces local memory ($\sim 32\,KB$ to $1\,KB$), $\rightarrow$ reduces number of multiplications and additions (4096 to 1000).

```
for( i = 0; i < 64; i++ )
    for( j = 0; j < 64; j++ )
        a[i] += B[i][j] * b[j];
```

$\downarrow$

```
    coefs[0] = b[0];
    coefs[2] = -(tri_consts[1] * b[0]);
    coefs[3] = tri_consts[0] * b[0];
    coefs[8] = -(tri_consts[1] * b[0]);
    coefs[10] = tri_consts[5] * b[0];
    coefs[11] = -(tri_consts[3] * b[0]);
    coefs[12] = tri_consts[0] * b[0];
    coefs[14] = -(tri_consts[3] * b[0]);
    coefs[15] = tri_consts[2] * b[0];
    coefs[32] = -(tri_consts[1] * b[0]);
    coefs[34] = tri_consts[5] * b[0];
    coefs[35] = -(tri_consts[3] * b[0]);
    coefs[40] = tri_consts[5] * b[0];
    coefs[42] = -(tri_consts[8] * b[0]);
    coefs[43] = tri_consts[7] * b[0];
```

$\vdots$

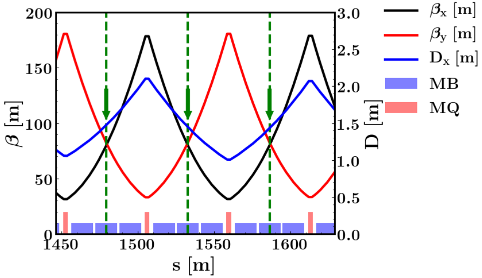
```
    coefs[58] += tri_consts[0] * b[62];
    coefs[59] -= b[62];
    coefs[61] += b[62];
    coefs[62] -= tri_consts[0] * b[62];
    coefs[63] += b[62];
    coefs[42] -= b[63];
    coefs[43] += b[63];
    coefs[46] += b[63];
    coefs[47] -= b[63];
    coefs[58] += b[63];
    coefs[59] -= b[63];
    coefs[62] -= b[63];
    coefs[63] += b[63];
```

E-cloud setup

E-cloud exists across the full length of the LHC beam pipe. Most significant contributors:

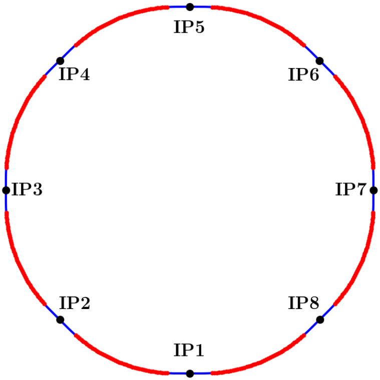
1. **E-cloud in arc dipoles (66%)**
2. E-cloud in quadrupoles (7%)

Place one interaction for each three dipoles.



- E-cloud buildup depends mildly on beam size.
- Beta functions and dispersion are the same.
- Effect due to change of beta functions (and dispersion) is ignored.

E-cloud setup

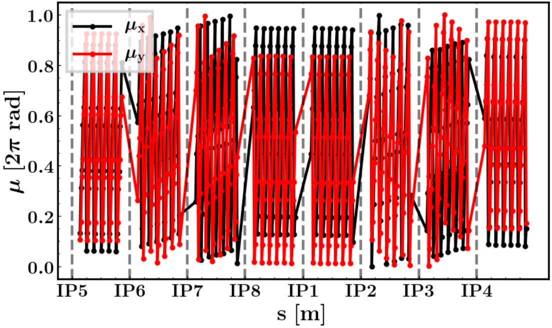


Install one e-cloud per half-cell
$\rightarrow$ 46 interactions per arc
$\rightarrow$ **368 interactions**.

Check in each interaction's location:

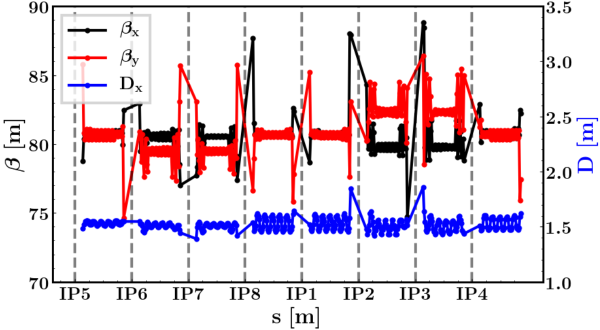
- phase advances $\mu_{x,y}$,
- beta functions $\beta_{x,y}$,
- horizontal dispersion D_x,
- beam sizes $\sigma_{x,y}$.

E-cloud setup - phase advances



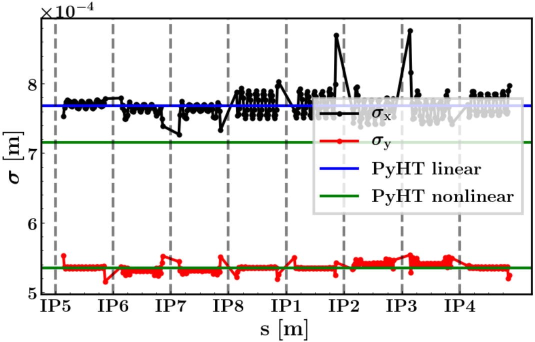
- Phase advances are distributed.
- Artificial resonance excitation is minimized.

E-cloud setup - $\beta_{x,y}$, D_x



- Beta functions and dispersion show only a minor beating.

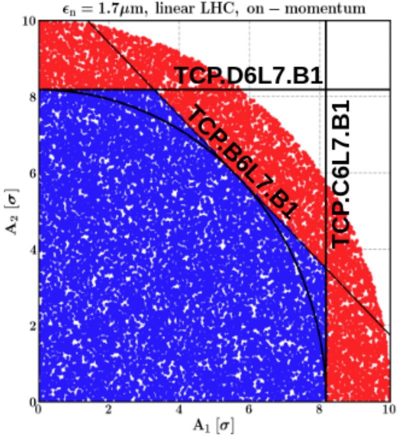
E-cloud setup - $\sigma_{x,y}$



- MAD-X slightly overestimates beam size[8] (7%) through the assumption of linear RF focusing.
- PyHEADTAIL used to check effect of non-linear RF focusing, small difference.

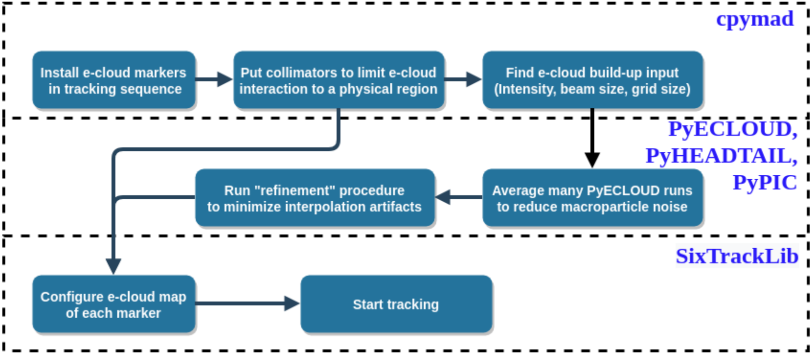
[8]Beam size defined as the standard deviation of a Gaussian distribution.

Collimators



- E-cloud does not exist outside the beam chamber.
- In fact, not enough memory to store fields across the full beam chamber.
- We use the three TCPs in IR7 to limit particles' oscillation in a realistic, physical region.

Overview



During configuration of the e-cloud maps:

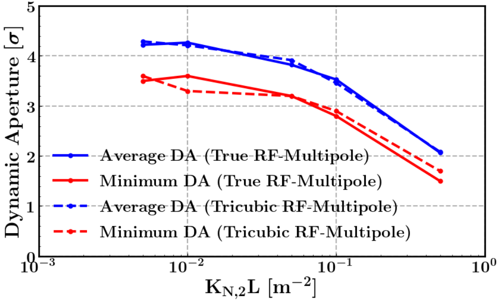
1. closed orbits (x, y, τ) are shifted to the center of the e-cloud,
2. dipolar kicks (p_x, p_y, p_τ) are subtracted.

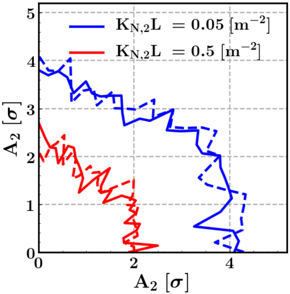
1. Recap
   - Motivation
   - Electron cloud kick
   - Tricubic interpolation
   - Macroparticle noise
   - Interpolation artifacts
2. Configuration of simulations
   - SixTrackLib implementation
   - Tracking sequence configuration
   - Collimators
3. Tests
   - RF-Multipole test
   - PyHEADTAIL footprint test
4. Tracking Results
   - Frequency Map Analysis
   - Dynamic Aperture
   - Long-term tracking (losses)

RF-Multipole test

To test the tricubic map,

1. We artificially introduced an RF-Multipole (sextupole) in the optics model of the LHC.
2. Using tricubic interpolation, we replicated the map of the RF-Multipole.
3. Comparison of dynamic aperture between real RF-Multipole map and replicated one.

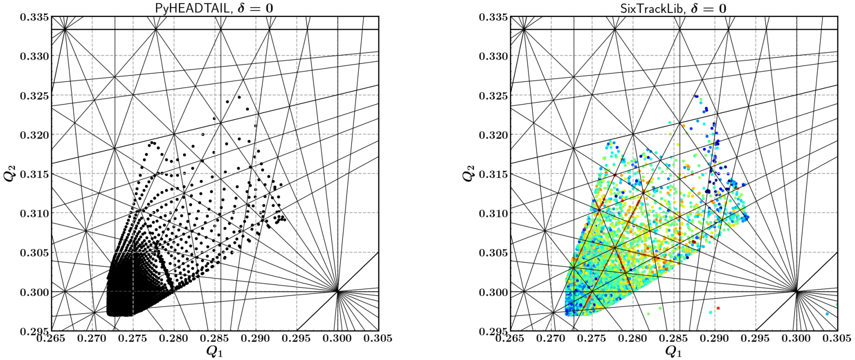




PyHEADTAIL footprint test

To test the tricubic map,

1. Use same map (e-cloud without magnetic field) in PyHEADTAIL (12 interactions, linear tracking) and SixTrackLib (368 interactions and non-linear tracking).
2. Footprints are the same (tune shift depends only beta functions)



Resources

After all the necessary configuration, we can use GPUs to track:

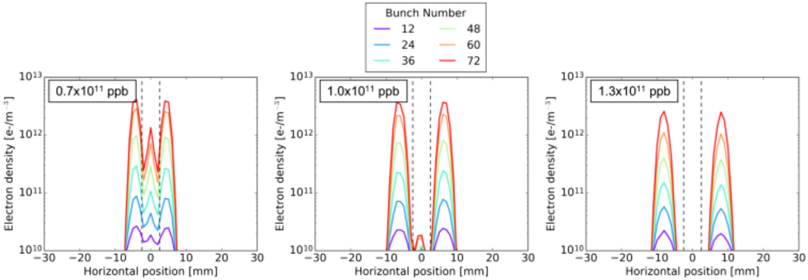
- $\sim$**20 Nvidia Tesla V100**[7] GPUs available in HTCondor at CERN.
- **4 Nvidia Tesla V100** GPUs in the CNAF cluster in Bologna (through HL-LHC collaboration).
- **4 Nvidia Titan V** GPUs of LIU and ABP (50/50) for the purpose of developing and testing GPU-based simulation codes (Many thanks to H. Bartosik)

Three types of simulations:

1. Frequency Map Analysis (20k turns) -> $\sim$ 30 *mins*
2. Dynamic Aperture (1M turns) -> $\sim$ 8 *hours*
3. Long-term tracking, losses (20M turns) -> $\sim$ 4 *days*

[7] https://www.nvidia.com/en-us/data-center/tesla-v100/

E-cloud density



[A. Romano, PRAB 21, 061002 (2018)]

E-cloud build-up in dipolar fields has the characteristic of forming two stripes of electrons in the beam chamber.

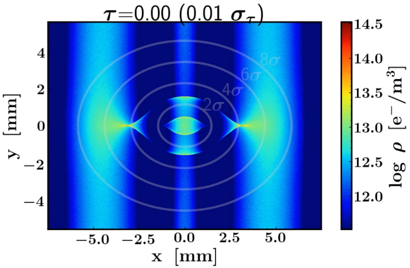
- Higher intensities make stripes move to larger distances.
- Lower intensities bring stripes closer to the beam's center.
- At lower intensities a stripe of electron forms in the center.

Simulations

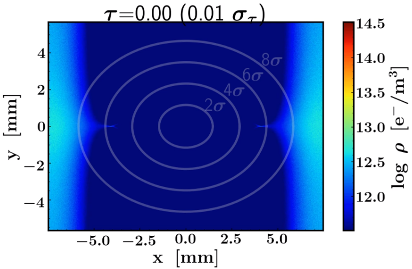
Comparison between:

- No e-cloud, $I_{MO} = 40\ A$
- Arc dipoles' e-cloud, $SEY = 1.35$, $0.7 \cdot 10^{11} ppb$, $I_{MO} = 40\ A$
- Arc dipoles' e-cloud, $SEY = 1.35$, $1.2 \cdot 10^{11} ppb$, $I_{MO} = 40\ A$

$0.7 \cdot 10^{11} ppb$



$1.2 \cdot 10^{11} ppb$

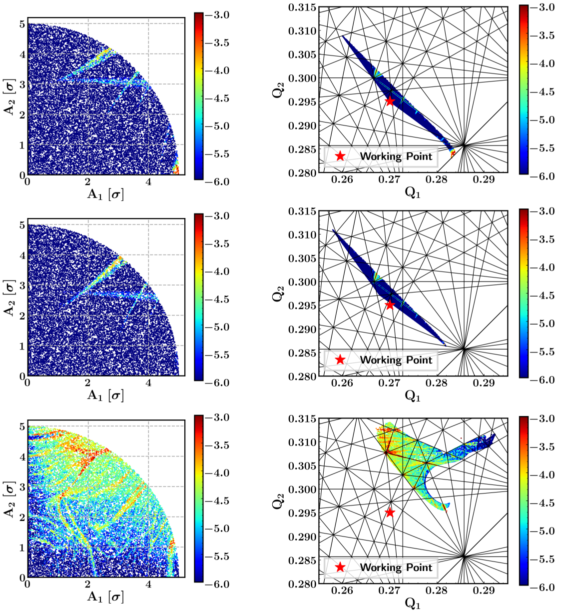


Frequency Map Analyses

No e-cloud

$1.2 \cdot 10^{11}$ppb

$0.7 \cdot 10^{11}$ppb



No central stripe

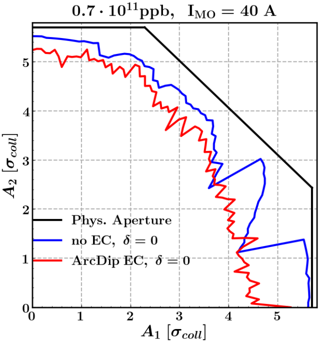
→ Tune-shift

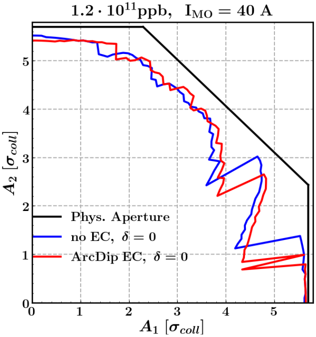
Central stripe

→ non-linear detuning

→ resonances

Dynamic Aperture (DA)

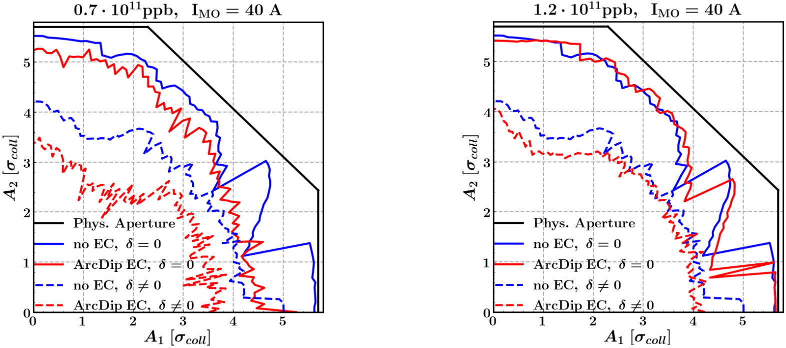




On-momentum:

- No central stripe ($1.2 \cdot 10^{11}$ppb) $\rightarrow$ **Almost no effect.**
- Central stripe ($0.7 \cdot 10^{11}$ppb) $\rightarrow$ Some reduction at large horizontal amplitudes.

Dynamic Aperture (DA)



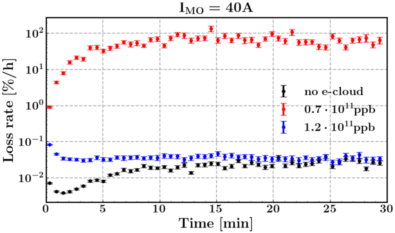
On-momentum:

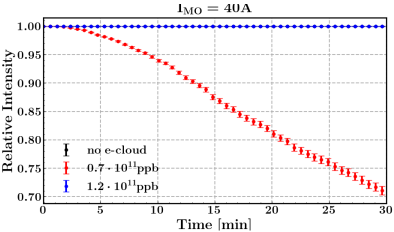
- No central stripe ($1.2 \cdot 10^{11}$ppb) $\rightarrow$ **Almost no effect.**
- Central stripe ($0.7 \cdot 10^{11}$ppb) $\rightarrow$ Some reduction at large horizontal amplitudes.

Off-momentum:

- No central stripe ($1.2 \cdot 10^{11}$ppb) $\rightarrow$ Small reduction in DA
- Central stripe ($0.7 \cdot 10^{11}$ppb) $\rightarrow$ **Significant reduction.**

Long-term tracking (losses)





No central stripe ($1.2 \cdot 10^{11}$ppb) $\rightarrow$ Slightly larger losses than without e-cloud

Central stripe ($0.7 \cdot 10^{11}$ppb) $\rightarrow$ **Very significant losses**, e-cloud may not be representative (constant SEY = 1.35).

- These simulations were performed mostly for testing purposes to check the sanity of the simulation method.

Summary

Conclusion:

- Fully defined the procedure to simulate slow e-cloud effects.
- **Implemented, optimized and tested** the map in SixTrackLib. (Tests with RF-Multipole, and comparison with PyHEADTAIL)
- Performed FMA, DA and long-term losses simulations.

On-going work:

- Introduced e-clouds in **arc quadrupoles (MQs)**.
- Better probe effect of SEY, tunes, intensity etc.
- Compare simulations with experimental data.

Summary

Conclusion:

- Fully defined the procedure to simulate slow e-cloud effects.
- **Implemented, optimized and tested** the map in SixTrackLib. (Tests with RF-Multipole, and comparison with PyHEADTAIL)
- Performed FMA, DA and long-term losses simulations.

On-going work:

- Introduced e-clouds in **arc quadrupoles (MQs)**.
- Better probe effect of SEY, tunes, intensity etc.
- Compare simulations with experimental data.

Thank you for your attention!

Konstantinos Paraschou