Status of the studies on Electron cloud incoherent effects

Konstantinos Paraschou\textsuperscript{1,2}, Giovanni Iadarola\textsuperscript{1}

\textsuperscript{1}CERN, Switzerland
\textsuperscript{2}Aristotle University of Thessaloniki, Greece

\textbf{Acknowledgements:} R. De Maria, N. Karastathis, J. Malewicz, Y. Papaphilippou, L. Sabato, M. Schwinzerl

\textbf{HL-LHC Work Package 2 meeting}
CERN, Tuesday, 10\textsuperscript{th} December 2019
PyECLoUD-PyHEA DTAIL simulations

Beams are sliced in $\zeta = -\beta ct$, which is basically time!
We want to modify the kicks to the beam particles to support long-term tracking (symplecticity).
Electron cloud kick

It is possible to prove\(^1,2\) that e-cloud kick can be written as the gradient of a scalar potential:

\[
\begin{align*}
  x, y, \zeta &\mapsto x, y, \zeta \\
  p_x &\mapsto p_x - \frac{qL}{\beta^2 \gamma mc^2} \frac{\partial \phi}{\partial x}(x, y, \zeta) \\
  p_y &\mapsto p_y - \frac{qL}{\beta^2 \gamma mc^2} \frac{\partial \phi}{\partial y}(x, y, \zeta) \\
  \delta &\mapsto \delta - \frac{qL}{\beta^2 \gamma mc^2} \frac{\partial \phi}{\partial \zeta}(x, y, \zeta)
\end{align*}
\]

This map can be generated from the Hamiltonian:

\[
H(x, y, \zeta; s) = \frac{qL}{\beta^2 \gamma mc^2} \phi(x, y, \zeta) \delta(s)
\]

---

\(^1\)Under usual thin-lens approximations.

\(^2\)see G. Iadarola, CERN-ACC-NOTE-2019-0033.
The electron cloud simulation

\[ H(x, y, \zeta; s) = \frac{qL}{\beta^2 \gamma mc^2} \phi(x, y, \zeta) \delta(s) \]

- The potential \( \phi \) can be calculated by PyECLoud simulations over a discrete grid.

- To apply kicks on arbitrary beam particle we need to interpolate \( \phi \) in a way that preserves symplecticity.
Symplecticity of a 6D kick

The conditions for a symplectic thin-lens kick are:

\[
\begin{align*}
\frac{\partial}{\partial x} \left( \frac{\partial \phi}{\partial y} \right) &= \frac{\partial}{\partial y} \left( \frac{\partial \phi}{\partial x} \right) \\
\frac{\partial}{\partial x} \left( \frac{\partial \phi}{\partial \zeta} \right) &= \frac{\partial}{\partial \zeta} \left( \frac{\partial \phi}{\partial x} \right) \\
\frac{\partial}{\partial y} \left( \frac{\partial \phi}{\partial \zeta} \right) &= \frac{\partial}{\partial \zeta} \left( \frac{\partial \phi}{\partial y} \right)
\end{align*}
\]

By Schwarz’s theorem this is verified if the mixed derivatives \( \left( \frac{\partial^2 \phi}{\partial x \partial y}, \frac{\partial^2 \phi}{\partial x \partial \zeta}, \frac{\partial^2 \phi}{\partial y \partial \zeta} \right) \) exist and are continuous.
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 the following quantities are continuous globally:

$$\left\{ f, \frac{\partial f}{\partial x}, \frac{\partial f}{\partial y}, \frac{\partial f}{\partial z}, \frac{\partial^2 f}{\partial x \partial y}, \frac{\partial^2 f}{\partial x \partial z}, \frac{\partial^2 f}{\partial y \partial z} \right\}$$

Lekien and Marsden 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.

---

Application to an e-cloud
Pinch is noisy

Simulation of the pinch still suffers from macroparticle noise.

Solution: Reduce noise by averaging many pinches.

- Averaging 2000 pinches reveals clear structure.
Interpolation issues

- At first sight interpolation looks proper
- Closer look reveals **irregularities**.

- Tricubic is **symplectic** but not accurate enough.
- Linear interpolation would be more accurate (**but not symplectic**).
Investigation using analytical potential

\[ \phi = \frac{x}{2} - \log (1 + e^x) , \quad E = \frac{1}{1 + e^x} - \frac{1}{2} \]

Calculating \( E = -\partial_x \phi \) with Finite Differences we observe the same irregularities when interpolating.
Investigation using analytical potential

\[ \phi = \frac{x}{2} - \log (1 + e^x), \quad E = \frac{1}{1 + e^x} - \frac{1}{2} \]

- Calculating \( E = -\partial_x \phi \) with Finite Differences we observe the same irregularities when interpolating.
- When using the exact values of the derivative to interpolate, error is significantly reduced.
Investigation using analytical potential

\[ \phi = \frac{x}{2} - \log(1 + e^x), \quad E = \frac{1}{1 + e^x} - \frac{1}{2} \]

- Calculating \( E = -\partial_x \phi \) with Finite Differences we observe the same irregularities when interpolating.
- When using the exact values of the derivative to interpolate, error is significantly reduced.

For e-cloud we don't have the exact derivatives but we can try to improve our way to approximate them.
Refinement procedure

We use Poisson’s equation to improve $\phi$ and its derivatives. For each $\zeta$-slice:

1. $\rho, \phi$ known on a coarse grid.
Refinement procedure

We use Poisson’s equation to improve $\phi$ and its derivatives. For each $\zeta$-slice:

1. $\rho, \phi$ known on a coarse grid.
2. Create a finer grid.
Refinement procedure

We use Poisson’s equation to improve $\phi$ and its derivatives. For each $\zeta$-slice:

1. $\rho, \phi$ known on a coarse grid.
2. Create a finer grid.
3. Linearly interpolate $\phi$-boundary.
4. 
5. 
6. 
7. 
Refinement procedure

We use Poisson’s equation to improve $\phi$ and its derivatives. For each $\zeta$-slice:

1. $\rho, \phi$ known on a coarse grid.
2. Create a finer grid.
3. Linearly interpolate $\phi$-boundary.
4. Linearly interpolate $\rho$ inside.
We use Poisson’s equation to improve $\phi$ and its derivatives. For each $\zeta$-slice:

1. $\rho, \phi$ known on a coarse grid.
2. Create a finer grid.
3. Linearly interpolate $\phi$-boundary.
4. Linearly interpolate $\rho$ inside.
5. Re-solve Poisson equation on finer grid.

At this point, $\phi$ solution is much smoother.
Refinement procedure

We use Poisson’s equation to improve $\phi$ and its derivatives.

For each $\zeta$-slice:

1. $\rho, \phi$ known on a coarse grid.
2. Create a finer grid.
3. Linearly interpolate $\phi$-boundary.
4. Linearly interpolate $\rho$ inside.
5. Re-solve Poisson equation on finer grid.
6. Use Finite differences on finer grid to calculate derivatives:
   \[
   \left\{ \phi, \frac{\partial \phi}{\partial x}, \frac{\partial \phi}{\partial y}, \frac{\partial \phi}{\partial \zeta}, \frac{\partial^2 \phi}{\partial x \partial y}, \frac{\partial^2 \phi}{\partial x \partial \zeta}, \frac{\partial^2 \phi}{\partial y \partial \zeta}, \frac{\partial^3 \phi}{\partial x \partial y \partial \zeta} \right\}.
   \]
Refinement procedure

We use Poisson’s equation to improve $\phi$ and its derivatives.

For each $\zeta$-slice:

1. $\rho, \phi$ known on a coarse grid.
2. Create a finer grid.
3. Linearly interpolate $\phi$-boundary.
4. Linearly interpolate $\rho$ inside.
5. Re-solve Poisson equation on finer grid.
6. Use Finite differences on finer grid to calculate derivatives:
   \[ \left\{ \phi, \frac{\partial \phi}{\partial x}, \frac{\partial \phi}{\partial y}, \frac{\partial \phi}{\partial \zeta}, \frac{\partial^2 \phi}{\partial x \partial y}, \frac{\partial^2 \phi}{\partial x \partial \zeta}, \frac{\partial^2 \phi}{\partial y \partial \zeta}, \frac{\partial^3 \phi}{\partial x \partial y \partial \zeta} \right\}. \]
7. Keeping the finer grid for all (500) slices would be impractical ($\sim 100$ GBs).
Refinement procedure

We use Poisson’s equation to improve \( \phi \) and its derivatives.

For each \( \zeta \)-slice:

1. \( \rho, \phi \) known on a coarse grid.
2. Create a finer grid.
3. Linearly interpolate \( \phi \)-boundary.
4. Linearly interpolate \( \rho \) inside.
5. Re-solve Poisson equation on finer grid.
6. Use Finite differences on finer grid to calculate derivatives:
   \[
   \{ \phi, \frac{\partial \phi}{\partial x}, \frac{\partial \phi}{\partial y}, \frac{\partial \phi}{\partial \zeta}, \frac{\partial^2 \phi}{\partial x \partial y}, \frac{\partial^2 \phi}{\partial x \partial \zeta}, \frac{\partial^2 \phi}{\partial y \partial \zeta}, \frac{\partial^3 \phi}{\partial x \partial y \partial \zeta} \}.
   \]
7. Keep refined \( \phi \) and derivatives on the original grid.

Keeping the finer grid for all (500) slices would be impractical (~ 100 GBs).
Irregularities are significantly suppressed through this procedure.

Maximum error\(^4\) is up to an order of magnitude smaller with just
\[ dx \leftarrow dx/2, \]
\[ dy \leftarrow dy/2, \]
\[ d\zeta \leftarrow d\zeta/2. \]

\(^4\)More details in K. Paraschou, Electron Cloud Meeting \#72
Impact of irregularities

- Simple tracking of a linear machine and one e-cloud symplectic kick.
- Very important to minimize irregularities.
- By reducing them, there is significant impact on the beam particle motion.
Computational requirements

- Grid in PyECLOUD already pushes the limits of a typical RAM.
- Significant development to optimize memory consumption during the refinement procedure.
- Even then, solution of Poisson equation on such a fine grid can easily exceed 100 GBs of RAM memory.
Computational requirements

- Grid in PyECLOUD already pushes the limits of a typical RAM.
- Significant development to optimize memory consumption during the refinement procedure.
- Even then, solution of Poisson equation on such a fine grid can easily exceed 100 GBs of RAM memory.

Used special resources:

- LIUPSGPU machine of ABP with 24 cores and 256 GB RAM.
- HTCondor nodes (BigMem) with 24 cores and 1 TB RAM.
- HTCondor nodes (BigMem) with 48 cores and 512 GB RAM.
Status and next steps

1. Tracking in a realistic machine with e-cloud interactions will be **computationally intensive** → requires significant speed-up.

2. We are porting our implementation into the SixTrackLib\(^5\) code, which also will allow profiting of GPUs to push tracking speed.
   - Code practically ready
   - Being tested on GPUs in Bologna

---

\(^5\)M. Schwinzerl, R. De Maria, “SixTrackLib: Current Status, Future Directions”
Beam-Beam studies in SixTrackLib with GPUs
SixTrackLib on GPUs for beam-beam studies

To prepare the ground for these studies, we made some experience using SixTrackLib on GPUs to simulate losses driven by beam-beam (in the context of the summer student project of Julie Malewicz).

We considered the following problem:

- At beginning of collisions ($t < 1 \text{ h}$), strong losses are observed within the first hour.
- We want to check whether this behaviour can be observed in simulations.
SixTrackLib on GPUs for beam-beam studies

Realistic LHC simulations are configured in python using cpymad\(^6\) (python interface to MAD-X):

- Import **sequence of elements** from **MAD-X description**.

\(^6\)https://github.com/hibtc/cpymad
SixTrackLib on GPUs for beam-beam studies

Realistic LHC simulations are configured in python using cpymad\(^6\) (python interface to MAD-X):

- Import **sequence of elements** from **MAD-X description**.
- **Beam-Beam configuration** (relative position of the two beams, beam sizes) using newly developed tools (Gianni, Riccardo).

\(^6\)https://github.com/hibtc/cpymad
SixTrackLib on GPUs for beam-beam studies

Realistic LHC simulations are configured in python using cpymad\(^6\) (python interface to MAD-X):

- **Import sequence of elements** from MAD-X description.
- **Beam-Beam configuration** (relative position of the two beams, beam sizes) using newly developed tools (Gianni, Riccardo).
- **Crosschecked against Sixtrack:** Footprints and Dynamic Aperture are fully consistent.

\(^6\)https://github.com/hibtc/cpymad
Some Numbers

Available GPUs:

- **12 Nvidia Tesla V100** GPUs available in HTCondor at CERN.
- **4 Nvidia Tesla V100** GPUs in the CNAF cluster in Bologna (through HL-LHC collaboration).

Tracking $10^4$ particles in the full LHC lattice with Beam-Beam:

- in **one** V100 GPU:
  - $\sim 5.4$ hours for $10^6$ turns
  - $\rightarrow \sim 4.5$ days for $20 \cdot 10^6$ turns (0.5 hours of beam time)!

For the study we tracked $6 \cdot 10^4$ particles using 6 GPUs.

---

Simulation method and results

- Transverse phase space is probed using a random uniform sampling.
- Repeated for different values of the longitudinal amplitude to cover the RF bucket.
- Losses are obtained by assigning to each particle a weight proportional to the corresponding phase space density for a Gaussian bunch.
Simulation method and results

- The simulation shows a transient in the losses comparable to the one observed in the machine.
- Magnitude of losses is very similar between simulation and measurement.
- In simulation, aperture restriction changes the first part. Later on, curves are alike.
Conclusion

Status:
- We know how to conserve the symplecticity of the e-cloud interaction (tricubic interpolation).
- We know how to mitigate macro-particle noise of PyECLoud simulations (average over several pinches).
- We know how to minimize artifacts of interpolation scheme (derivatives evaluation on refined grid).

Next steps:
- Finalize and test e-cloud lenses in SixTrackLib.
- Benchmark behaviour of “long-term” observables on analytic $\zeta$-dependent Hamiltonians (Crab Cavity multipoles).
- Build model for the e-clouds in the arcs of the LHC and study losses and emittance at injection energy.

Thank you for your attention!

Konstantinos Paraschou
Conclusion

Status:

- We know how to conserve the symplecticity of the e-cloud interaction (tricubic interpolation).
- We know how to mitigate macro-particle noise of PyECloud simulations (average over several pinches).
- We know how to minimize artifacts of interpolation scheme (derivatives evaluation on refined grid).

Next steps:

- Finalize and test e-cloud lenses in SixTrackLib.
- Benchmark behaviour of “long-term” observables on analytic $\zeta$-dependent Hamiltonians (Crab Cavity multipoles).
- Build model for the e-clouds in the arcs of the LHC and study losses and emittance at injection energy.

Thank you for your attention!

Konstantinos Paraschou