Machine Learning–Driven Start-to-End Simulation and Optimisation of the PALLAS Laser-Plasma Accelerator

K. Cassou, M. Lenivenko - CNRS/IJClab

A. Sytov – INFN Ferrara Division

A. Huber – University of Bordeaux

Laser-plasma acceleration

- Electron acceleration using plasma Wakefield
- 1000 times higher acceleration field than in conventional accelerators
- Laser-plasma acceleration consists of many nonlinear processes

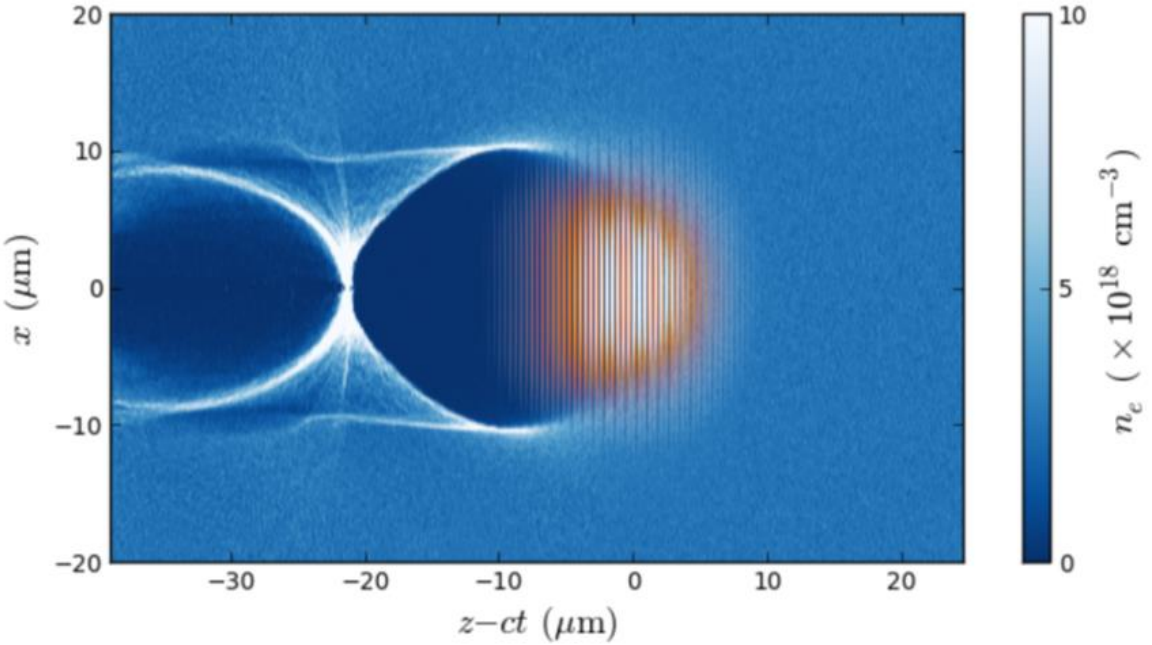
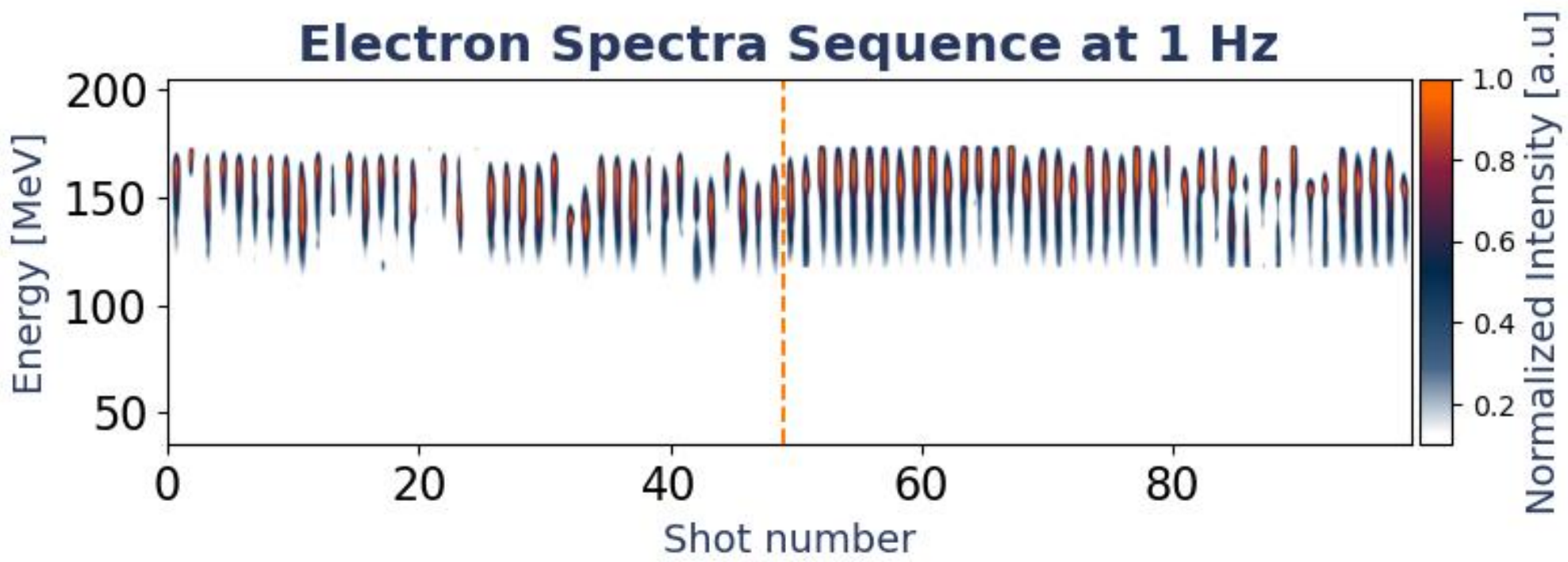


Image taken from R. Lehe PhD Thesis
https://pastel.hal.science/tel-01088398v1

Laser-plasma acceleration

Main challenges:

- **Shot-to-shot stability**
- Beam quality at all aspects at the same time



PALLAS – commissioning July 2025

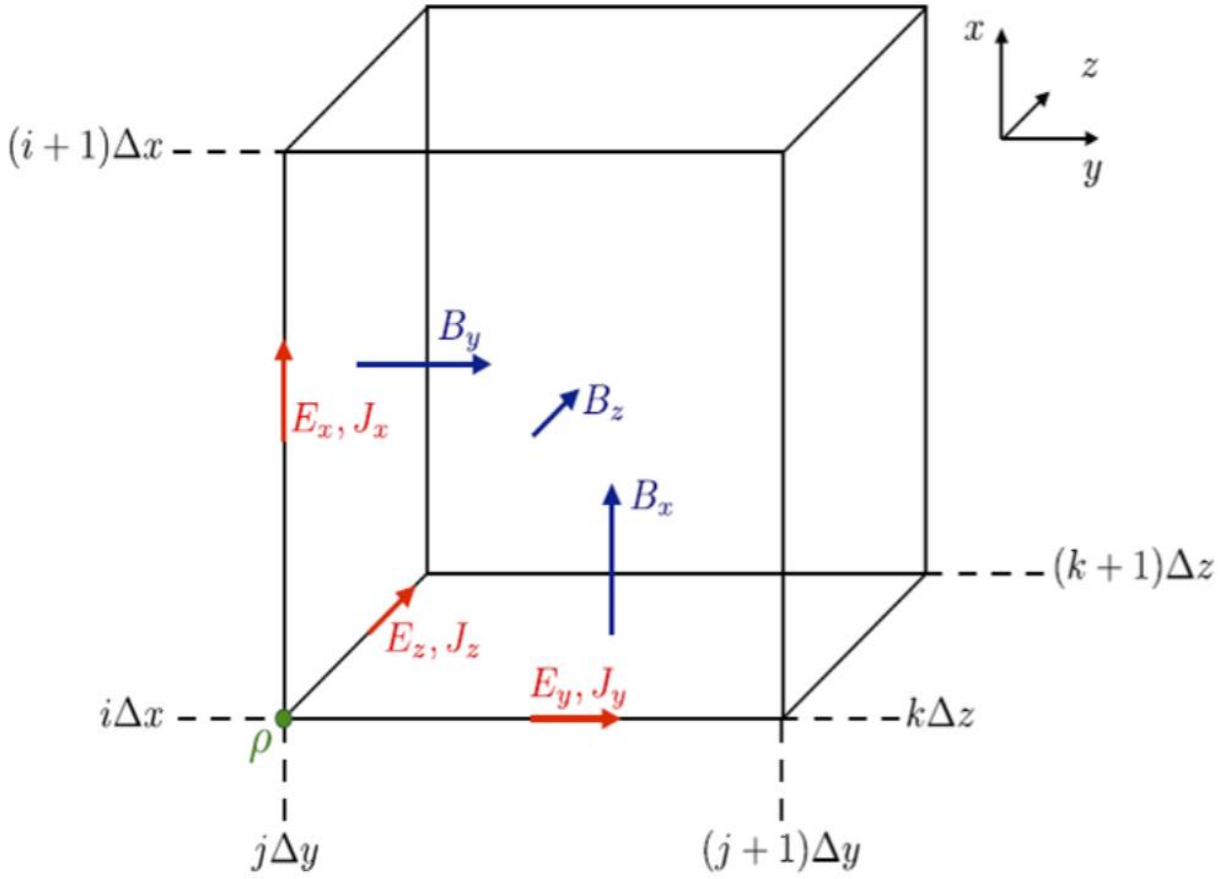
Particle-in-Cell(PIC) simulations

PIC simulations (Particle-In-Cell) :

- Very accurate but substantial computational resources needed
- Numerical challenge(Maxwell-Vlasov set of equations)

In PALLAS PIC simulations are performed with **Smilei** code [1,2]

Simplified low-fidelity simulation takes 30 minutes on 240 cores. High-fidelity simulation can take a week.



[1] J. Derouillat et al. Comput. Phys. Commun. 222, 351-373 (2018)
[2] https://smileipic.github.io/Smilei/

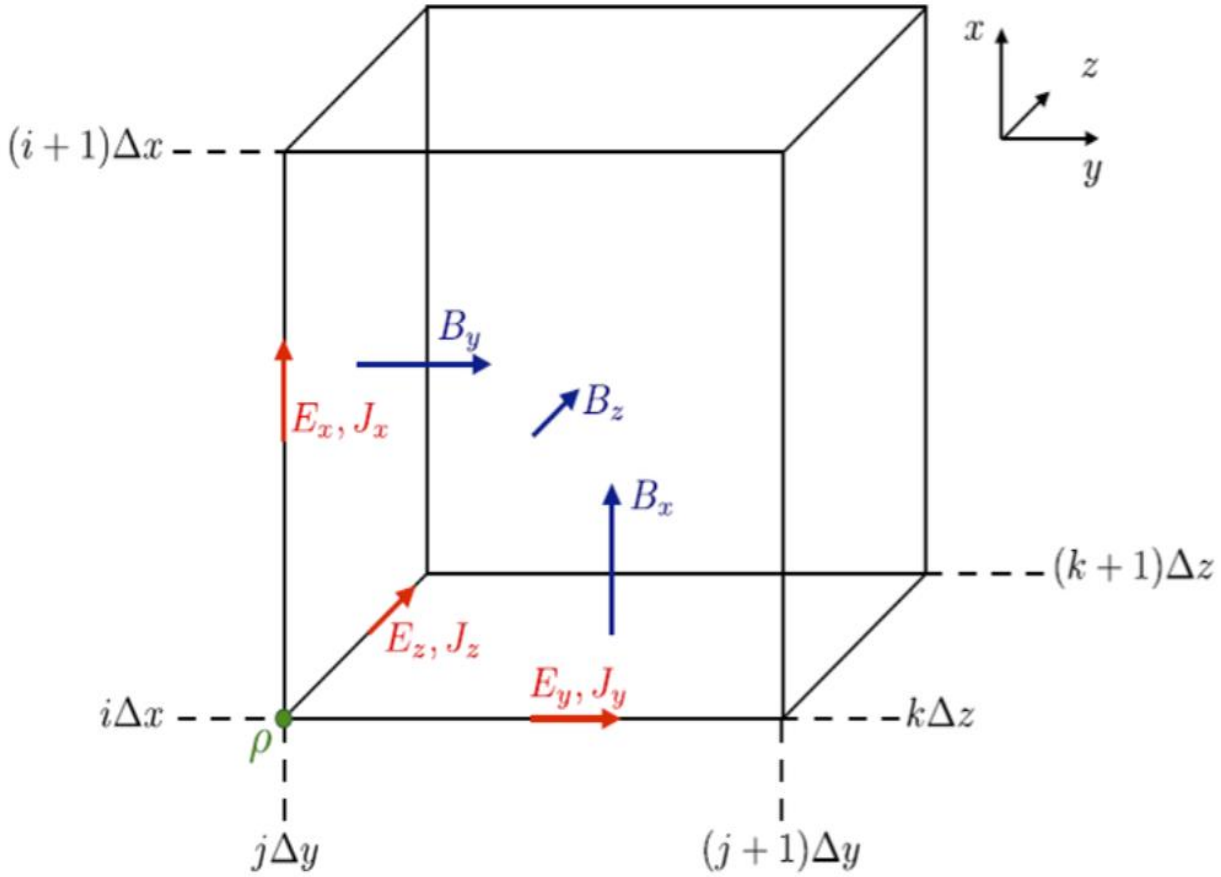
Particle-in-Cell(PIC) simulations

PIC simulations (Particle-In-Cell) :

- Very accurate but substantial computational resources needed
- Numerical challenge(Maxwell-Vlasov set of equations)

In PALLAS PIC simulations are performed with **Smilei** code [1,2]

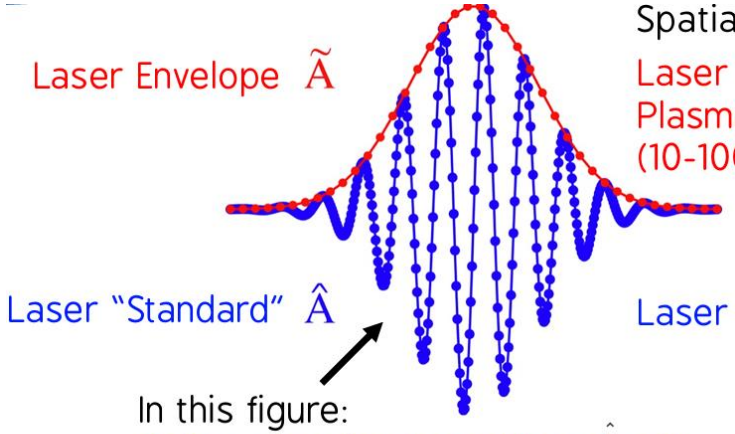
Simplified low-fidelity simulation takes 30 minutes on 240 cores. High-fidelity simulation can take a week.



PIC is SLOW

[1] J. Derouillat et al. Comput. Phys. Commun. 222, 351-373 (2018)
[2] https://smileipic.github.io/Smilei/

The Need for Speed leads to Machine Learning?



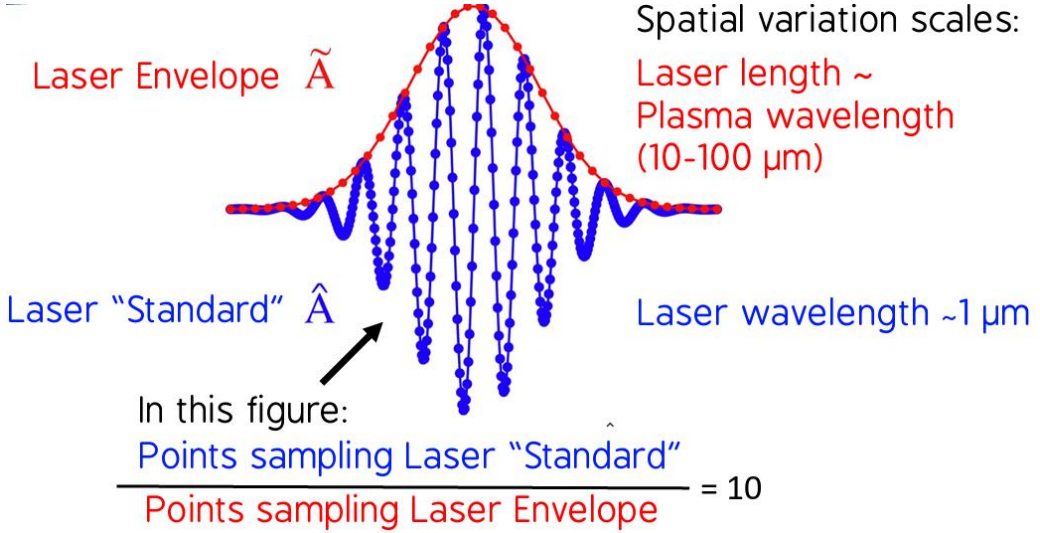
F. Massimo et al., Phys. Rev. E (2020)

L = 800 μm	AM cylindrical (2 modes) Simulation time: 9.3 kcpu-hours — Standard laser	AM cylindrical (1 mode) Simulation time: 30 minutes, 1 cpu-core — Envelope, 1 particle per cell
Q [pC]	175	179
$2\sigma_x$ [μm]	3.4	3.5
$2\sigma_y$ [μm]	2.3	2.4
$2\sigma_z$ [μm]	1.1	1.2
$\varepsilon_{n,y}$ [mm-mrad]	3.9	4.0
$\varepsilon_{n,z}$ [mm-mrad]	1.2	1.2
E_{avg} [MeV]	90.2	89.6
σ_E/E [rms, %]	11.91	11.52
	Tunnel ionization	Envelope tunnel ionization

The Need for Speed leads to Machine Learning?

PIC slowness leads to:

- Difficult to explore large parameter spaces (nonlinear, multiparametric, coupled physical processes)
- Not viable for full start-to-end simulations
- Machine Learning use case



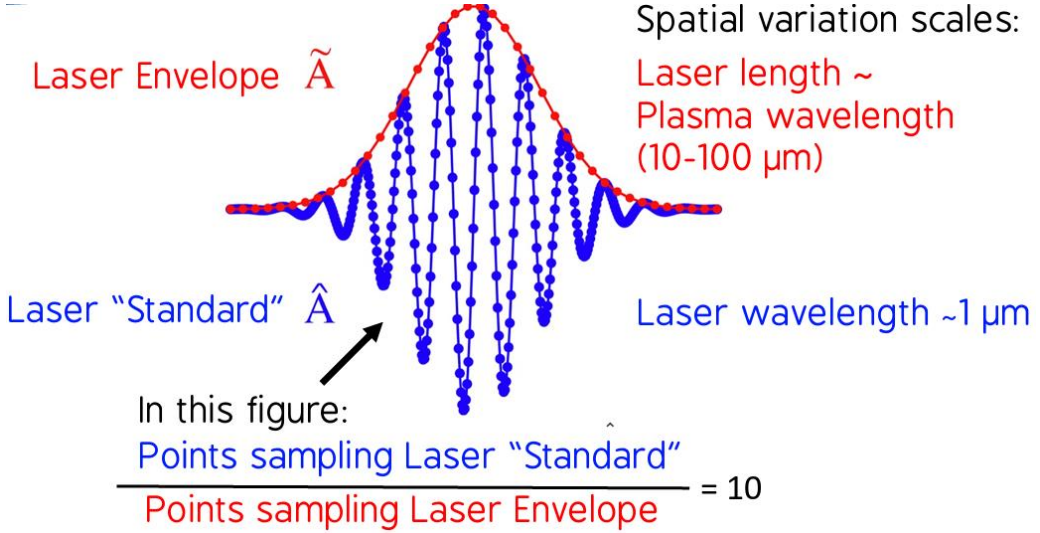
F. Massimo et al., Phys. Rev. E (2020)	AM cylindrical (2 modes) Simulation time: 9.3 kcpu-hours	AM cylindrical (1 mode) Simulation time: 30 minutes, 1 cpu-core
L = 800 μm	Standard laser	Envelope, 1 particle per cell
Q [pC]	175	179
$2\sigma_x$ [μm]	3.4	3.5
$2\sigma_y$ [μm]	2.3	2.4
$2\sigma_z$ [μm]	1.1	1.2
$\varepsilon_{n,y}$ [mm-mrad]	3.9	4.0
$\varepsilon_{n,z}$ [mm-mrad]	1.2	1.2
E_{avg} [MeV]	90.2	89.6
σ_E/E [rms, %]	11.91	11.52
	Tunnel ionization	Envelope tunnel ionization

The Need for Speed leads to Machine Learning?

PIC slowness leads to:

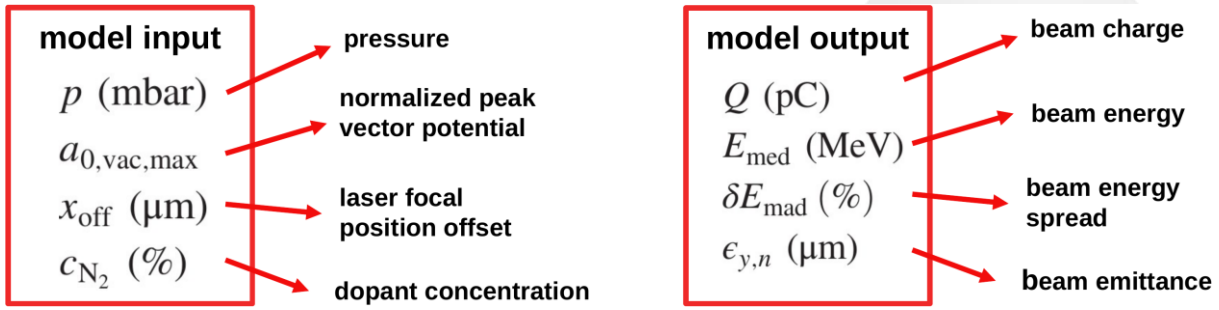
- Difficult to explore large parameter spaces (nonlinear, multiparametric, coupled physical processes)
- Not viable for full start-to-end simulations
- Machine Learning use case

Machine learning model training time depends on the model complexity, but evaluation is instant!



F. Massimo et al., Phys. Rev. E (2020)	AM cylindrical (2 modes) Simulation time: 9.3 kcpu-hours	AM cylindrical (1 mode) Simulation time: 30 minutes, 1 cpu-core
L = 800 μm	Standard laser	Envelope, 1 particle per cell
Q [pC]	175	179
$2\sigma_x$ [μm]	3.4	3.5
$2\sigma_y$ [μm]	2.3	2.4
$2\sigma_z$ [μm]	1.1	1.2
$\varepsilon_{n,y}$ [mm-mrad]	3.9	4.0
$\varepsilon_{n,z}$ [mm-mrad]	1.2	1.2
E_{avg} [MeV]	90.2	89.6
σ_E/E [rms, %]	11.91	11.52
	Tunnel ionization	Envelope tunnel ionization

ML Direct model: “Digital twin”

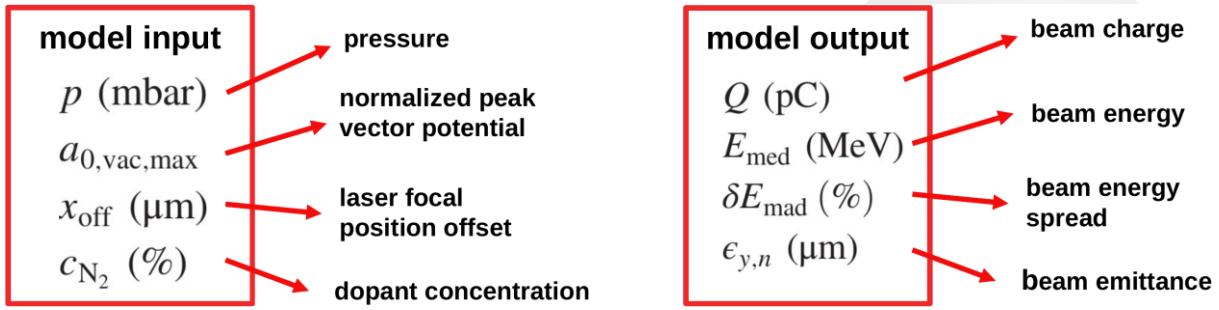


ML model trained [1] on the low-fidelity PIC simulations. The model was implemented and trained using the *Keras API* with a *TensorFlow* backend. Key information:

- Trained on 3600 simulations
- Tested on the 12000 simulations
- Statistical accuracy more then 96%

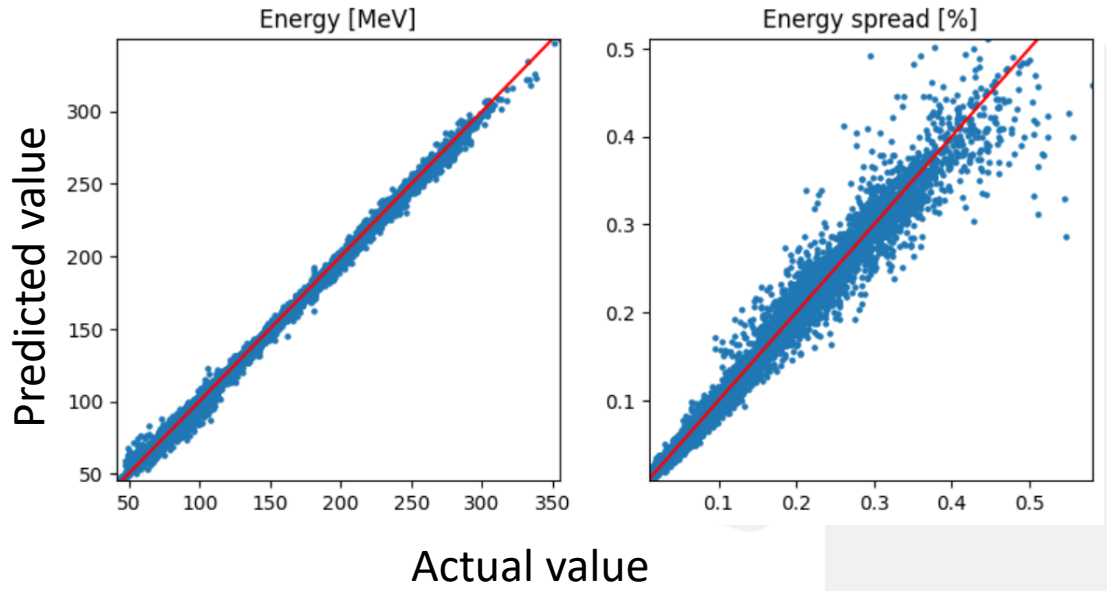
[1]: Gueladio Kane, Surrogate modelling for laser plasma acceleration, GdR SCIPAC 2025

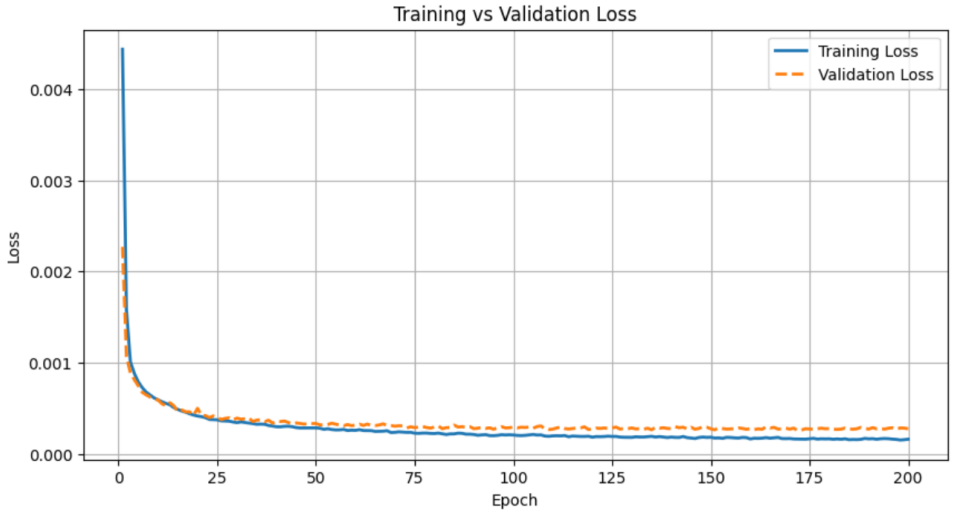
ML Direct model: “Digital twin”



ML model trained [1] on the low-fidelity PIC simulations. The model was implemented and trained using the *Keras API* with a *TensorFlow* backend. Key information:

- Trained on 3600 simulations
- Tested on the 12000 simulations
- Statistical accuracy more then 96%

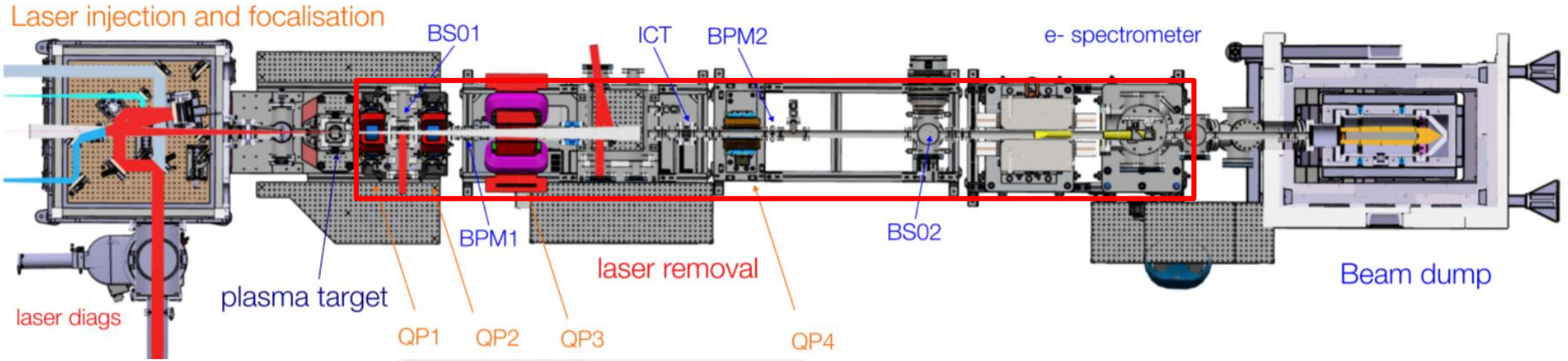




[1]: Gueladio Kane, Surrogate modelling for laser plasma acceleration, GdR SCIPAC 2025

The Next Challenge: Beam Transport

The beam from the plasma source is not yet usable. It must be captured, focused, and transported



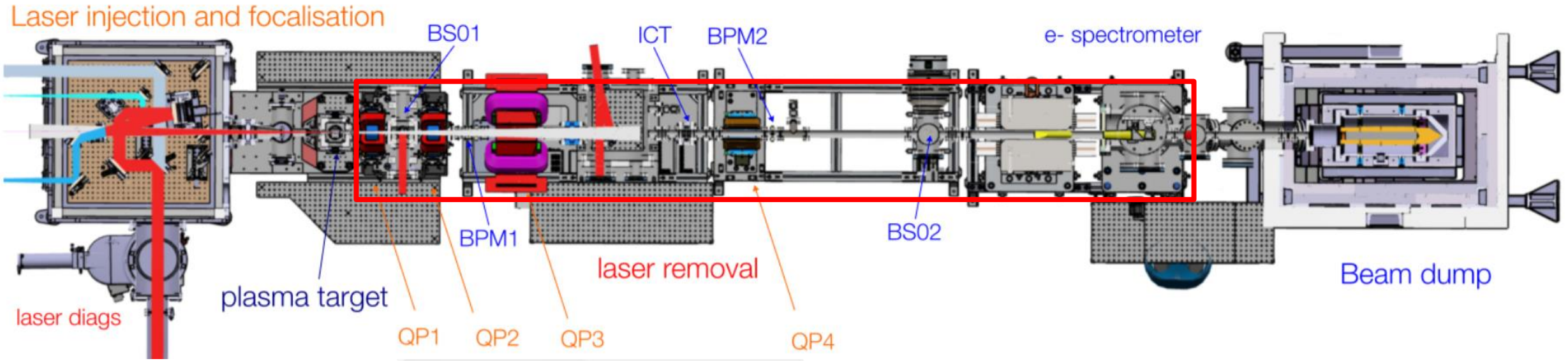
We simulate electron beamline propagation using 2 different approaches:

- **RF-Track tracking code [1]**, includes collective effects
- **Geant4** start-to-end simulation made by our colleagues A. Sytov (INFN-Fe) and A. Huber (LP2IB) [2]

[1]: RF-track https://gitlab.cern.ch/rf-track part of CERN Xsuite: https://github.com/xsuite/xsuite
[2]: GEANT4 simulation repository https://gitlab.in2p3.fr/pallas/design/-/tree/main/ecbl/G4_Simulation?ref_type=heads
[3]: https://github.com/xopt-org/Xopt

The Next Challenge: Beam Transport

The beam from the plasma source is not yet usable. It must be captured, focused, and transported



We simulate electron beamline propagation using 2 different approaches:

- **RF-Track tracking code [1]**, includes collective effects
- **Geant4** start-to-end simulation made by our colleagues A. Sytov (INFN-Fe) and A. Huber (LP2IB) [2]

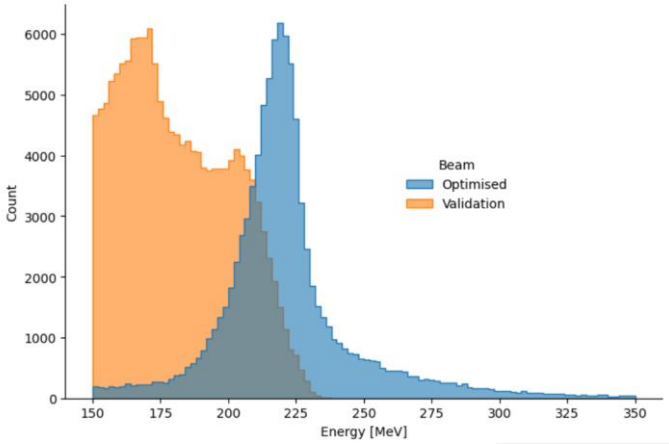
We want to optimise electron beam arriving to the BS02(screen) and/or e-spectrometer using Bayesian optimisation (BO) using Xopt library [3]

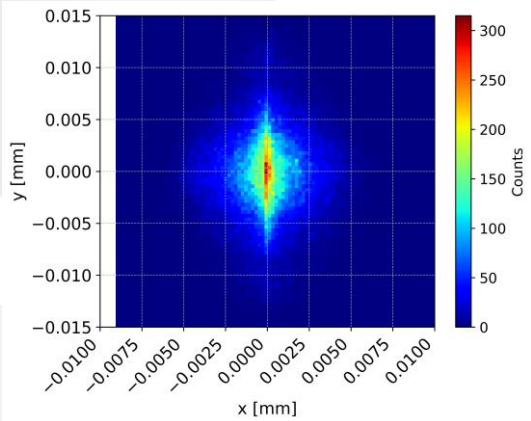
[1]: RF-track https://gitlab.cern.ch/rf-track part of CERN Xsuite: https://github.com/xsuite/xsuite
[2]: GEANT4 simulation repository https://gitlab.in2p3.fr/pallas/design/-/tree/main/ecbl/G4_Simulation?ref_type=heads
[3]: https://github.com/xopt-org/Xopt

Optimisation using RF-Track

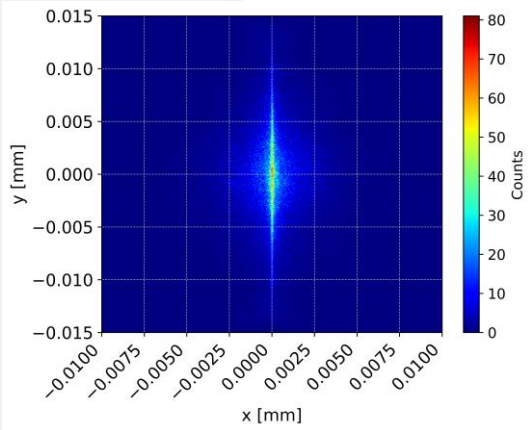
In the RF-Track optimisation case we use following configuration:

- Quadrupoles gradients and Quadrupoles positions as variables
- Constrain on the 90% transportation





Optimisation beam



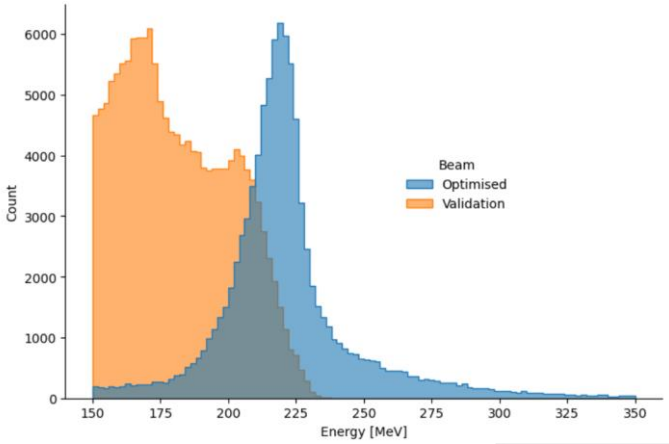
Validation beam

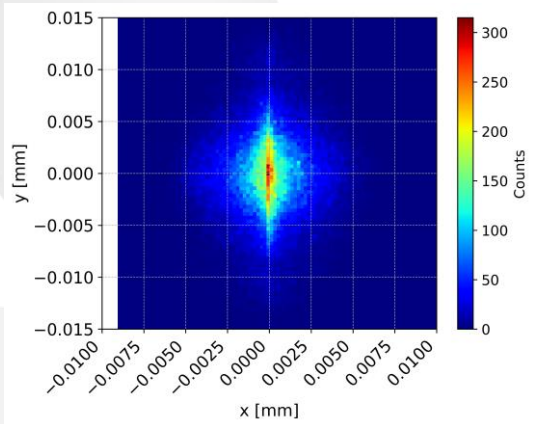
After plasma cell

Optimisation using RF-Track

In the RF-Track optimisation case we use following configuration:

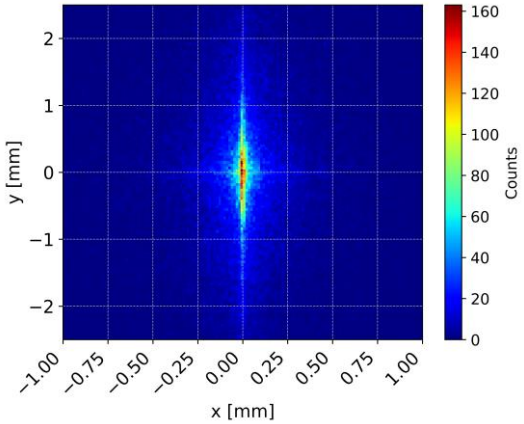
- Quadrupoles gradients and Quadrupoles positions as variables
- Constrain on the 90% transportation

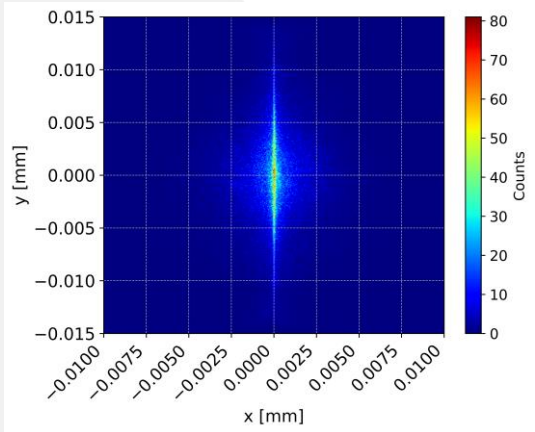




Optimisation beam

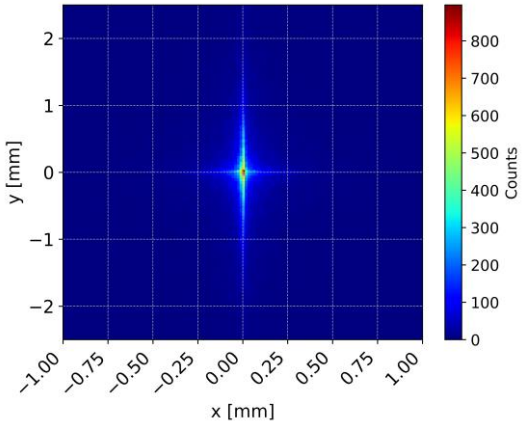
BO





Validation beam

3.3 m



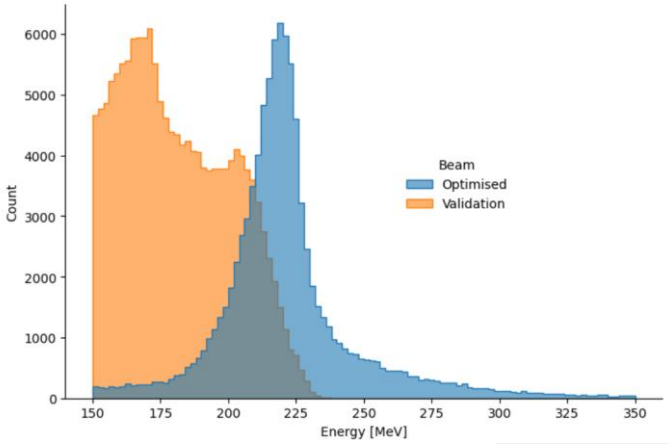
After plasma cell

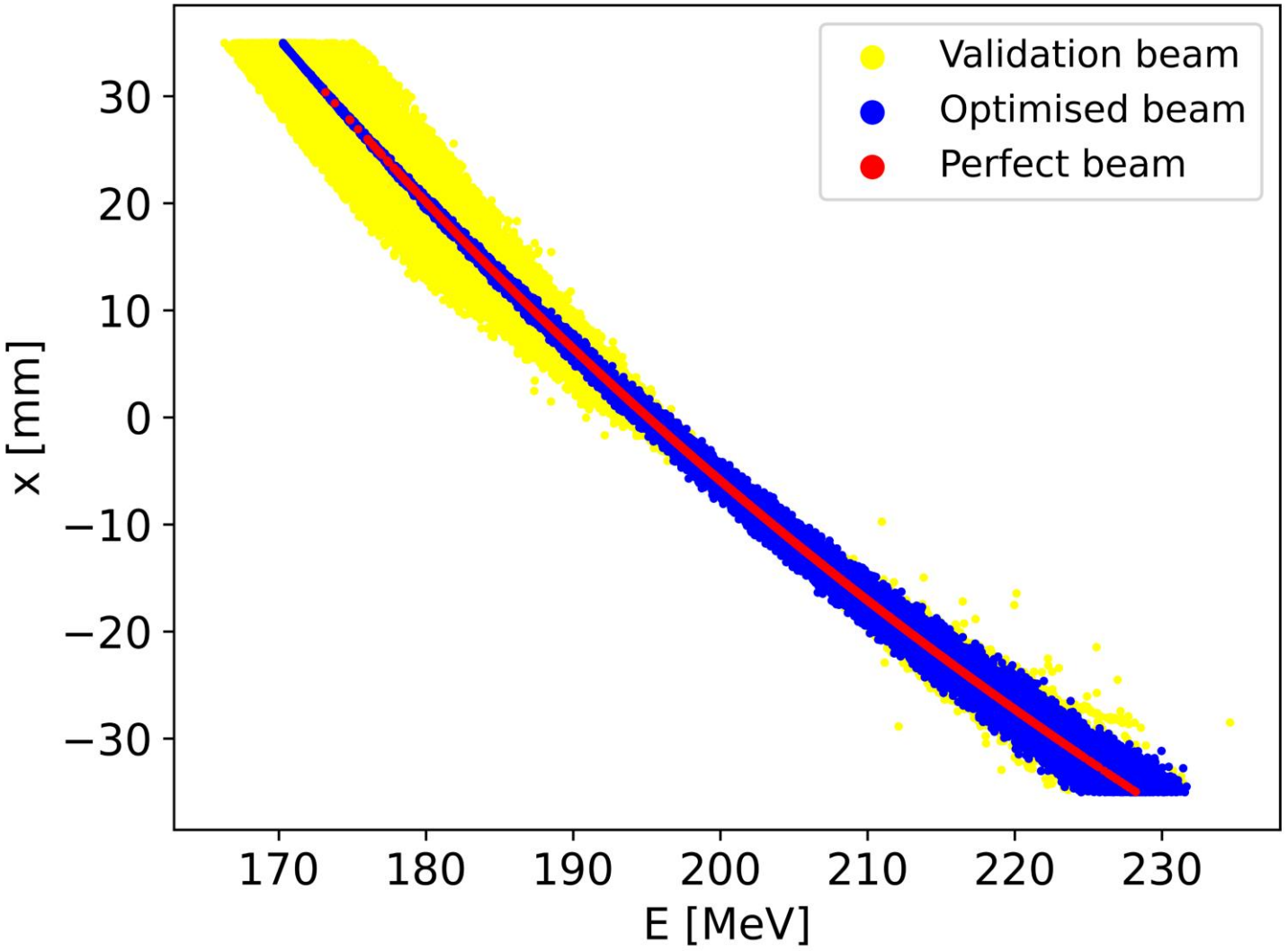
At the entrance of the dipole

Optimisation using RF-Track

In the RF-Track optimisation case we use following configuration:

- Quadrupoles gradients and Quadrupoles positions as variables
- Constrain on the 90% transportation

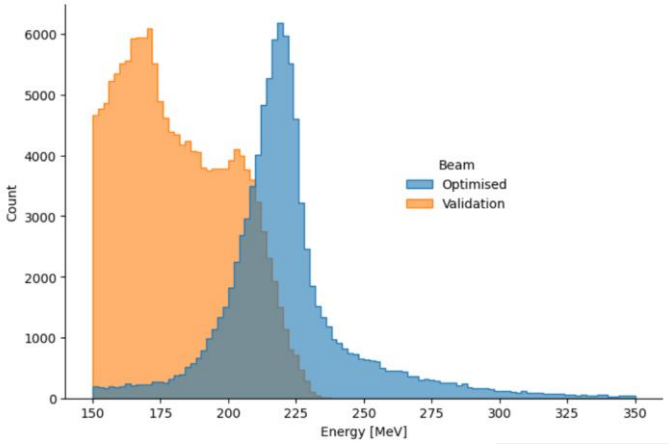


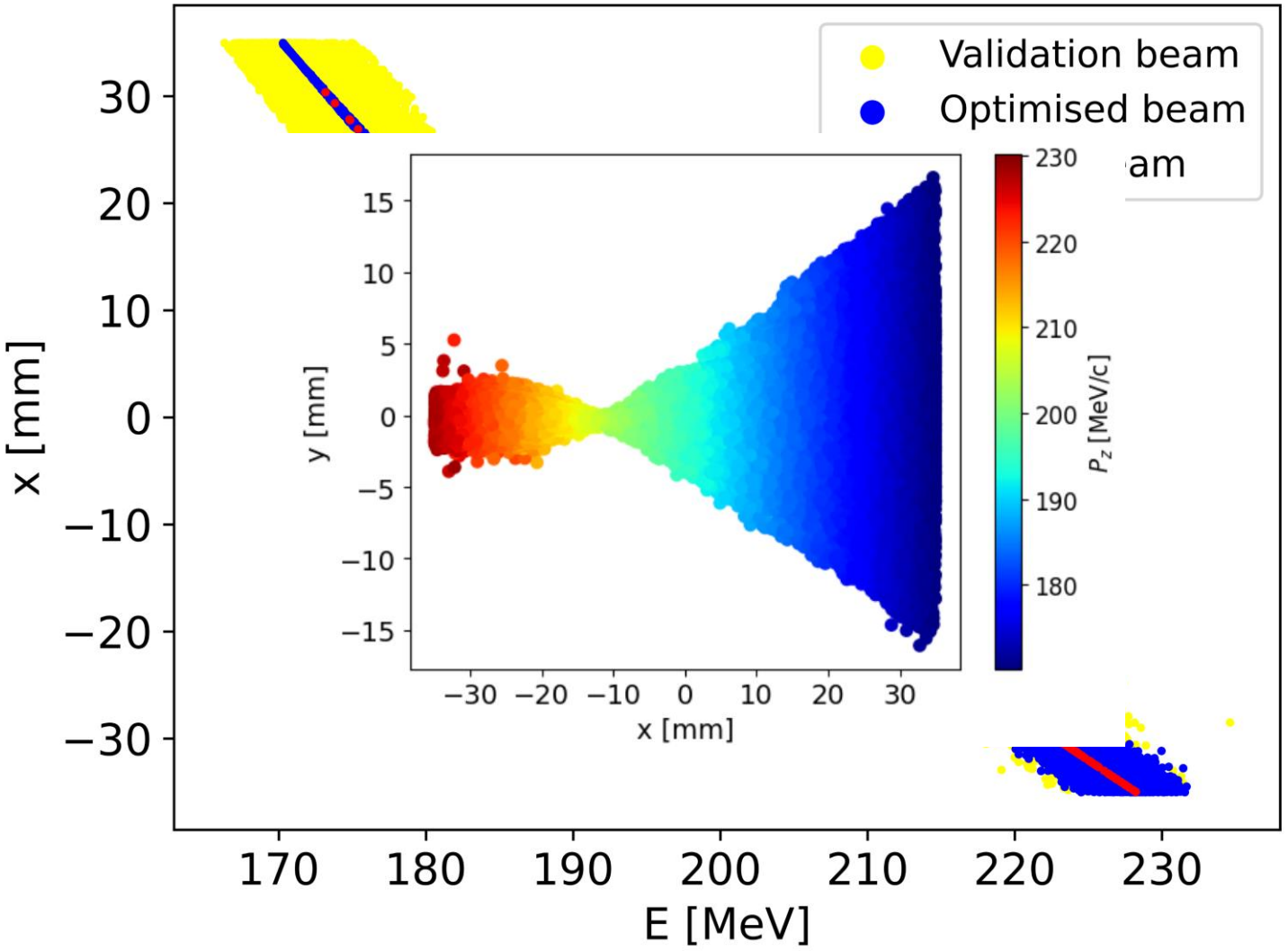


Optimisation using RF-Track

In the RF-Track optimisation case we use following configuration:

- Quadrupoles gradients and Quadrupoles positions as variables
- Constrain on the 90% transportation

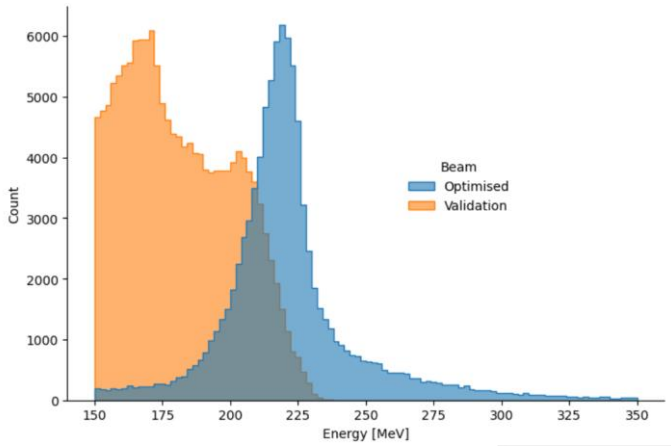




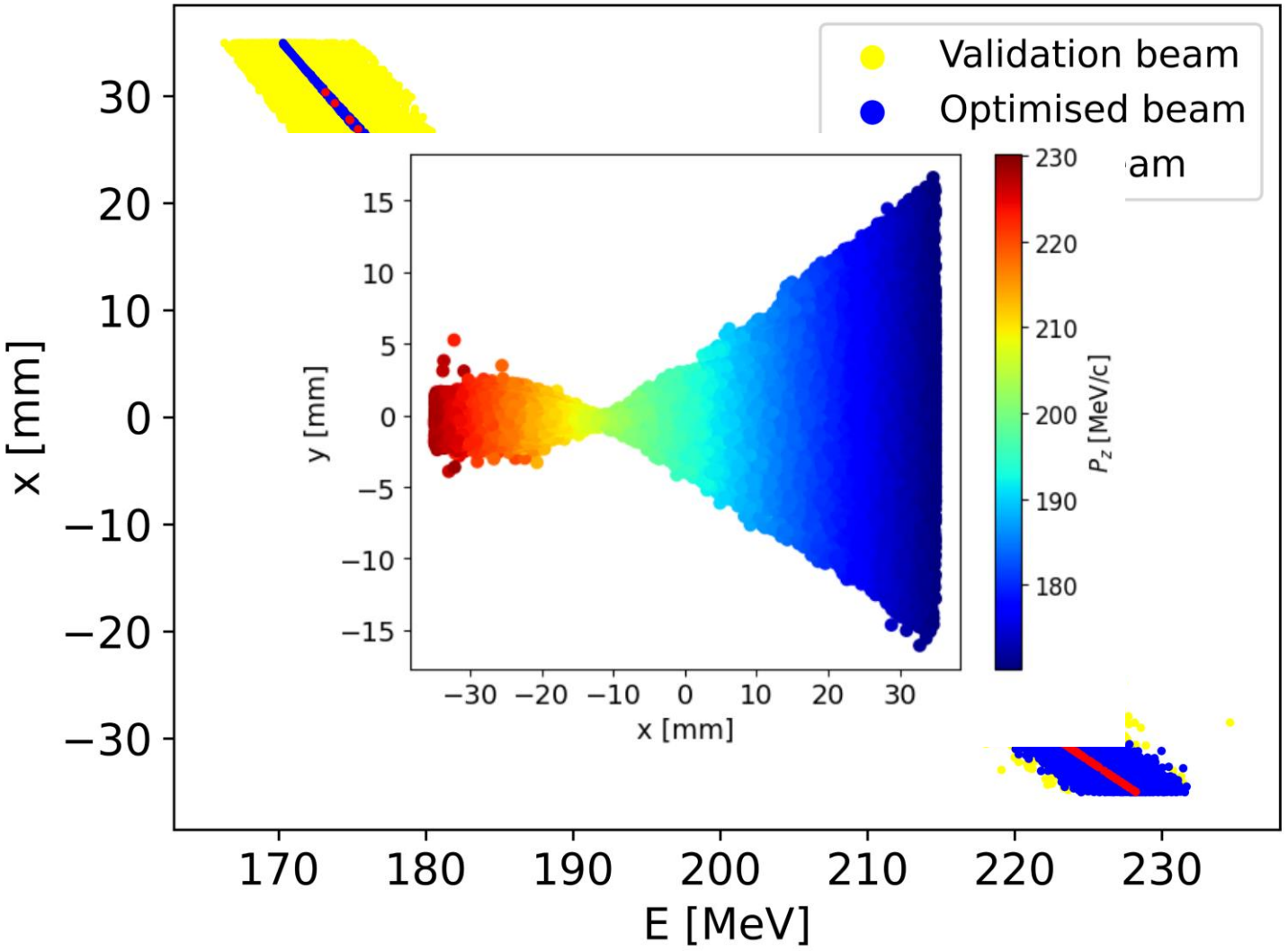
Optimisation using RF-Track

In the RF-Track optimisation case we use following configuration:

- Quadrupoles gradients and Quadrupoles positions as variables
- Constrain on the 90% transportation



Optimised configurations achieved in 15 minutes



Integration of ML model into GEANT4 simulation

Context:

Laser-plasma accelerators are compact sources of high-energy electrons (PALLAS experiment & LWFA acceleration)

Simulations are needed to understand and optimize these systems

Problems:

Full-scale simulations are very resource-intensive (PIC models)

Objective:

Integrate ML-based models into Geant4 to efficiently simulate experiments

Implementation of Geant4 simulation

- ML model is implemented into **G4ParticleGun** [1] using **ONNX**[2] interface.
- Multithreading on CPU is possible.
- ML model input parameters are changeable with Macro file.

```
#ifndef B1PrimaryGeneratorAction_h
#define B1PrimaryGeneratorAction_h 1

#include "G4VUserPrimaryGeneratorAction.hh"
#include "globals.hh"
#include "G4GeneralParticleSource.hh"
#include "G4ParticleGun.hh"
#include <memory>
#include <onnxruntime_c_api.h>
#include <onnxruntime_cxx_api.h>
class G4ParticleGun;
class G4Event;
```

Keras function (Python) -> **tf2onnx.convert.from_keras** -> ONNX file (C++)

```
/gun/SetStatusONNX true
/laser/SetOffsetLaserFocus 1680 ## um
/laser/SetNormVecPotential 1.23
/laser/SetFracDopTargetChamber 0.0617 # %(/100)
/laser/SetPressure 47.8 #mbar
```

```
G4double fXof_min = -399.8247;
G4double fXof_max = 1798.325;
G4double fXof_rescaled = fXof / (fXof_max - fXof_min) - fXof_min/(fXof_max - fXof_min);

G4double fA0_min = 1.100516;
G4double fA0_max = 1.849792;
G4double fA0_rescaled = fA0 / (fA0_max - fA0_min) - fA0_min/(fA0_max - fA0_min);

G4double fCN2_min = 0.002064164;
G4double fCN2_max = 0.1199827;
G4double fCN2_rescaled = fCN2 / (fCN2_max - fCN2_min) - fCN2_min/(fCN2_max - fCN2_min);

G4double fP1_min = 10.09451;
G4double fP1_max = 99.95741;
G4double fP1_rescaled = fP1 / (fP1_max - fP1_min) - fP1_min/(fP1_max - fP1_min);
```

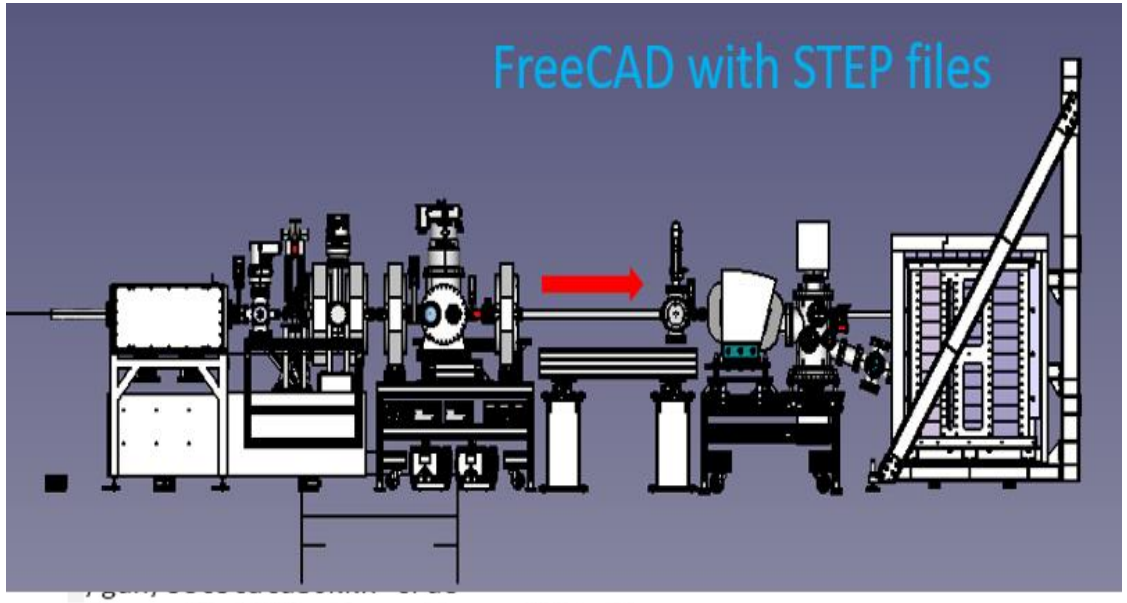

[1] https://apc.u-paris.fr/~franco/g4doxy/html/classG4ParticleGun.html
[2] https://onnx.ai/

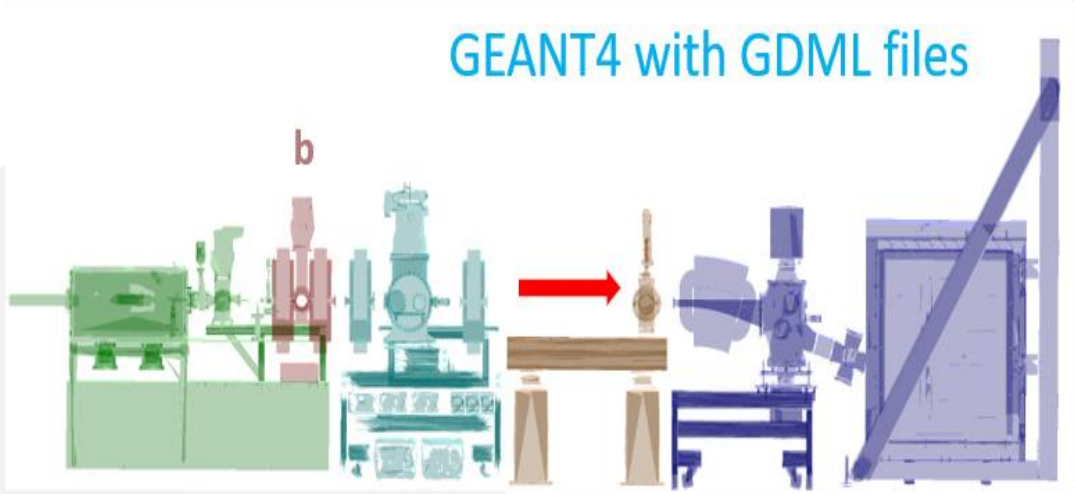
Implementation of Geant4 simulation

- ML model is implemented into **G4ParticleGun** [1] using **ONNX**[2] interface.
- Multithreading on CPU is possible.
- ML model input parameters are changeable with Macro file.

- FreeCAD software is used to convert STEP files into GDML files
- GDML files are used to initialise geometry
- QPs gradients and position are changeable with Macro file

This approach allows easy exploration of varied scenarios



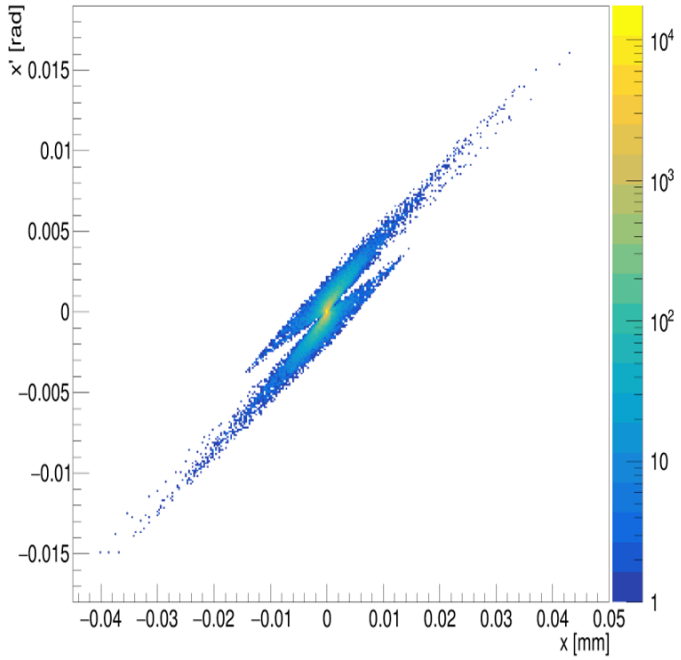


[1] https://apc.u-paris.fr/~franco/g4doxy/html/classG4ParticleGun.html
[2] https://onnx.ai/

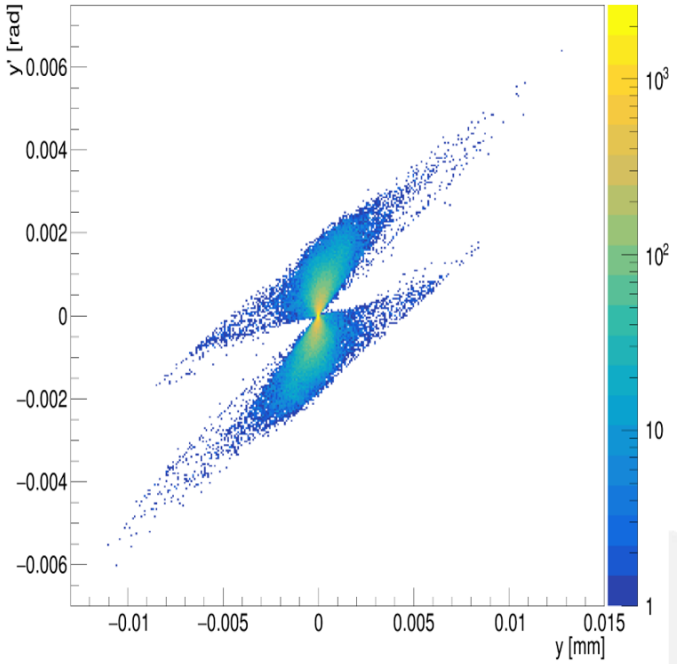
Geant4 simulation results

There is an example of the Geant4 simulation results.

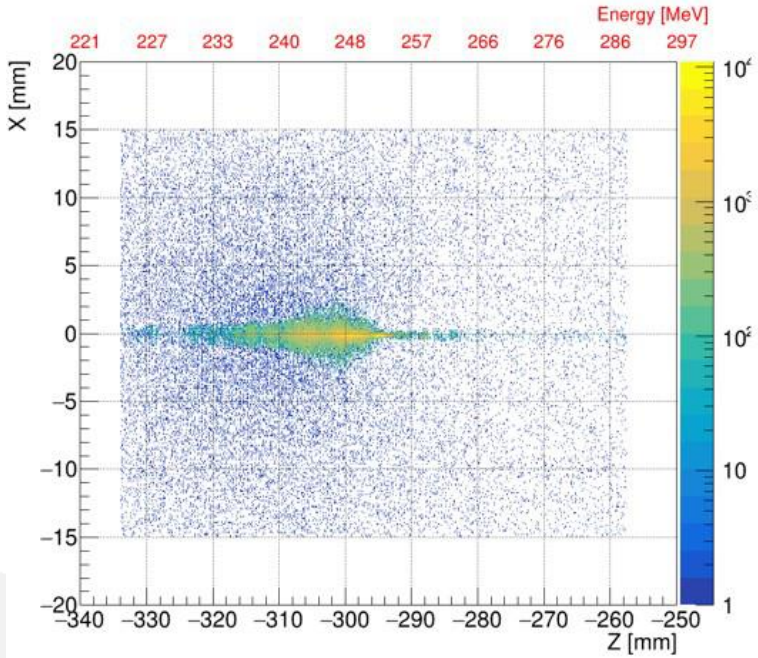
Horizontal phase space



Vertical phase space



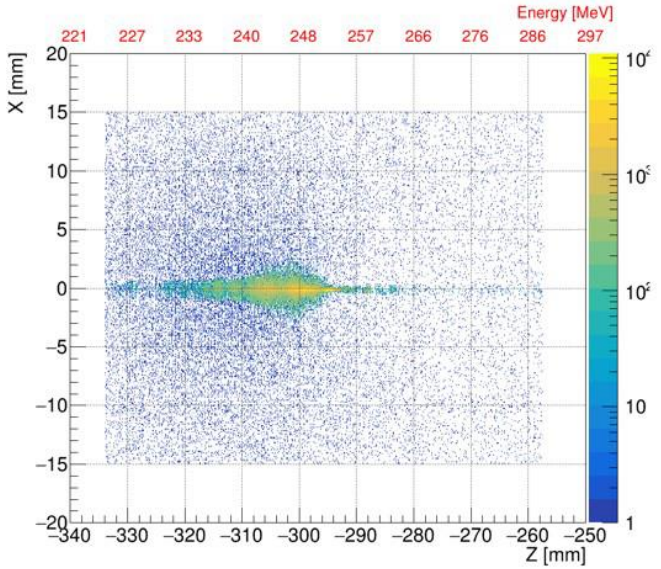
Beam distribution on the spectrometer



Geant4 simulation optimisation

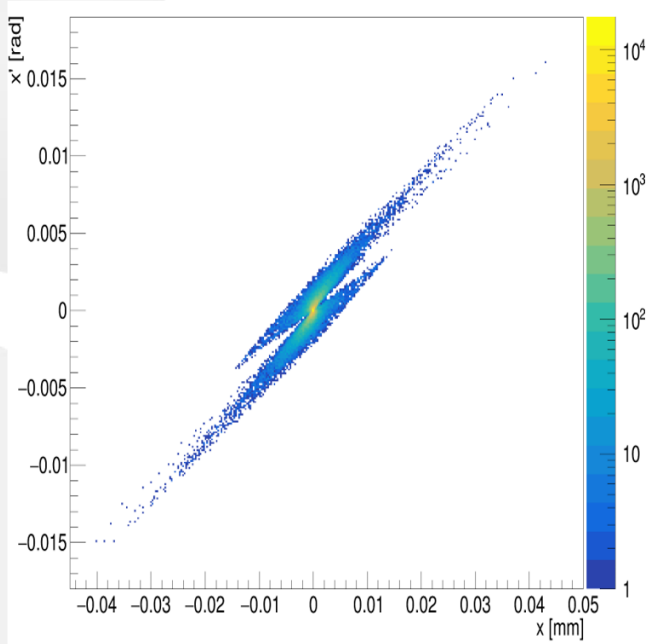
In the Geant4 optimisation case we have 2 consecutive optimisation:

- Beam optimisation coming from the direct model “digital twin”
- Beam propagation in the beamline optimisation similar to RF-track optimisation

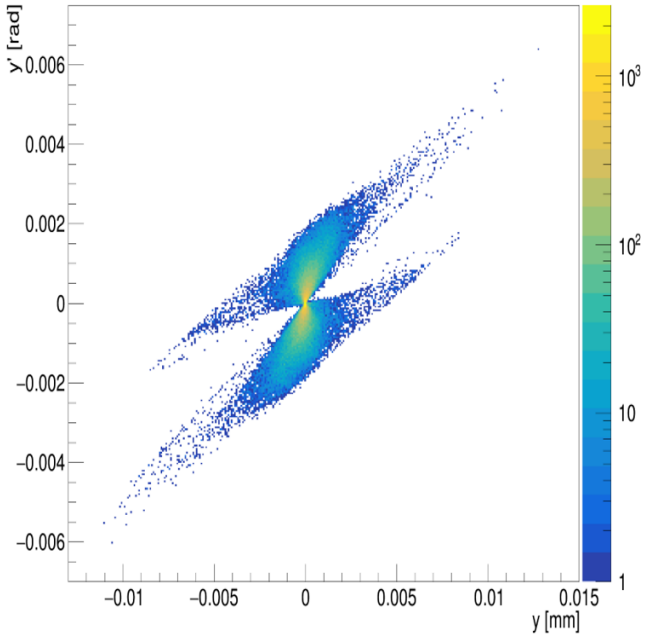


Beam distribution on the spectrometer

Horizontal phase space



Vertical phase space

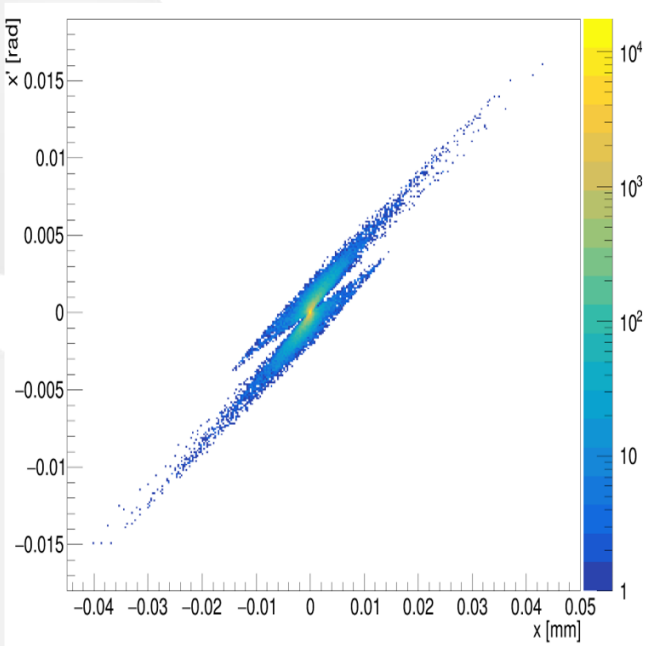


Geant4 simulation optimisation

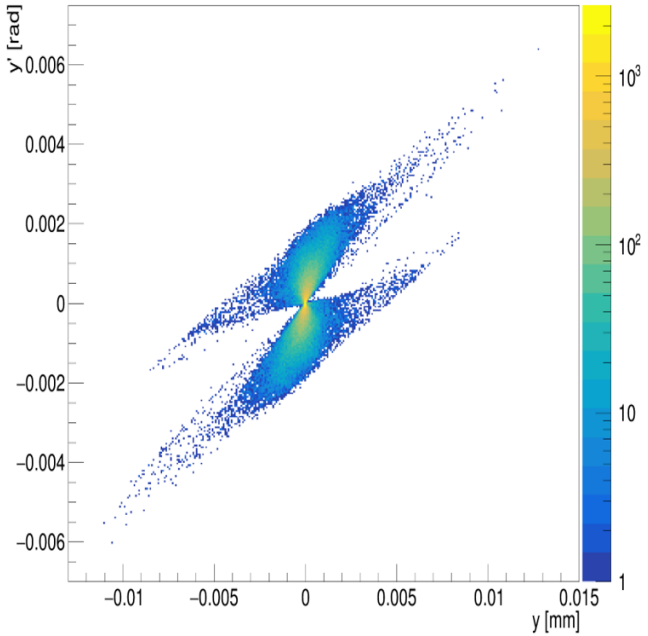
In the Geant4 optimisation case we have 2 consecutive optimisation:

- Beam optimisation coming from the direct model “digital twin”
- Beam propagation in the beamline optimisation similar to RF-track optimisation

Horizontal phase space



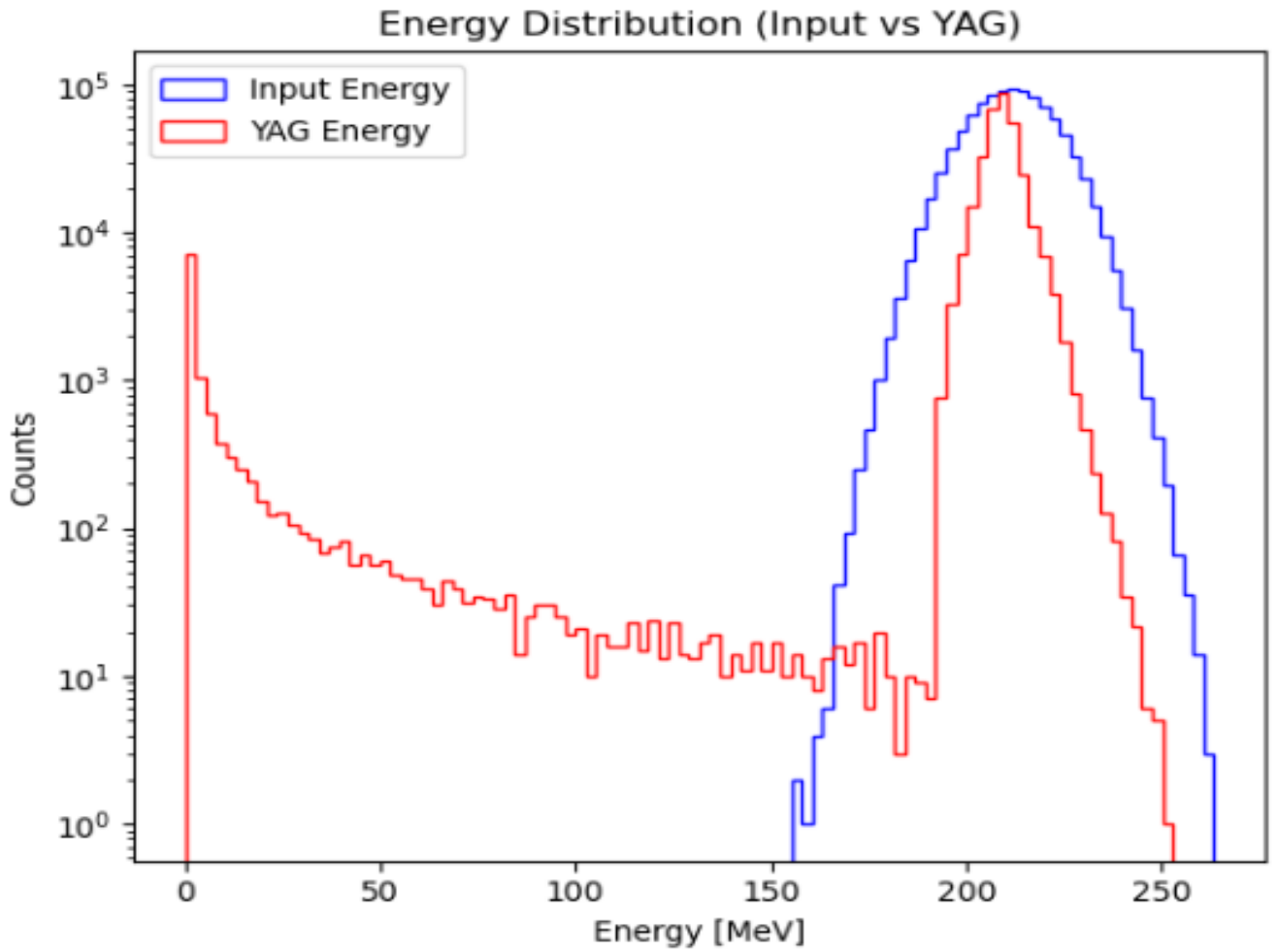
Vertical phase space



Geant4 simulation optimisation

In the Geant4 optimisation case we have 2 consecutive optimisation:

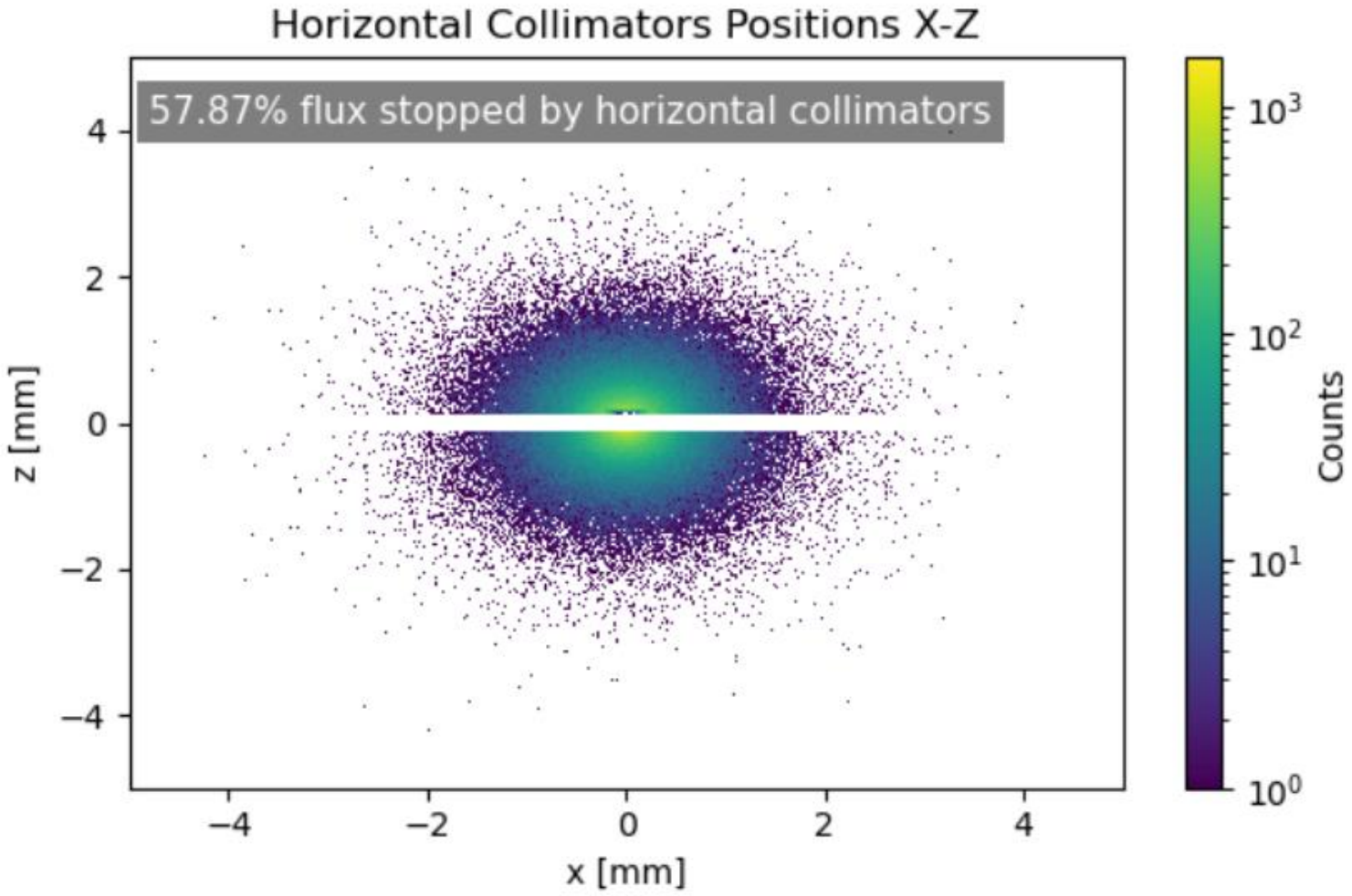
- Beam optimisation coming from the direct model “digital twin”
- Beam propagation in the beamline optimisation similar to RF-track optimisation



Geant4 simulation optimisation

In the Geant4 optimisation case we have 2 consecutive optimisation:

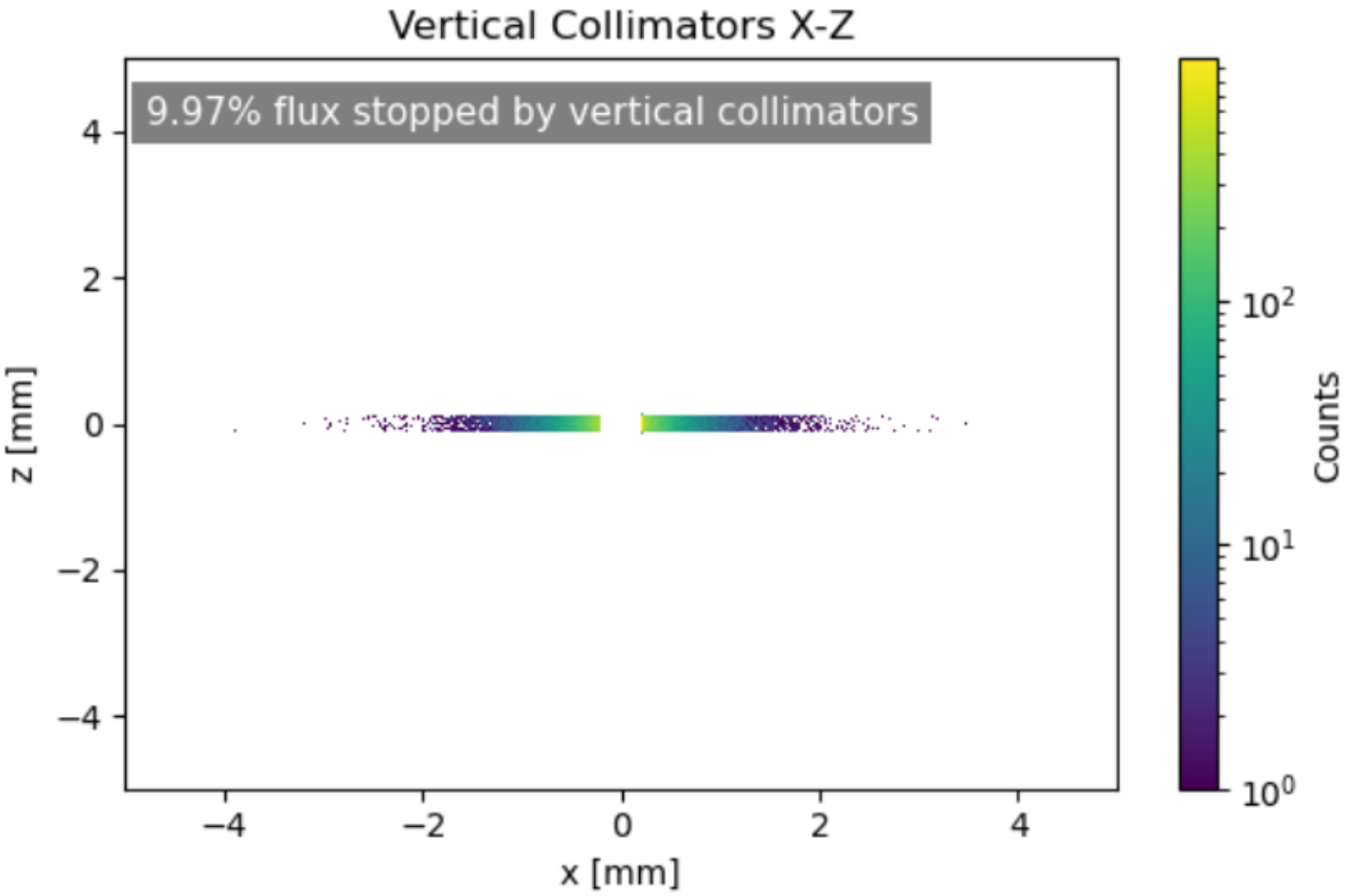
- Beam optimisation coming from the direct model “digital twin”
- Beam propagation in the beamline optimisation similar to RF-track optimisation



Geant4 simulation optimisation

In the Geant4 optimisation case we have 2 consecutive optimisation:

- Beam optimisation coming from the direct model “digital twin”
- Beam propagation in the beamline optimisation similar to RF-track optimisation



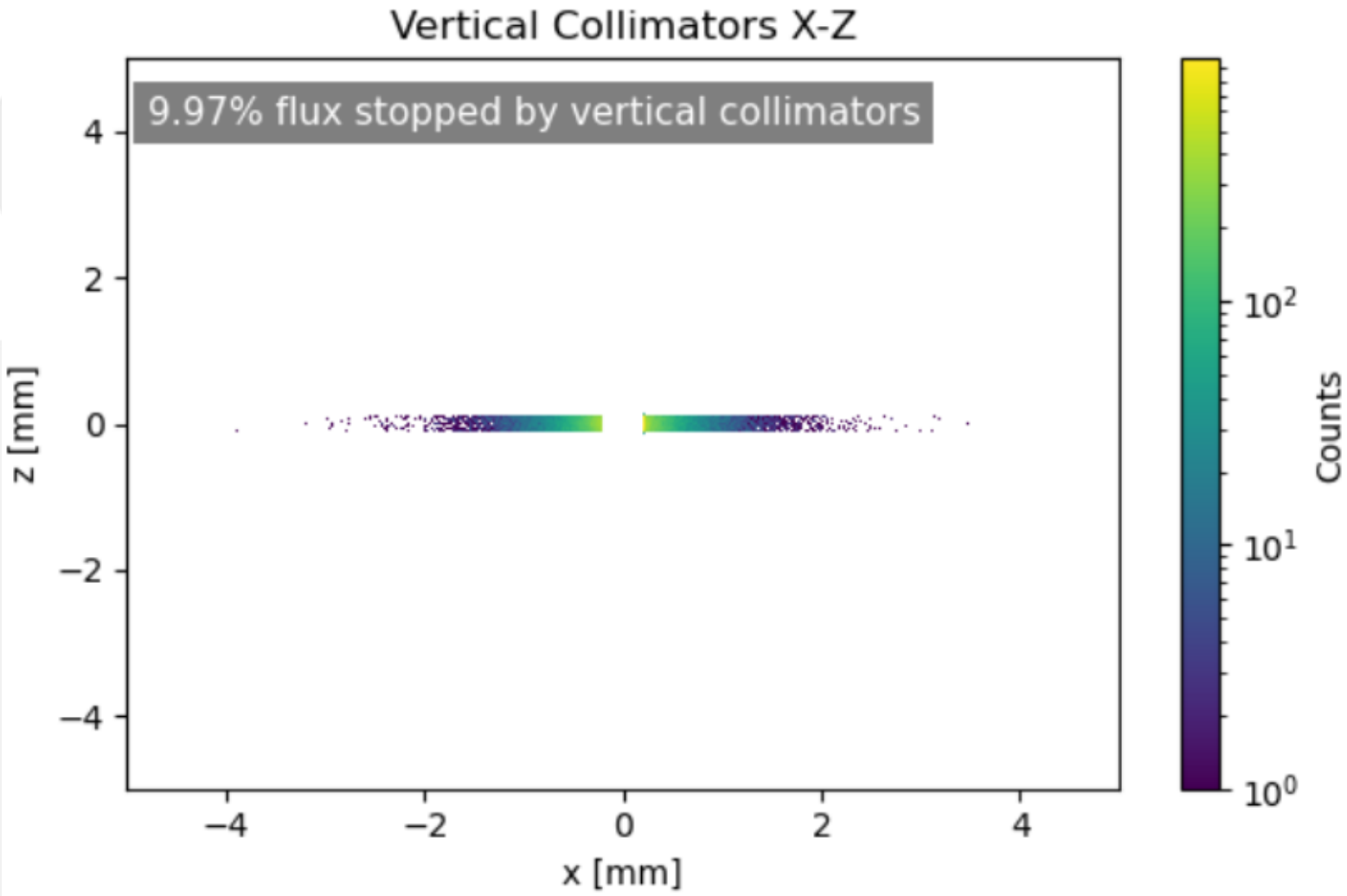
Geant4 simulation optimisation

In the Geant4 optimisation case we have 2 consecutive optimisation:

- Beam optimisation coming from the direct model “digital twin”
- Beam propagation in the beamline optimisation similar to RF-track optimisation

With Geant4 simulation we can optimise effect of the collimator on the electron beam which is impossible in the RF-Track

At the same time simulations are longer then in the RF-Track



Optimisation approaches comparison

	RF-Track	Geant4
Simulation Time	1 second	Up to 15 seconds if no MT
Optimisation Time	15 minutes	Up to an hour
Key Physics	Collective effects	Particle-Matter Interactions, Secondary Particles
Geometry Handling	Internally defined lattice	Import via GDML
Key Advantage	Speed: Ideal for rapid, large-scale parameter scans.	Fidelity: Ideal for detailed verification and detector simulation.

To conclude

At Pallas we are working on optimisation of

- Laser-plasma injector
- Beam transportation and diagnostics

This is done using ML methods as

- ML modelling
- Bayesian optimisation

These approaches give us optimised configuration maximum in 1 hour without any human-code interaction.

To conclude

At Pallas we are working on optimisation of

- Laser-plasma injector
- Beam transportation and diagnostics

This is done using ML methods as

- ML modelling
- Bayesian optimisation

These approaches give us optimised configuration maximum in 1 hour without any human-code interaction.

To notice: **Geant4** start-to-end simulation will be available as "*advanced example*"

Xopt Script Overview – Defining the problem

SLAC

Define the domain/goals

$x_1, x_2 \in [0, \pi]$

$\mathbf{x}^* = \arg\min f(\boldsymbol{x})$

$g(\boldsymbol{x}) \leq 0$

In [2]:

```
from xopt import VOCS
import math

vocs = VOCS(
    variables = {
        "x1": [0, math.pi],
        "x2": [0, math.pi]
    },
    objectives = {"f": "MINIMIZE"},
    constraints = {"g": ["LESS_THAN", 0]}
)
```

Define the objectives/constraints

$f(x_1, x_2) = x_1^2 + x_2^2$

$g(x_1, x_2) = 1 - x_1^2 - x_2^2$

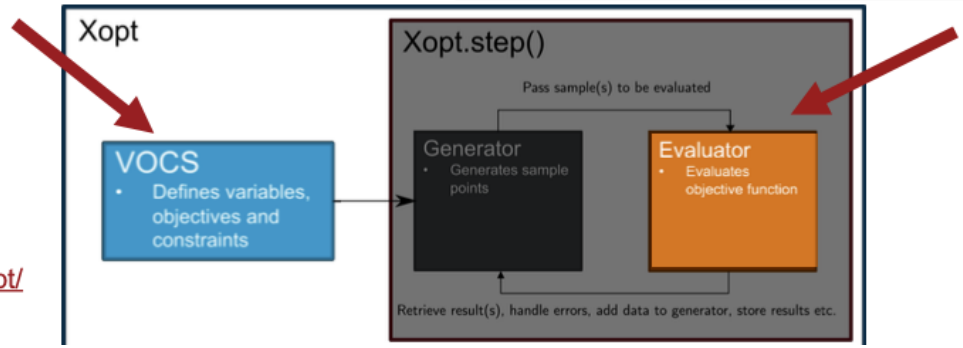
```
evaluate(inputs: dict) -> dict
```

In [1]:

```
from xopt import Evaluator

def evaluate_function(inputs: dict) -> dict:
    objective_value = inputs["x1"]**2 + inputs["x2"]**2
    constraint_value = -inputs["x1"]**2 - inputs["x2"]**2 + 1
    return {"f": objective_value, "g": constraint_value}

evaluator = Evaluator(function=evaluate_function)
```



https://christophermayes.github.io/Xopt/examples/basic/xopt_basic/

7