Neutron and gamma beam simulation using OpenMC and Python's libraries for Machine Learning

M.Eng. Norberto Sebastián Schmidt

norberto.schmidt@cab.cnea.gov.ar

Neutron Physics Department - Bariloche Atomic Center
Comisión Nacional de Energía Atómica (CNEA) - Argentina

3rd Workshop of Spanish Users on Nuclear Data on
"Machine Learning in Nuclear Science and Technology Applications"

May 27, 2021

Comisión Nacional de Energía Atómica

Instituto Balseiro

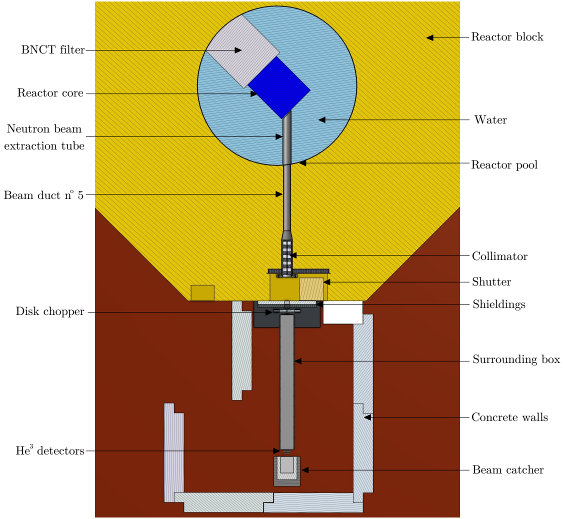
Acknowledgments

- To my Master of Nuclear Engineering thesis directors: Ph.D. José Ignacio Márquez Damián (ESS) and Ph.D. Javier Dawidowski (CNEA).
- To M.Eng. Ariel Márquez (CNEA), Ph.D. José Robledo (CONICET) and B.Eng. Mauricio Debárbora (INVAP), for their helping knowledge.
- To B.Eng. Zoe Prieto (IB) and B.Eng. Inti Abbate (IB), for continuing the development of this method.

Summary

Motivation

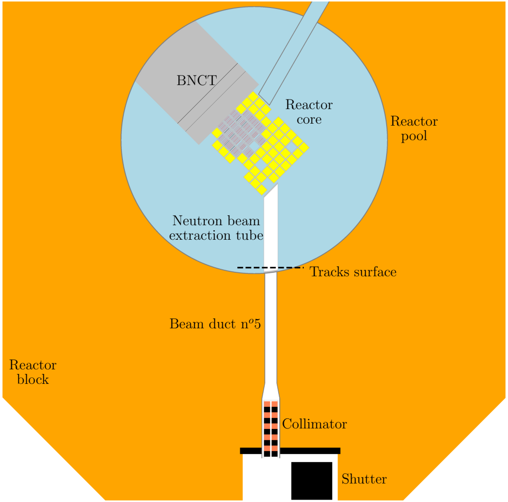
RA-6 Research Reactor - Time-of-flight spectrometry experimental facility



- Design of a time-of-flight spectrometry facility in the RA-6 Research Reactor.
- Necessity to estimate the neutron current at the exit of the duct.

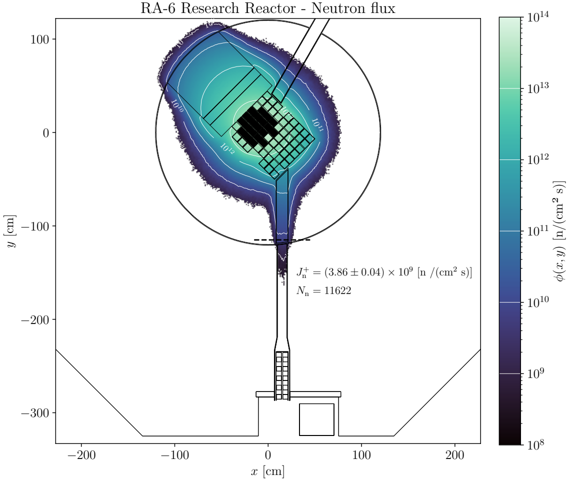
OpenMC transport code description

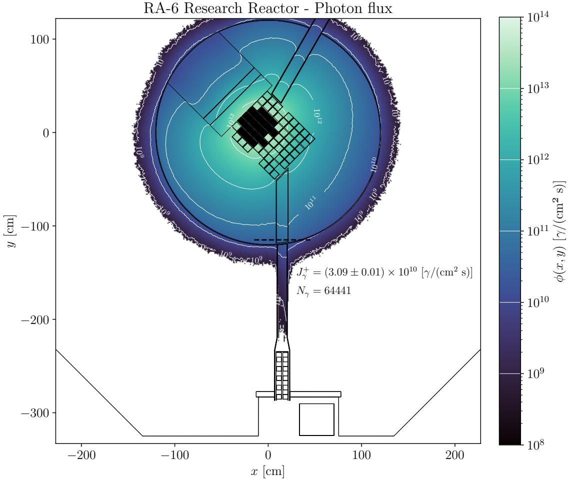
RA-6 Research Reactor - OpenMC model



- Open-source (`https://openmc.org/`).
- Modeling the RA-6 Research Reactor from IEU-COMP-THERM-014 NEA Benchmark.
- Modification to write the particles that cross a given surface.
- Possibility to transform track files to `.mcpl` format.
- Use the same track files to simulate the particles in different codes (`McStas`, `PHITS`, etc.).

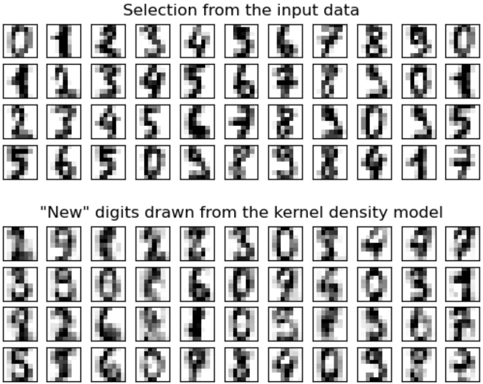
RA-6 Research Reactor radiation simulation





Source particles: 1×10^9 n, time elapsed: $3.6 \times 10^5\,\mathrm{s} = 4.2\,\mathrm{d}$, CPU: i7 8700 with 12 threads.

How to generate more particles?



Example of `scikit-learn` kernel density estimation.

`https://scikit-learn.org/stable/auto_examples/neighbors/plot_digits_kde_sampling.htm`

How to generate more particles?

A proposed alternative to phase-space recycling using the adaptive kernel density estimator method

Neelam Tyagi
The University of Michigan Department of Nuclear Engineering and Radiological Sciences, Ann Arbor, Michigan 48109-2104, and The University of Michigan, Department of Radiation Oncology, Ann Arbor, Michigan 48109-0010

William R. Martin
The University of Michigan Department of Nuclear Engineering and Radiological Sciences, Ann Arbor, Michigan 48109-2104

J. Du
Power Information Network, An Affiliate of J.D. Power and Associates, Troy, Michigan 48098

A. F. Bielajew
The University of Michigan Department of Nuclear Engineering and Radiological Sciences, Ann Arbor, Michigan 48109-2104

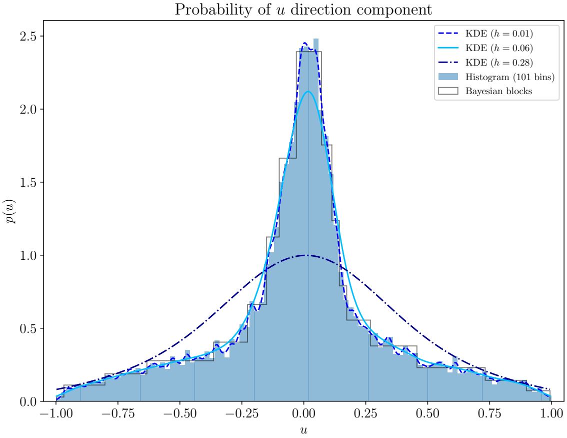
Indrin J. Chetty[a)]
The University of Michigan, Department of Radiation Oncology, Ann Arbor, Michigan 48109-0010

(Received 13 June 2005; accepted for publication 8 December 2005; published 31 January 2006)

We have implemented a nonparametric density estimation technique, the adaptive kernel density estimator (AKDE), to generate additional phase space (PS) variables in the vicinity of simulated PS points in Monte Carlo linear accelerator simulation. The method involves the placement of kernels at simulated PS points that have a "window width" that depends on the density of simulated PS points. This method has been tested on known one-dimensional (1-D) and two-dimensional (2-D) probability density functions (PDFs) and has been used to sample (photons only) from PS files generated from accelerator simulations. The original simulated PS vector (x, y, u, v, E) was reduced to a rotationally invariant PS vector (r, θ, α, E) that takes advantage of the azimuthal symmetry (ϕ) above the collimating jaws. The new PS vector $(r', \theta', \alpha', E')$ is sampled in the vicinity of the sampled PS vector (r, θ, α, E). The first step in assessing the accuracy of the method was a correlation analysis among the AKDE generated PS variables compared with correlations among the original PS variables. "In-air" particle fluence distributions between AKDE samples and the original PS distribution showed agreement within 2% (−8.8% to 6.8%) across the entire phase space plane. Central axis energy distributions and angular distributions agreed on average to within 1.5% (range=−1.5% to 6.6%) and 0.1% (range=0 to 3.0%), respectively. Dose profiles were calculated for field sizes 3×3 cm^2, 10×10 cm^2, and 30×30 cm^2 for AKDE and compared against calculations performed with PS recycling. AKDE calculated depth doses and profiles were within 2% and 2%/1 mm, respectively, of those computed using PS recycling. © *2006 American Association of Physicists in Medicine.* [DOI: 10.1118/1.2163250]

Key words: adaptive kernel density estimator, Monte Carlo, phase space, linac simulation

Univariate kernel density estimation



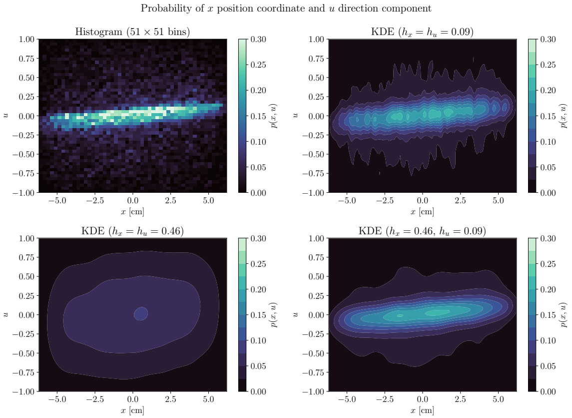
$$\hat{f}(x,h) = \frac{1}{N \cdot h} \sum_{i=1}^{N} K\left(\frac{x - X_i}{h}\right)$$

Gaussian univariate kernel function:

$$K(z) = \frac{e^{-(z^2/2)}}{\sqrt{2\pi}}$$

$$\Rightarrow h = \sigma$$

Multivariate kernel density estimation



$$\hat{f}(\mathbf{x}, \mathbf{H}) = \frac{1}{N \cdot \det(\mathbf{H})} \sum_{i=1}^{N} K\left(\mathbf{H}^{-1}(\mathbf{x} - \mathbf{X}_i)\right)$$

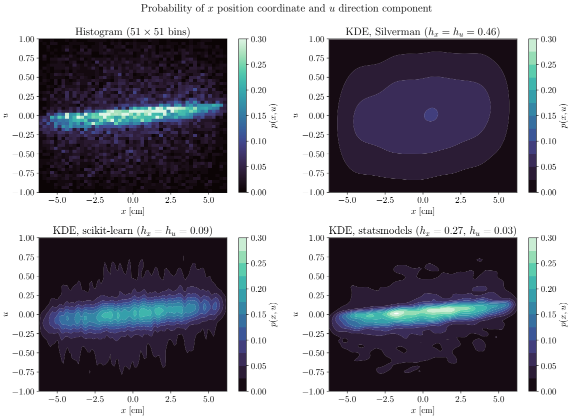
Gaussian multivariate kernel function:

$$K(\mathbf{z}) = \frac{e^{-(\mathbf{z}^T \mathbf{z}/2)}}{(2\pi)^{(\dim(\mathbf{z})/2)}}$$

If $\mathbf{H}$ is diagonal:

$$\hat{f}(\mathbf{x}, \mathbf{H}) = \frac{1}{N} \sum_{i=1}^{N} \prod_{j=1}^{\dim(\mathbf{x})} \frac{1}{h_j} K\left(\frac{x_j - X_{j,i}}{h_j}\right)$$

Best bandwidth selection



Python's libraries for KDE:

1. `scipy`: Scott's and Silverman's rules (depends only on $\dim(x)$ and N).
2. `scikit-learn`: grid search cross-validation (same h for each variable).
3. `statsmodels`: operator cross-validation (different h for each variable).

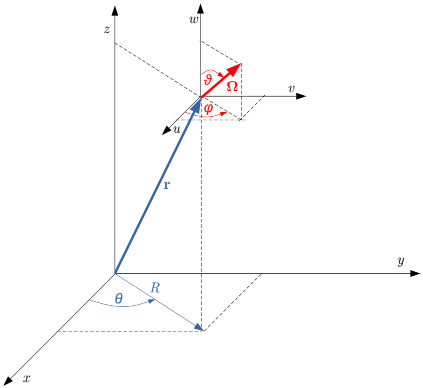
Sampling particles with KDE

1. Run a `OpenMC` eigenvalue core calculation.
2. Write the variables $\mathbf{r}, \boldsymbol{\Omega}, E,$ wgt of the particles that cross certain surface in a file.

$$\mathbf{r} = \vec{r} = (x, y, z = z_0) = (R, \theta, z = z_0)$$

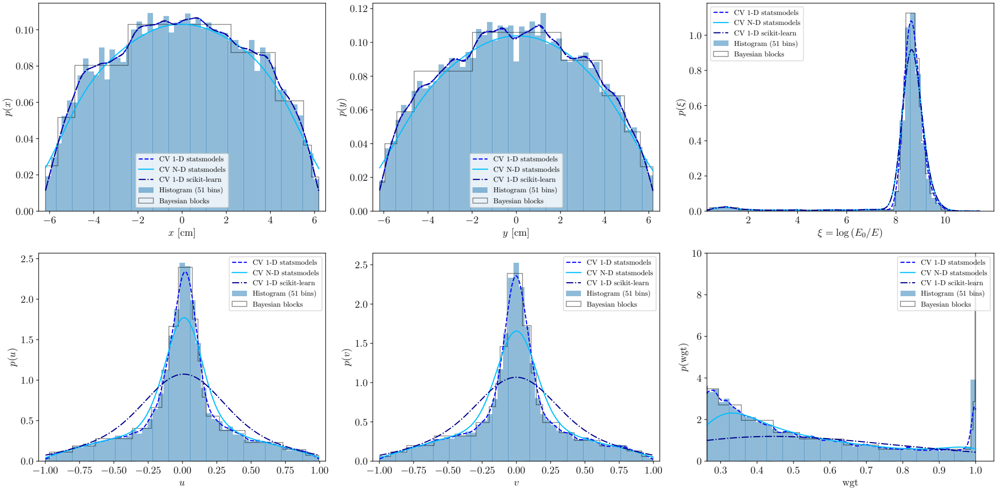
$$\boldsymbol{\Omega} = \hat{\Omega} = (u, v, w) = (\rho = 1, \varphi, \vartheta)$$

3. Compute the multivariate KDE using `statsmodels` CV bandwidth selection for each variable.
4. Write the sampled particles in a `.h5` format file.
5. Run a `OpenMC` fixed source calculation.



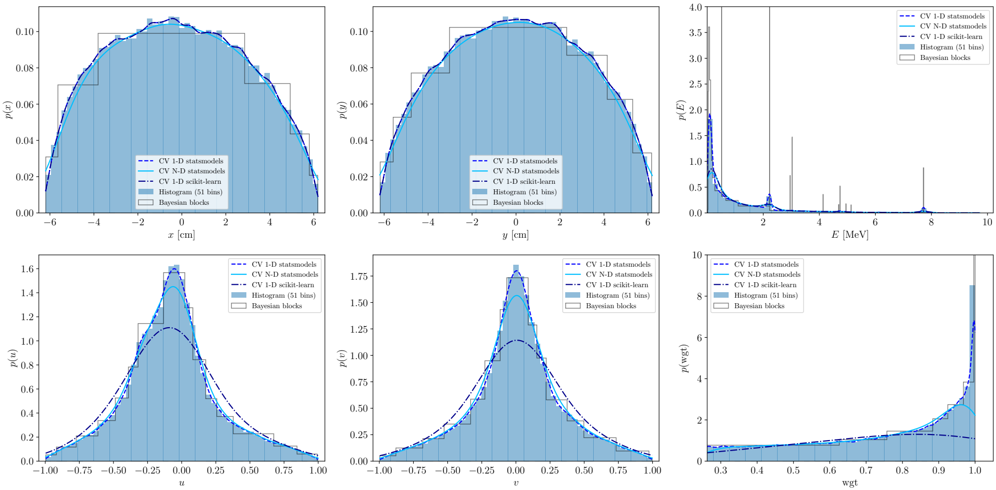
Coordinates ilustration.

Sampling particles with KDE



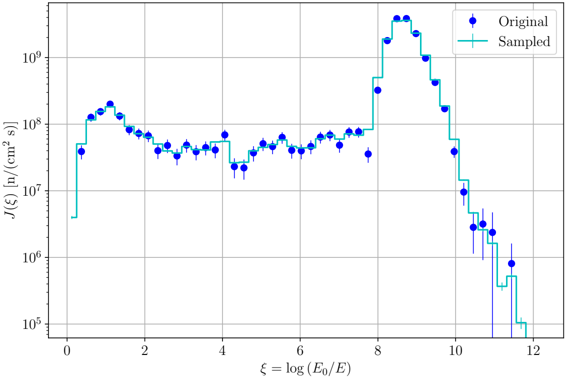
Best bandwidth comparison for neutron variables (in original coordinates).

Sampling particles with KDE

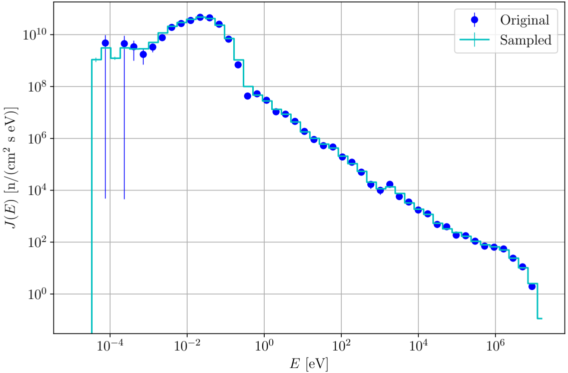


Best bandwidth comparison for photon variables (in original coordinates).

Calculation of current distributions

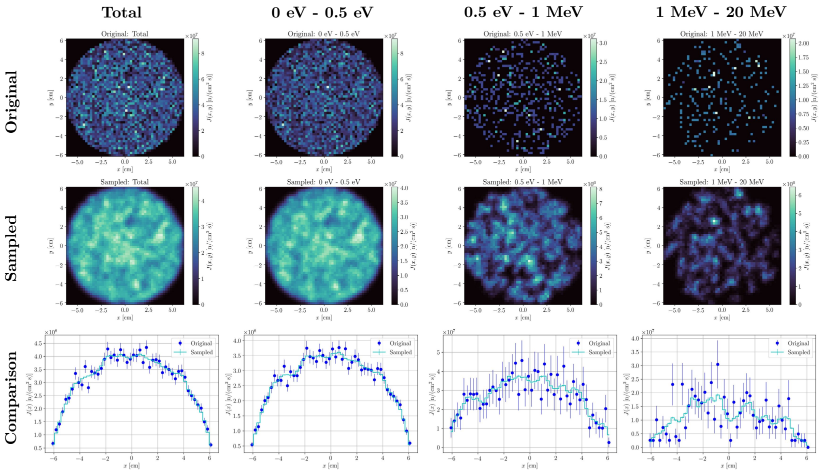


Neutron current lethargy distribution $J(\xi)$.



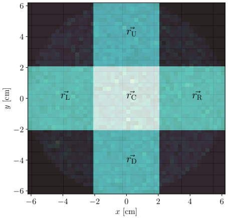
Neutron current energy distribution $J(E)$.

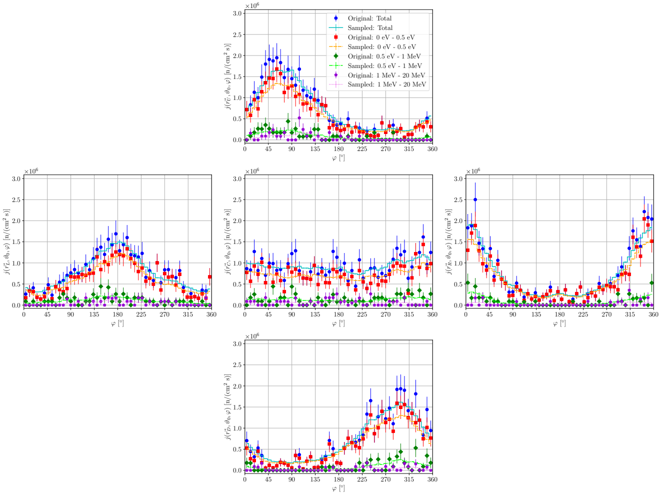
Calculation of current distributions

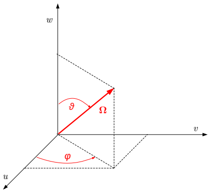


Neutron current spatial distribution $J(x, y)$

Calculation of current distributions

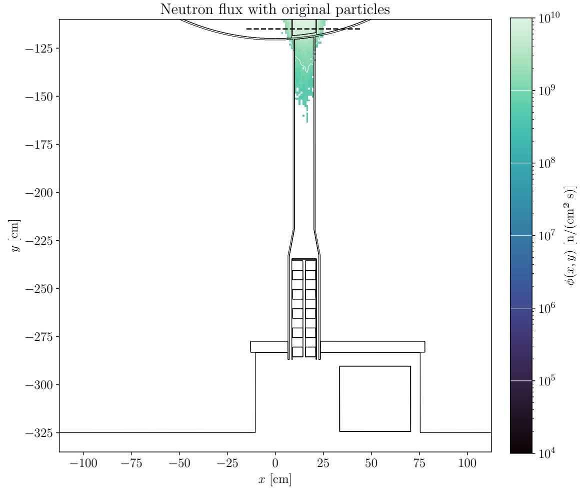


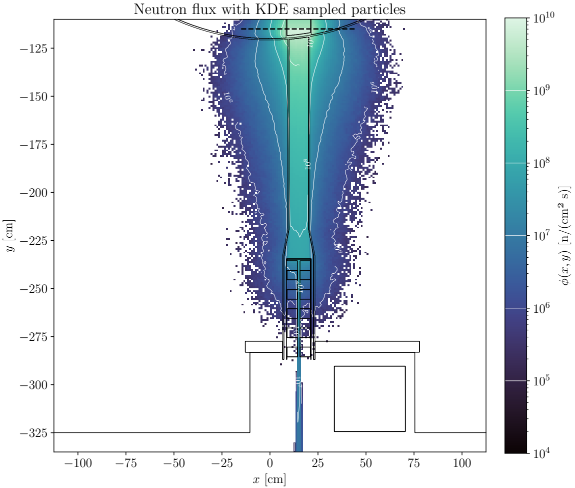




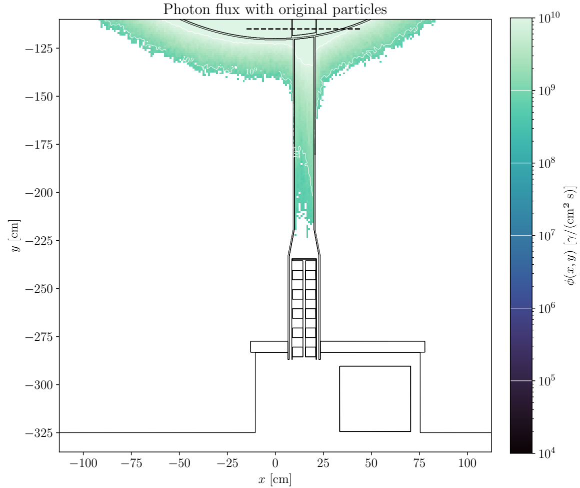
Neutron current angular distribution $j(\varphi)$

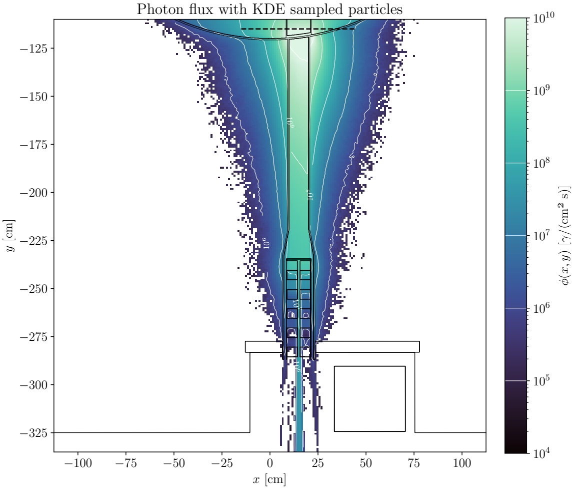
Comparison with the original tracks results





Comparison with the original tracks results





Conclusions

1. All this work was done using open source codes and free software tools.
2. A modification to write track files with `OpenMC` was generated.
3. New particles from these track files can be sampled using kernel density estimation.
4. Python has several libraries that estimate automatically the best bandwidth for the particles' variables.
5. Distributions of sampled particles with `statsmodels` resemble with originals.

Thank you for your time.

Questions?