ISBN: 978-3-95450-208-0 doi:10.18429/JACoW-IPAC2019-WEPTS025

MINISCATTER, A SIMPLE GEANT4 WRAPPER

K. N. Sjobak[1], H. Holmestad, University of Oslo, Oslo, Norway
[1]Also at CERN, Geneva, Switzerland

Abstract

In order to estimate what happens to particle beams when they hit windows, gas, and various other targets, a simple tool has been developed based on Geant4. This tool wraps geometry setup, primary beam generation from Twiss parameters, visualization, and automatic analysis and plots in a simple-to-use command-line tool. Furthermore, a Jupyter-friendly Python interface for running simulations and parallelized parameter scans is included. The code, its interface, and a few selected examples will be presented.

INTRODUCTION

MiniScatter is an easy-to-use tool for running Geant4 [1–3] simulations for relatively simple geometries, consisting of scattering targets, detectors, and magnets lined up behind each other. As it is primarily intended for users at particle accelerator facilities, beam distributions are described using Twiss parameters for generation and analysis. Energy depositions and energy deposition densities, particle types, phase-space distributions, raw hits etc. can be extracted from the ROOT [4] output, which makes it easy to postprocess.

The program itself is controlled via command-line arguments, through which geometry-, target-, field-, beam-, and analysis parameters can be set. Additionally, a Python wrapper is included, which works well with Jupyter [5]. This provides a flexible interface to run the program and interact with the output, as well as scanning parameters. Such scans can also be done in parallel, utilizing all local CPUs to quickly collect simulation data on the influence of e.g. initial beam distribution, material properties, magnetic field strengths etc.

The initial motivation for writing MiniScatter was the need to estimate the effect of a thin foil on a particle beam in the GRACE beamline at AEgIS [6], and later in front of the plasma lens [7, 8] at CLEAR [9, 10]. In the GRACE beamline thin foils were used to reduce the energy of the antiproton beam in order to do detector tests, while for the plasma lens a thin polymer window is used to exclude the gas spillage around the plasma lens from the upstream accelerator vacuum. In both cases it was essential to know the effect of these foils on the downstream particle beam distribution. Another use-case is the design of a plasma lens positron source, which requires the selection of target thicknesses and evaluation of the transport of mixed-species beams through magnetic fields and narrow apertures. Finally, it has also been used to demonstrate the dose deposition distribution when stopping the beam in targets, especially water, visualizing the impact of different shielding geometries, etc..

In all these cases, having this tool available meant that quick evaluations of the effect of materials could be done, as well as the effect of the target- and beam parameters. This tool, has been released on GitHub [11] under the GPLv3 license [12].

IMPLEMENTATION

The code of MiniScatter is organized like a typical Geant4 application, with a `DetectorConstruction`, `PhysicsList`, `PrimaryGeneratorAction`, and more. Each of these classes are responsible for different parts of the simulation, such as setting up the geometry, defining the physics models used in the simulation, generating the initial particle distribution, and collecting data and writing it to file. The top-level `main` routine is responsible for collecting the user input (command-line arguments), setting up the Geant4 framework, reading macro files, and launching the GUI. If a number of events is requested, it will start the run[1].

Geometry and "Magnet Classes"

The geometry is defined by the `DetectorConstruction` class, where it is built up of a target volume, a tracking "detector", and one or more "magnets".

The target volume is made of a user-selectable material and thickness. It is always positioned at $z = 0$, unless the thickness is set to zero, in which case the target volume is not created. Any material known to Geant4 (NIST materials), or gasses of variable pressure, can be used. The target volume is sensitive, collecting data on energy deposits and particle exit positions.

The tracking detector is a very thin volume where the material is set to be the same as in the general all-containing mother volume so that it does not influence the particles, but simply register their position, momentum, and type as they enter the volume. The detector is placed an adjustable distance behind the target.

Finally, one or more "magnets" may be used. These are called "magnets" because they *can* contain a magnetic field, however in general they are simply elements with a more complex geometry that can be placed anywhere along the z-axis, as long as they are not overlapping with each other or with the target[2]. The position and properties of these elements are specified using command line switches.

[1] Terminology: *Run* – A collection of one or more events. There is typically one run per launch of `MiniScatter`. *Event* – The simulation of one initial particle, from generation until all secondaries have been absorbed or have exited the simulation domain. *Step* – Each particle is transported through the geometry in steps, each step being affected by physics processes selected through Monte Carlo samples based on the current material, particle and beam properties, as well as transport processes and deflection in electromagnetic fields.

[2] As the naming of "magnets" are confusing given that they are currently just general elements, the name may change in the future. Also in the future, the target may be redefined as just another general element.

Each of the "magnet" elements include a sensitive detector of the same type as the target volume; this sensitive detector is defined on an enveloping volume in a parallel world. This ensures that even if the element is comprised of many nested sub-volumes, all steps inside the envelope gets registered.

Since each magnet type is defined in a separate class, the simulation is easily extendable to new element types. Currently the supported "magnets" are plasma lenses and circular-aperture collimators, with plans to extend to more types of elements.

The magnetic fields of the "magnet" elements are defined in separate classes from the geometry. These classes describe the field at any point inside the magnets, and handle the transformation from the local to the global coordinate system used for particle tracking.

Physics Lists

MiniScatter is limited to the Geant4 reference physics lists, including options like single scattering (`__SS`) which can be important for very thin targets. The user is responsible for selecting a suitable physics list for their needs.

Initial Particle Distribution

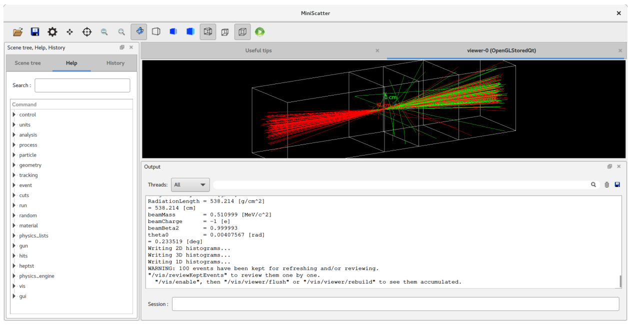
The initial particle distribution can either be a parallel point-source "pencil beam", a parallel beam "plane source", or described by the Twiss parameters α, β, ϵ_N. These parameters describe the variances in the initial positions in phase space, such that the variance in position is $\sigma_x^2 = \beta_x \epsilon_g$, in angle $x' = \Delta x/\Delta z$ is $\sigma_{x'}^2 = \epsilon_g(1+\alpha_x^2)/\beta_y$, and the covariance $\sigma_{x,x'} = -\alpha\epsilon_g$, where $\epsilon_g = \epsilon_N/(\gamma_{\text{rel}}\beta_{\text{rel}})$ is the geometrical emittance and γ_{rel} and β_{rel} are the relativistic γ- and β-factors. A separate set of Twiss parameters can be defined and used for the vertical ($\{y, y'\}$) plane. The initial energy is the same for all particles.

The beam distribution can be generated around any point x, z. Optionally, the beam may be created at $z = 0$ (in the middle of the target), and then the particle positions extrapolated backwards along their initial velocity vector to a user-selectable z-position. This can be used to bring the starting point of a beam converging on a point inside the target to the outside of the target.

Output Files

The main output of a simulation is given as a ROOT [4] file which can be opened directly, without needing any dictionaries. The lack of dictionaries makes the files fully self-describing and easy to handle from other programs, especially through PyROOT. All output files are written from the class `RootFileWriter`, which at the end of each event fills the histograms with the data collected from the sensitive detectors, and at the end of a run computes final statistics.

The output file contains a long list of histograms with statistical data, along with some arrays containing Twiss parameters, particle type counts, and metadata. Histograms are provided for the initial particle distribution, the target, the detector, and the "magnet" elements. These histograms contain information about total energy deposition per event, energy deposition density per run, and phase-space distribution per particle species. Furthermore, the file includes the number of particles of each type, Twiss parameters, and meta-data.



```
RadiationLength = 538.214 [g/cm^2]
= 538.214 [cm]
beamMass        = 0.510999 [MeV/c^2]
beamCharge      = -1 [e]
beamBeta2       = 0.999993
theta0          = 0.00407567 [rad]
= 0.233519 [deg]
Writing 2D histograms...
Writing 3D histograms...
Writing 1D histograms...
WARNING: 100 events have been kept for refreshing and/or reviewing.
"/vis/reviewKeptEvents" to review them one by one.
  "/vis/enable", then "/vis/viewer/flush" or "/vis/viewer/rebuild" to see them accumulated.
```

Figure 1: Example of the GUI output when running with the example command line options. The geometry (a 30 mm thick water volume centered at $z = 0$ and a thin detector plane at $z = 30$ mm) is shown in the white wireframe, the red tracks converging towards the center are the incoming electrons, and the multicolored cone escaping on the right hand side are the scattered beam.

COMMAND LINE USE

The main user interface to MiniScatter is through command line arguments; Geant4 macros can only be used for standard commands as no special messengers are provided. This interface is self-documenting; all the valid command line arguments and their default settings can be seen by running `./MiniScatter -h`.

As an example, a simple simulation can be ran as:

```
./MiniScatter -n 100 -b e- -t 30 -m G4_WATER
-z *-100.0 -c 10:0.00001:0 -d 50 -g
```

This creates 100 (`-n`) electrons (`-b`) and sends them toward a 30 mm (`-t`) thick target made out of water (`-m`). The beam distribution is converging towards a point in the middle of the target but actually starting 100 mm before the target (`-z *`), so that if the target would not be there it would have (`-c`) $\beta = 0.00001$ m, $\alpha = 0$, and $\epsilon_N = 10$ μm. The detector is placed at $z = 30$ (`-d`) after the center of the target.

When run with `-g`, a Geant4 GUI is launched, in which the geometry and the particle tracks can be visualized (`-g`) as shown in Fig. 1. While very useful for debugging, for production it is usually more useful to run in command line mode as it is faster and easier to call from scripts. Here, it is useful to set the output folder (`-o`) and filename (`-f`).

"Magnet" elements are added using the `--magnet` keyword, followed by the element position, type, length, gradient, and type-specific key-value pairs. Any number of magnets can be added to the simulation.

PYTHON AND JUPYTER INTEGRATION

When running more than a few simulations, it is often more practical to use the Python3 wrapper through the Jupyter interface. This wrapper has three main parts: (1)

ISBN: 978-3-95450-208-0 doi:10.18429/JACoW-IPAC2019-WEPTS025

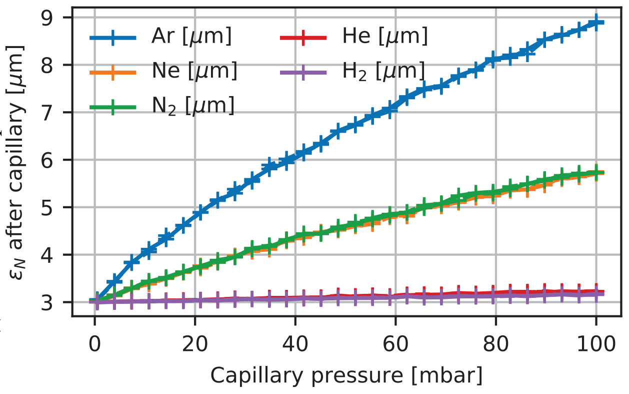


Figure 2: Beam normalized emittance as a function of gas pressure for Argon, Neon, Nitrogen, Helium, and Hydrogen.

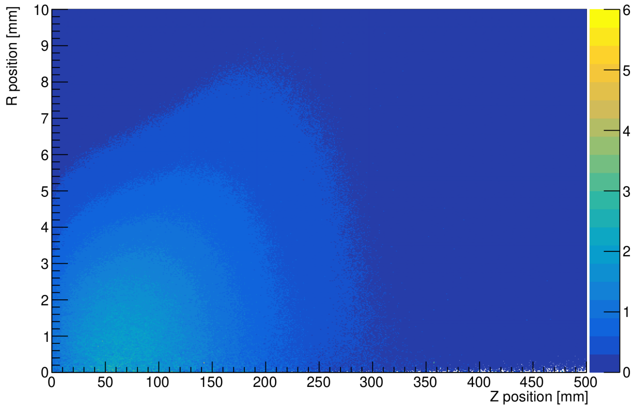


Figure 3: Dose deposition in water as a function of r and z, for a 215 MeV parallel electron beam with $\sigma_x = \sigma_y = 3$ mm.

A module to setup the command line and run MiniScatter, and to load results from simulations; (2) A module to scan parameters through a range, leveraging the first module; (3) A module to produce standard plots such as dose maps from the output. The first two modules expects a Python dictionary where the keys maps to the available command line arguments. This is leveraged in the scanner, where the user provides the name of one variable to be scanned along with the points it should be scanned through and a "base simulation setup" which provides all the other parameters. Since most modern systems have multiple CPU cores available, the scanner is built to leverage them. It does this by keeping the simulations to be ran in a queue, and then use as many cores as allowed until the queue is exhausted, running one copy of MiniScatter per core. At the end of a scan or a single simulation the output is automatically stored to ROOT- and HDF5 [13] files, so that if the user re-runs the same simulation (for example after restarting Jupyter) the output can be quickly and seamlessly reloaded, not needing to re-run the simulation.

Several example notebooks for running MiniScatter and analyzing the output are included in the repository. The notebooks may be run locally, or if wanting a quick start, on CERN SWAN [14]. SWAN makes it possible to get started with simulations in a few minutes, without needing to install anything on the local computer.

EXAMPLE SIMULATIONS

Two simulations are shown, each corresponding to one of the examples distributed as Jupyter notebooks along with the code. Note that these are meant as examples for what is possible using MiniScatter, not as complete physics studies.

Emittance Growth Due to Gas Scattering

One application of MiniScatter has been to estimate the emittance growth due to passing a beam through a gas-filled capillary. For this, the target is assumed to be 15 mm long and filled with gasses of various species, and the gas pressure is scanned at a fixed temperature of 300 K, determining the density. The initial beam has $\beta_x = \beta_y = 1.5$ m and $\alpha_x = \alpha_y = 0$ at the center of the target, and an initial $\epsilon_N = 3$ μm and $E_0 = 215$ MeV. The resulting final emittances from the charged particles with $E > 0.95E_0$ on the detector plane 15 mm after the center of the target, are shown in Fig. 2. Note that this simulation has been ran with a single-scattering physics model, which can sometimes be important for thin or non-dense targets.

Dose Distribution from an Electron Beam in Water

Another application of MiniScatter is to estimate the radiation dose deposited in targets as a function of beam charge, beam energy, Twiss parameters, and depth in the target. For this reason, one of the produced ROOT histograms contains the total amount of energy deposited per bin in the simulation. A Python postprocessing tool is provided to normalize this to Grays for a given amount of primary particles. An example of this output is shown in Fig. 3.

Note that for this type of simulation, when possible it is recommended to use the Geant4 NIST materials (e.g. G4_WATER) when the particle beam consists of ions (including protons) and to define custom materials, as this will trigger the loading of more accurate physics models which affect e.g. the position of the Bragg peak [15, 16].

OUTLOOK AND CONCLUSIONS

MiniScatter is a user-friendly tool for quickly simulating particle transport and energy deposition through relatively simple geometries, and has been used as such by the authors. The associated Python tools simplify parameter scans and analysis, making it easy to quantify the effect of various parameters.

The tool has therefore been shared with the community in hope that it will be useful, and development is continuing in order to add new features, simplify interfaces etc. Community contributions in the form of new features, documentation, testing and bug reports etc. are very welcome.

REFERENCES

[1] J. Allison *et al.*, "Geant4 developments and applications," *IEEE Transactions on Nuclear Science*, vol. 53, no. 1, pp. 270–278, Feb. 2006.

[2] S. Agostinelli *et al.*, "Geant4—a simulation toolkit," *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment*, vol. 506, no. 3, pp. 250–303, Jul. 2003.

[3] J. Allison *et al.*, "Recent developments in Geant4," *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment*, vol. 835, pp. 186–225, Nov. 2016.

[4] Rene Brun and Fons Rademakers, "ROOT - An Object Oriented Data Analysis Framework", *Nucl. Inst. & Meth. in Phys. Res. A*, vol. 389, pp. 81-86, Apr. 1997. `http://root.cern.ch/`

[5] Thomas Kluyver *et al.*, "Jupyter Notebooks – a publishing format for reproducible computational workflows", in *Proceedings of Positioning and Power in Academic Publishing: Players, Agents and Agendas*, pp. 87–90, 2016.

[6] H. Holmestad, "Data analysis, simulations, and reconstruction of antiproton annihilations in a silicon pixel detector", PhD thesis, University of Oslo, October 2018. `https://www.duo.uio.no/handle/10852/65313`

[7] C. A. Lindstrøm *et al.*, "Overview of the CLEAR plasma lens experiment," *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment*, vol. 909, pp. 379–382, Nov. 2018.

[8] C. A. Lindstrøm *et al.*, "Emittance Preservation in an Aberration-Free Active Plasma Lens," *Phys. Rev. Lett.*, vol. 121, no. 19, p. 194801, Nov. 2018.

[9] D. Gamba *et al.*, "The CLEAR user facility at CERN," *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment*, vol. 909. pp. 480-483, Nov. 2018.

[10] K. Sjobak *et al.*, "Status of the CLEAR electron beam user facility at CERN", presented at IPAC'19, Melbourne, Australia, May 2019, paper MOPTS054, this conference.

[11] Miniscatter, `https://github.com/kyrsjo/Miniscatter`

[12] GNU General Public License version 3, 2007, `https://www.gnu.org/licenses/gpl-3.0.en.html`

[13] HDF5, The HDF group, `https://www.hdfgroup.org`

[14] D. Piparo, E. Tejedor, P. Mato, L. Mascetti, J. Moscicki, and M. Lamanna, "SWAN: A service for interactive analysis in the cloud", in *Future Generation Computer Systems*, vol. 78, pp. 1071–1078, Jan. 2018.

[15] "Water vs. G4_WATER", `http://hypernews.slac.stanford.edu/HyperNews/geant4/get/geometry/1014.html?inline=-1`

[16] T. Toshito *et al.*, "Validation of New Geant4 Electromagnetic Physics Models for Ion Therapy Applications," *Progress in Nuclear Science and Technology*, vol. 2, no. 0, pp. 918–922, Oct. 2011.