Integrating Tracking and Beam-Matter-Interaction for Beam line Design

L. Clemente*,[a], M. Bussmann[a]

[a] *Forschungszentrum Dresden-Rossendorf, Dresden, Germany*

Abstract

Current developments in particle acceleration technologies, specifically interesting for medical applications come along with a massive reduction of space of the source. Although a complete accelerator might fit into a room, the beam line for beam transportation is still a massive system which can for reasons of radiation protection not be scaled easily.

In this publication, we present a modular software tool integrating advanced particle tracing with effects like space charge and sophisticated beam matter interactions. As tracker module we use a common particle tracing algorithm, the general particle tracer (GPT), and a established beam matter interaction simulation toolkit, Geant4, as interaction module. The algorithm is described, the software architecture explained and the code is verified in several exemplary application cases.

Key words: Laser Acceleration, Particle Tracing, Beam line design
PACS: 41.75.Jv, 07.05.Fb, 87.56.bd, 41.85.-p, 41.75.Ht

1. Introduction

With current advances in particle acceleration, especially laser acceleration techniques, a leading thought is to develop small medical devices, requiring only a fraction of space compared to a conventional apparatus. Applications may be x-ray diagnosis or radiation treatment (e.g. with protons). Laser sources are e.g. described in [ref], for possible applications see [ref].

One remaining challenge with small particle sources is the design of a small beam line, as large systems would annihilate the effect of space gain. Designing a compact beam line for medical purposes is a even more delicate process: The patient has to be secured from unwanted radiation, but the overall size must not exceed certain maxima. Along with the decrease of beam line size comes especially the aspect of radiation protection, which has to be taken into account during the design process.

Conventional beam lines are often designed using software packages that fit for one specific aspect. There exist advanced particle tracing algorithms for detailed simulation of particles passing electromagnetic fields and more complex processes, e.g. space charge effects. Another class of codes simulates the interaction of particles with matter based on physicals effect models. Building a compact beam line requires both of this aspects combined together. Secondary particles generated e.g. in pinholes may not be neglected.

We therefore developed a modular simulation tool integrating particle tracing and beam matter interaction and fulfilling these requirements. A common particle tracing algorithm can thus easily be combined with an interaction simulation tool for beam line design.

2. Software design

2.1. Structure

The simulation toolkit flint (FZD Laser Interactor and Tracker) consists of two main modules, a "tracker" and an "interactor". An interface between both at stepping level is established and both modules are used for calculation of a particles trajectory, secondaries, etc. The user has thus access to all elements of both modules. In particular, this means tracking massive amounts of particle through complex geometries and fields,

*Corresponding author
Email address: l.clemente@fzd.de (L. Clemente)

considering effects like space charge on the tracker side or e.g. fission on the interactor side.

2.2. The combination of GPT and GEANT4

There are only few limitations to a tracker or interactor to be fitted into flint. In this paper we use GPT [ref] as tracker and Geant4 [ref] as interactor, but other combinations would be possible with little effort as well. Both tools fulfil all requirements for successful integration.

2.3. Integration into the tracker

The interface between GPT and Geant4 is provided by a GPT custom element, which calls the Geant4 if necessary. This custom element would also be easily integratable into other trackers. Basically every tracker step is performed as usual. After every step, the software check every particle's trajectory for intersection with an interactor solid. The pre and the post step point of the tracker are considered. Afterwards the interactor is used to recalculate the step and the results are reinjected as shown in 2.4. After this process some detailed output is written. Then, the tracker continues normally.

2.4. Combined stepping

In the following, we will describe the exchange of information between the tracker and the interactor for one particle:

1. A step is tracked normally by GPT, using information about all other particles in the simulation.
2. The step is recalculated individually for each particle using Geant4.
   - A particle is generated and started at the pre tracker point, facing with tracker post velocity in direction of the tracker's post step point (thus flying as a secant).
   - The calculation stops, when the stepping time of the tracker is exceeded.
3. Reinjection of the results from the interactor to the tracker.
   - The new position and velocities are taken directly.
   - The rotation of momentum between pre interactor and post interactor is calculated.
   - The post tracker momentum is rotated accordingly.

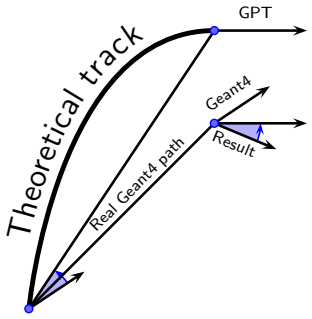

Figure 1: An illustration of the stepping algorithm

An example step to illustrate the stepping algorithm is shown in Figure 1.

Secondaries produced during the interaction calculation are differentiated by mass and charge. Massless or uncharged particles are exclusively tracked by the interactor and not reinjected to the tracker as most trackers can not handle these. All other particles are added normally to the tracker's particle set.

2.5. Input for the calculation

The definition of the calculation world of flint takes place in one single place, a XML file. This file basically has two parts, one for defining tracker options and on for defining the geometry, processes and particles to be used by the interactor. Flint also includes a tool for visualising geometries defined by this XML file. An exemplary visualisation of a geometry can be seen in [ref].

2.6. Output

Flint currently includes several output options. Of course, all output of the tracker and interactor can be used. Flint also provides some additional options like unified trajectories of all particle types (including massless and uncharged particles) or energy deposition in the geometry. This output is written as ASCII, macros for gnuplot [ref] visualisation are provided.

2.7. Parallisation

Parallelising a tracker is often a very difficult process. In the interactor part though, all particles are handled independent from each other, not interfering. This is why flint parallelises the interactor and, additionally the output module. Parallisation is performed using OpenMP [ref] and results in significantly faster simulation.

3. Verification and Examples

4. Outlook

5. Old stuff

5.1. Integration into GPT

The interface between GPT and GEANT4 is provided by two GPT custom elements. The element `G4Virtual` has to be instantiated once in a calculation and is responsible for initializing MPI and the save termination of all GEANT4 processes. The second custom element launches the interaction calculation, a quader shaped `G4Area` can be placed to use GEANT4 calculations in a specific area. This should be used as a bounding box arround GEANT4 solids. **ToDo: check for collisions in GPT?**

After every succesful GPT step, all particles are tested for beeing in a physical volume. If a single particle is inside a `G4Area`, its path will be recalculated. Therefore, the old partcle position before the step, the particle's current position, its velocity's norm in the end of the track, its mass and charge are sent to GEANT4 . In the interactor process, the particle type is recognized by analysis of the sent particle's mass and charge. A similar particle is then started at the pre step point with the post step velocity pointing in direction of the post step point. Thus, the way calculated by GEANT4 is (if no interactions occur) a secant of the actual GPT path.

The track length specified by GPT is tracked in GEANT4 , although processes may occur. After the needed length, the particle's new post step point and its vectorial velocity is sent back to GPT. The post step velocity's and the new post step point are set. The new vectorial velocity is not exclusively used, as the particle's track would then be error. Instead, the change of the velocity vector in GEANT4 is calculated and added to the GPT post step vector. The angel in between the GEANT4 start vector $\vec{p}_{G4old}$ and the GEANT4 end vector $\vec{p}_{G4new}$ can be calcuclated using the scalar product $\varphi = \vec{p}_{G4new} \cdot \vec{p}_{G4old}$. The axis around which the GPT post vector $\vec{p}_{GPTold}$ has to be turned, is given by $\vec{n} = \vec{p}_{G4new} \times \vec{p}_{G4old}$. The new GPT vector $\vec{p}_{GPTnew}$ can then be calculated as

This calculation ensures the minimal error possible.

5.2. Parallelization

As GEANT4 is a monte-carlo toolkit, is can be easily parallelized. In flint, the user can specify a number of GEANT4 processes that should be used for calculation at one time. The single GPT process then provides tasks for all GEANT4 processes and fetches the results from the finished interactors, but has to wait for all interactors to finish before starting a new step. Thus the parallization is only profitable with very large numbers of particles. Another possibility is to launch multiple GPT processes with their corresponding GEANT4 partners. The calculation can thus be run parallel on clusters.

5.3. Configuration

Large simulation and therefore large geometries require a simple, standardized method to configure a simulation and setup all parameters. flint provides the possibility to use XML-Files for that purpose. In a grammar based on GDML **Zitat einfuegen!** the geometry, all fields and some additional parameters, like the particle definitions to be loaded by the interactor, can be edited and adjusted to every calculation. A script provides GPT with a GPT ini file generated from this XML file and GEANT4 reads the file itself. The whole simulation can thus be configured in a single file.

References

[0] Agostinelli et al.
[0] GPT Manual