Optimizing the integration of charged particles on GPU

Nazar Misyats

Supervised by Severin Diederichs and John Apostolakis

September 2025

Abstract

The upcoming HL-LHC runs are projected to require unprecedented demands on computing resources, with detector simulations representing a major part of the workload. To address these challenges, CERN is exploring GPU-accelerated particle transport through the AdePT project, a plugin to Geant4 which offloads the simulation of electrons, positrons and photons to GPU. In this report, we investigate possible performance optimizations for the charged particle integration. First, we experiment with reduced floating-point precision in the numerical integration loop. We show that single precision arithmetic provides significant speedups with negligible loss of accuracy in uniform magnetic fields. Second, we propose a new method for estimating distances to complex detector geometries, in particular for the electromagnetic calorimeter of ATLAS. We show that this estimator returns more accurate distance estimations which translates into an overall speedup of simulations.

Contents

1 Introduction

The projected computing requirements for future HL-LHC runs are expected to rise by a factor of four (Figure 1). In particular, Monte Carlo simulations represent a large portion of the computing needs. Detector simulations are primarily performed with Geant4[1], a widely used Monte Carlo toolkit that simulates the passage and interaction of particles through matter [1, 2, 3]. The current limited yearly increase in CPU performance cannot on its own keep up with the demand, and therefore exploitation of novel hardware and algorithmic improvements are required.

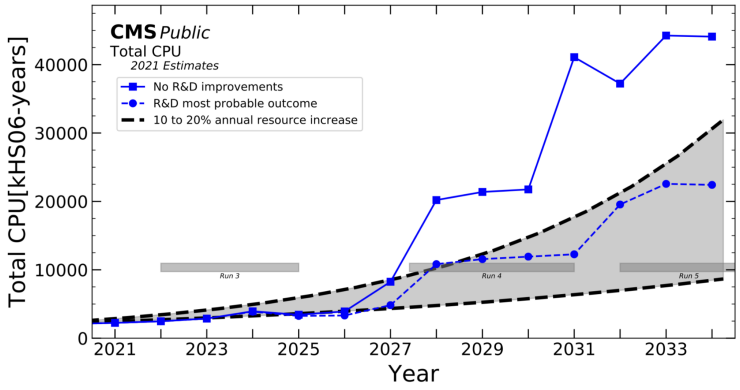

Figure 1: Projected CPU demand for upcoming CMS simulations, from [4].

Modern high-performance computing (HPC) hardware, particularly GPUs, offers new opportunities to signficiantly accelerate simulations. GPUs provide massive parallelism and high throughput for floating-point operations. However, porting physics simulations to this architecture is not straightforward. Particle transport involves complex control flow, frequent branching, and geometry-dependent calculations that are not naturally suited to the parallel execution model of GPUs.

CERN initiated AdePT[2] (Accelerated Demonstrator for Particle Transport), an R&D project that reimplements a subset of Geant4's functionalities for GPUs in CUDA. AdePT offloads only the transport of electrons, positrons, and photons to GPU, as their interactions are computationally simpler compared to hadrons or heavy ions. Despite this restriction, adapting the algorithms for geometry navigation and particle integration to GPUs requires significant redesign of data structures and numerical methods in order to minimize thread divergence and ensure efficient use of memory.

This project contributes to these efforts by investigating possible performance optimizations for the charged particle integration component of AdePT. In particular, we explore the feasibility of using lower precision arithmetic to speed up computations on GPUs, and developed improved methods for estimating distances to detector boundaries, with a focus on the complex geometries of ATLAS' electromagnetic calorimeter. These studies aim to reduce execution time while maintaining the accuracy required for physics simulations. In the following sections, we first describe the general numerical integration scheme implemented in AdePT, and then present the details of the two studies mentioned.

2 Integration scheme overview

2.1 Transport procedure

A charged particle moving through a medium inside a magnetic field follows a curved trajectory until it eventually interacts (with some probability) with the medium or reaches the boundary between two mediums. For example, a positron may eventually annihilate with an electron inside the material of a detector. If energetic enough, it may instead move through the medium with negligible energy deposition until it transitions to another material in the detector.

Given a particle with initial position $\mathbf{r}_i$ and momentum $\mathbf{p}_i$, the simulation of its trajectory can be summarized as follows.

[1] https://geant4.web.cern.ch
[2] https://github.com/apt-sim/AdePT

1. Compute the probabilistic curvilinear distance s_i the particle travels until it eventually interacts with the medium.
2. Compute the distance s_g the particle travels until reaching the closest geometry boundary.
3. Integrate the particle's trajectory over the *distance to next interaction* $s_n = \min(s_i, s_g)$ until its final state $\mathbf{r}_f$, $\mathbf{p}_f$.
4. If the particle was successfully integrated over s_n, simulate the corresponding interaction or keep track of the crossed boundary. Otherwise, repeat from step 1 using $\mathbf{r}_f$, $\mathbf{p}_f$ for $\mathbf{r}_i$, $\mathbf{p}_i$.

We give more details on steps 1, 2 and 3 in the following paragraphs.

2.2 Distance to next interaction

For the first step, we know the cross section σ of each possible interaction given the type of particle and medium it is traveling through. The mean free path λ of the particle for a particular interaction is then

$$\lambda = \frac{1}{n\sigma}$$

where n is the density of the medium. In the case of a particle traveling in vacuum, its mean free path is the average path until (potential) decay if it is unstable, and infinite instead. Then, the random distance to this interaction is calculated as

$$s_i = -\lambda \ln(r)$$

where r is sampled uniformly in $[0, 1]$. We repeat this computation for each possible interaction and keep the smallest s_i.

The second step is trickier. Since the particle travels a curved path, calculating the distance to the closest boundary intersection is not feasible. Instead, we estimate a safety d_s, which is the straight line distance from $\mathbf{r}_i$ to the closest surface, which is guaranteed to be such that $s_g \geq d_s$.

Consider the situation where a particle just crossed a boundary and points away from the surface. In that case, while the distance to the next encountered boundary could be large, the safety is 0 and thus s_n would be 0. We therefore keep track of which boundary each particle crossed last, and we thus know when a particle is inside its next volume. In that case, assuming a constant curvature from the initial state, we find the largest curvilinear distance s_c for which the difference between a chord starting at $\mathbf{r}_i$ remains below a *chord error* Δc, and choose $s_g := \max(d_s, s_c)$. The distance to next interaction is finally calculated as

$$s_n := \min(s_i, \max(d_s, s_c)).$$

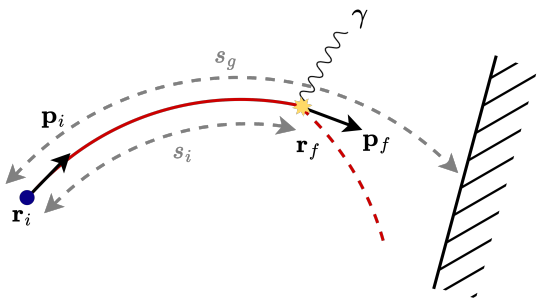

Figure 2: Distance to interaction and geometry of an electron interacting through Bremsstrahlung.

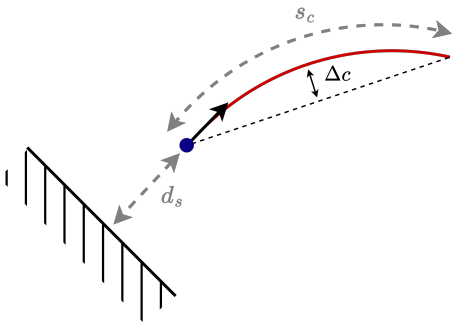

Figure 3: Visualization of the safety and chord length.

2.3 Adaptive integration scheme

We now need to integrate the trajectory of the particle over s_n until its final state $\mathbf{r}_f$, $\mathbf{p}_f$ from $\mathbf{r}_i$, $\mathbf{p}_i$. Since the particle evolves in a magnetic field, no work is applied hence its momentum p remains constant while only its direction $\hat{\mathbf{u}}$ changes. More precisely, we have

$$\mathbf{p} = p\hat{\mathbf{u}} \text{ with } p = \gamma m v$$

where m is the mass of the particle, v its velocity and γ the Lorentz factor for v. The general equations of motion expressed in terms of the traveled curvilinear distance s for a particle of charge q in a (potentially non-uniform) magnetic field $\mathbf{B}$ are

$$\frac{\mathrm{d}\mathbf{r}}{\mathrm{d}s} = \hat{\mathbf{u}}, \quad \frac{\mathrm{d}\mathbf{p}}{\mathrm{d}s} = \frac{q}{p}\hat{\mathbf{u}} \times \mathbf{B}(\mathbf{r}).$$

This system being non-linear, the trajectory must be solved numerically. Geant4 and AdePT use the Dormand-Prince integration scheme with an adaptive step size (DP45). This scheme consists in computing two 4th and 5th order Runge-Kutta steps to estimate an error that would be then used to fine tune the step size during integration. The objective is to reach the final state with as little steps as possible. The simplified general algorithm is described in Algorithm 1. An example of an integrated trajectory is shown in Figure 4.

Algorithm 1 Adaptive Dormand-Prince integrator

procedure DP45($\mathbf{r}_i, \mathbf{p}_i, s_f, \varepsilon_{\max}$, trials)
  $\mathbf{r}, \mathbf{p} \leftarrow \mathbf{r}_i, \mathbf{p}_i$
  $\Delta s \leftarrow s_f$
  $s \leftarrow 0$
  trial $\leftarrow 0$
  while $s < s_f$ and trial $\leq$ trials **do**
    $\mathbf{r}_4, \mathbf{p}_4 \leftarrow$ StepRK4$(\mathbf{r}, \mathbf{p}, \Delta s)$ ▷ 4th and 5th order Runge-Kutta steps
    $\mathbf{r}_5, \mathbf{p}_5 \leftarrow$ StepRK5$(\mathbf{r}, \mathbf{p}, \Delta s)$
    $\delta_r, \delta_p \leftarrow \|\mathbf{r}_5 - \mathbf{r}_4\|, \|\mathbf{p}_5 - \mathbf{p}_4\|$
    $\varepsilon = \max(\delta_r/\Delta s, \delta_p/p_i)/\varepsilon_{\max}$ ▷ Relative error
    if $\varepsilon \leq 1$ **then** ▷ Good: increase step size by growth power
      $\mathbf{r}, \mathbf{p} \leftarrow \mathbf{r}_4, \mathbf{p}_4$ ▷ Store new state and corresponding distance
      $s \leftarrow s + \Delta s$
      $\Delta s \leftarrow \varepsilon^{-1/5}\Delta s$ ▷ $\varepsilon \leq 1$ so $\varepsilon^{-1/5} \geq 1$
    else ▷ Bad: decrease step size by shrink power
      $\Delta s \leftarrow \varepsilon^{-1/4}\Delta s$ ▷ $\varepsilon > 1$ so $\varepsilon^{-1/4} < 1$
    end if
    trial $\leftarrow$ trial $+ 1$
  end while
  return $\mathbf{r}, \mathbf{p}, s$
end procedure

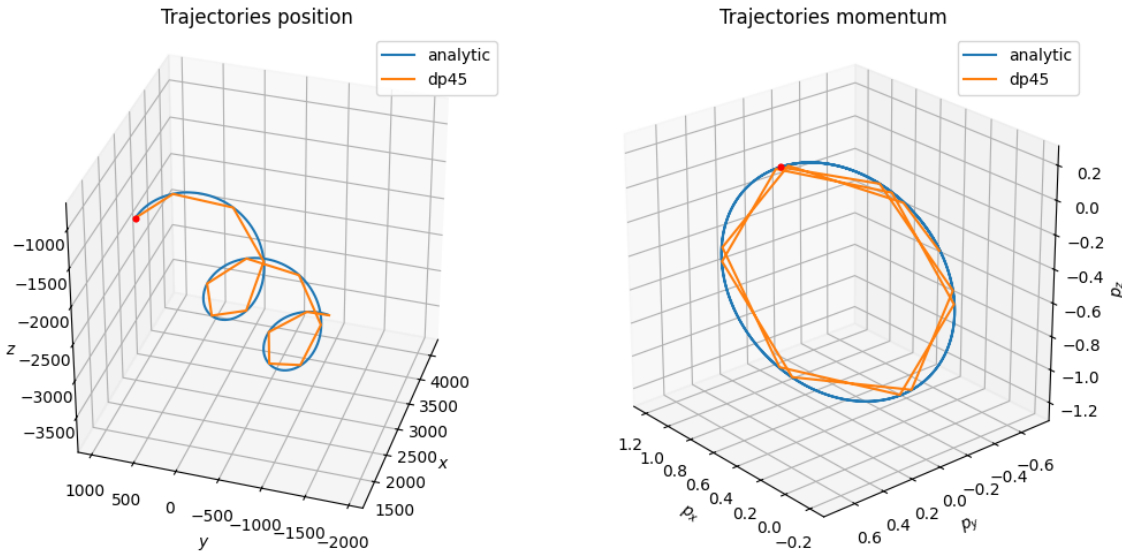

Figure 4: Visualization of the analytic and integrated position and momentum of an electron in a uniform magnetic field. The red dot marks the initial state.

3 Optimizing particle integration

3.1 Lower precision integration

Scientific simulations traditionally use double precision (64-bit) floating point types to represent real numbers (C's `double` type). This ensures a larger range and smaller machine epsilon than single precision (32-bit) floating points (C's `float` type), so that rounding errors have negligible effects on the integration error. However, arithmetic operations in double precision are more expensive than single precision. This difference is particularly significant on GPUs, where single precision arithmetic is more than an order of magnitude faster than double precision on consumer GPUs [5], and up to two times faster on HPC GPUs. Switching the precision of the particle simulation from double to single precision could therefore lead to a significant speedup. However, larger rounding errors will ultimately lead to larger errors in the integrated positions and momentum.

To test the feasibility of running particle simulation in float, we implement the DP45 integrator both in single and double precision, and compare the error of both integrators against the analytical trajectory in the simplified case of a uniform magnetic field.

When $\mathbf{B}$ is uniform, particles follow a helix aligned with the field's direction. More precisely, for a uniform field $\mathbf{B} = (0, 0, B)$, the motion of a particle with initial state $\mathbf{r}_i = (x_0, y_0, z_0)$, $\hat{\mathbf{u}}_i = (u_{0,x}, u_{0,y}, u_{0,z})$ is

$$\hat{\mathbf{u}}(s) = \begin{cases} u_{0,x}\cos(\omega s) + u_{0,y}\sin(\omega s) \\ u_{0,y}\cos(\omega s) - u_{0,x}\sin(\omega s) \\ u_{0,z} \end{cases} \qquad \mathbf{r}(s) = \begin{cases} x_0 + \frac{u_{0,x}}{\omega}\sin(\omega s) + \frac{u_{0,y}}{\omega}(1-\cos(\omega s)) \\ y_0 + \frac{u_{0,y}}{\omega}\sin(\omega s) - \frac{u_{0,x}}{\omega}(1-\cos(\omega s)) \\ z_0 + u_{0,z}s \end{cases}$$

where $\omega = qB/p$ is the synchrotron frequency. For a magnetic field in a direction other than along the z axis, we apply rotations to convert from and to a basis where the field is correctly aligned.

We simulate 10,000 trajectories of electrons placed in random magnetic fields with random initial states. For each particle, we sample uniformly in the ranges $0.1 \leq B \leq 4$ T, $1 \leq \mathbf{r}_i \leq 1000$ mm and 1 MeV $\leq \mathbf{p}_i \leq 10$ GeV. Each initial state is integrated for 100 target distances uniformly distributed from 0 to 10 m using the adaptive DP45 scheme. For each final state calculated, we compare them with the corresponding position and momentum returned by the analytic solution at the same distance. The error is averaged over all 10,000 tracks.

Figure 5 shows the average error over the distance. We observe that in practice, the average error induced by the loss of precision using floats instead of doubles is negligible. The error on the position increases by an order of 10^{-7} mm over 10 m. However, we should expect larger integration errors for more complex magnetic fields (e.g. quadrupoles or sextupoles).

To assess the performance gain of running the integration in float, we implement both float and double integration in CUDA. We simulate 16,384 particles with random initial states but same uniform magnetic field. All simulations are launched in a single CUDA kernel on an RTX 3080. Each thread integrates a single particle over 10 m and only stores the final state. The maximum number of trials is limited to 128. The execution time is averaged over 100 repeats. The benchmark results for different block sizes are shown in Table 1.

Block size	**Number of blocks**	**Run time (ms)**		**Single speed up**
		Double precision	**Single precision**	
32	512	0.494	0.0689	+720%
64	256	0.163	0.0632	+260%
128	128	0.166	0.0659	+215%
256	64	0.190	0.0867	+220%

Table 1: Average run time comparison of the integration of 16,384 particles on GPU.

These large speedups however are only relevant for isolated integration. In production runs, integration steps are separated by regular calls to geometry routines and interaction simulations. To measure the actual possible speedup in a full simulation, we run ttbar events in the CMS 2018 geometry in a uniform 3.8 T magnetic field using AdePT with mixed precision. That is, most variables are kept in double, while the internal integration steps are done in float. We run the simulation with 16 threads, 14 million track slots and 10 million hit slots. We observe a speedup of the order of **10%**. Further profiling is required to check the physical consistency of the runs against the double precision baseline.

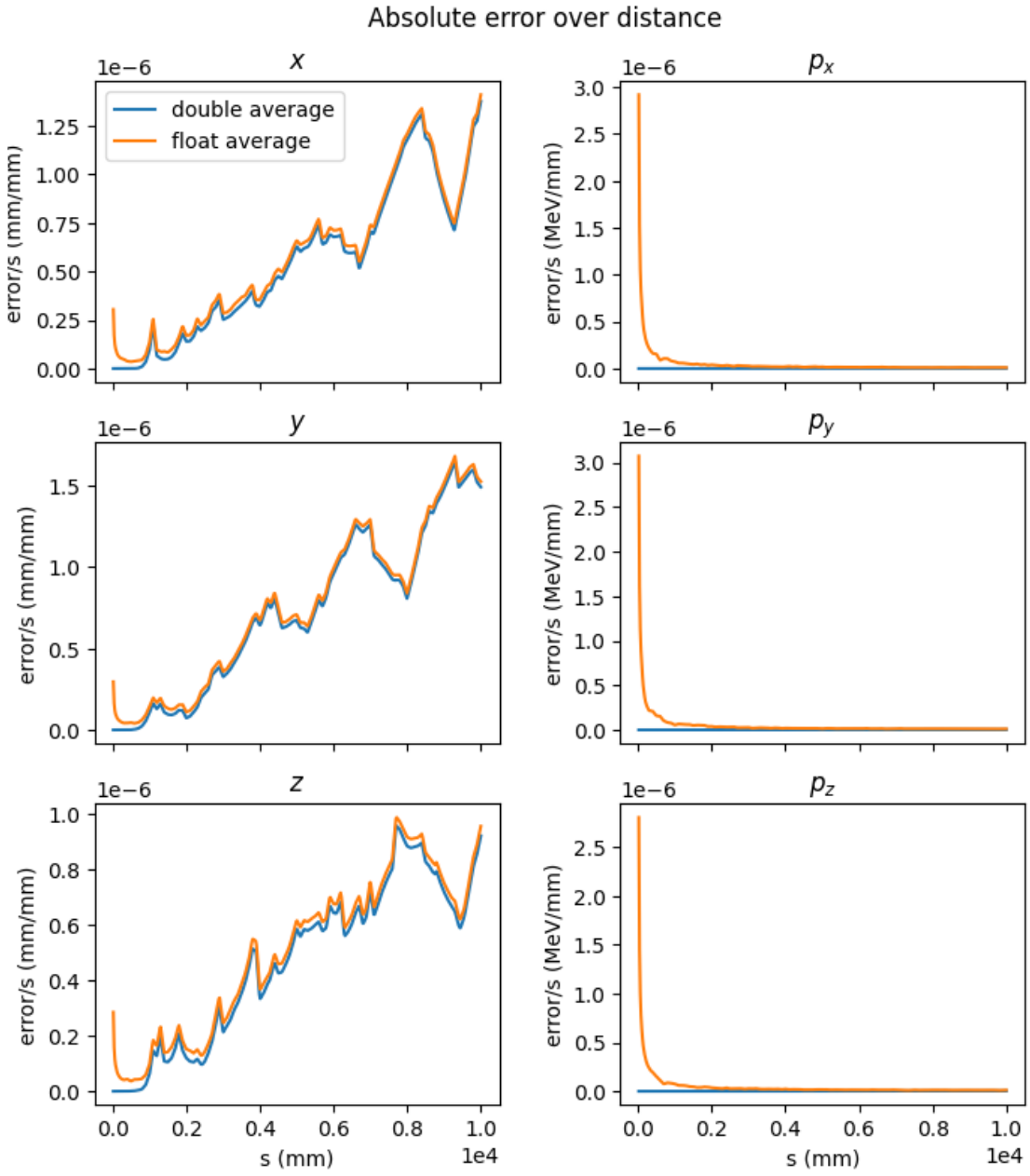

Figure 5: Absolute error of double and float integration against analytical solution.

3.2 Faster safety estimation

The main advantage of the adaptive step size in the DP45 integrator is to limit the number of steps taken when the distance to integrate is small enough so that the error between 4th and 5th order integration is negligible. This means that we should aim to maximize the distance to next interaction. This would prevent the integrator from doing more steps than required if a single step would be sufficient.

Since the distance to interaction with the medium s_i is random, we can only try to maximize the estimate $s_g = \max(d_s, s_c)$. In particular, the safety d_s is the straight distance from the particle's position to the closest point surface point. For many geometries, d_s cannot be computed analytically, and even if it could, it may be computationally expensive. For performance considerations, the safety must therefore be computed so that

- it is fast to compute;
- it always *underestimates* the exact distance.

The first condition means the safety should be computed with few arithmetic operations and avoid conditional branching (critical in GPU performance). The second condition is necessary to ensure we do not miss a boundary.

The Electromagnetic Endcap Calorimeter (EMEC) in ATLAS contains absorber and electrode plates that are shaped in an accordion structure (shown in Figure 7). These are placed in a circular pattern inside the detector (shown in Figure 6). This structure was initially implemented as a custom geometry in Geant4 which showed poor performances. They would later be modeled via the builtin *generic trapezoid* (gentrap) solid in Geant4 (described in [6]), which consists in two quadrilateral connected by straight but tilted edges with twisted lateral surfaces. A visualization of this solid is shown in Figure 8.

Figure 9 shows number of calls to the safety estimation routines for the different Geant4 volumes used in an ATLAS simulation run. Since the box safety is used inside the gentrap safety, we see that the large majority of safety estimation in EMEC relate to gentrap. Hence, performance of its safety estimator is critical. In particular, the distance to twisted side surfaces (namely parabolic hyperboloids) can hardly be calculated analytically. One can show that the exact distance can be found by solving the root of a five degree polynomial, which has no general expressions in terms of radicals. Therefore, the distance to the lateral surfaces must be approximated.

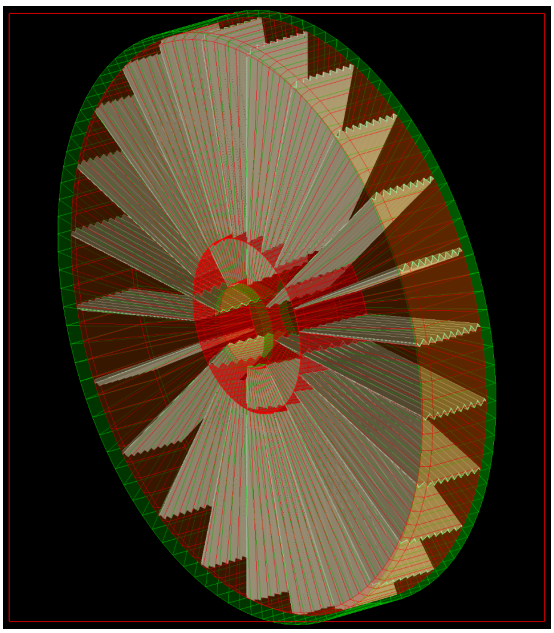

Figure 6: EMEC.

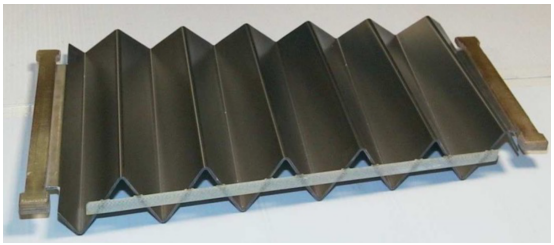

Figure 7: EMEC's absorbers.

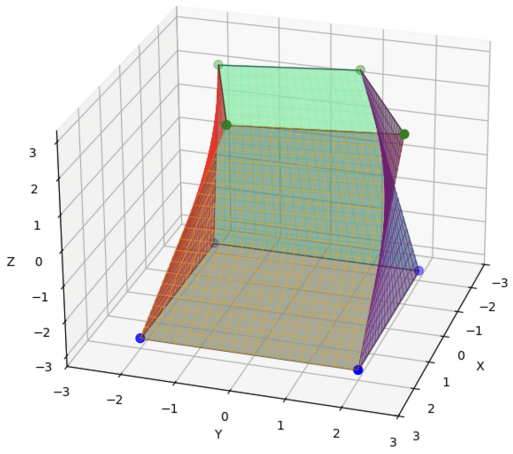

Figure 8: Visualization of a strongly twisted generic trapezoid.

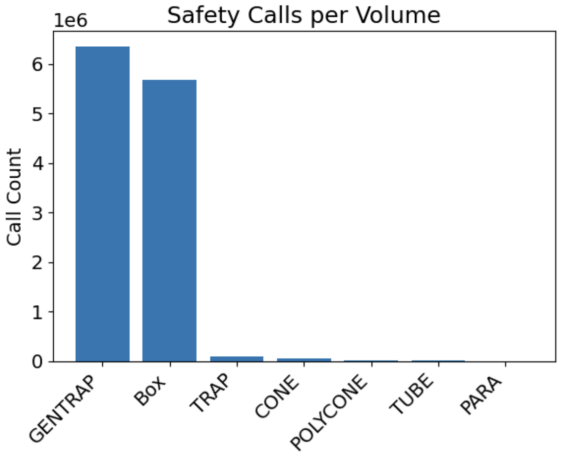

Figure 9: Safety calls per volume in ATLAS run.

A lateral surface of the generic trapezoid is a quadric surface S defined implicitly by $f(x, y, z) = 0$ where f is of the form

$$f(x, y, z) = Axz + Byz + Cz^2 + Dx + Ey + Fz + G.$$

Each coefficient A, B, C, D, E, F, G is precomputed for the coordinates of the points defining each surface. The exact distance from a point $\mathbf{x}$ to the S cannot be computed analytically. However, using the Lipschitz continuity of the gradient, we can derive the following guaranteed underestimation of the true distance from $\mathbf{x}$:

$$\hat{d}(\mathbf{x}) = \frac{2|f(\mathbf{x})|}{\|\nabla f(\mathbf{x})\| + \sqrt{\|\nabla f(\mathbf{x})\|^2 + 4k|f(\mathbf{x})|}} \quad \text{with} \quad k = |C| + \sqrt{A^2 + B^2 + C^2}.$$

We test the underestimation $\hat{d}$ for different twisted surfaces. Similarly to how gentrap's side surfaces are defined, we set two points $A_1 = (-1, 0, -1), B_1 = (1, 0, -1)$ to form the lower segment, and $A_2 = (-1, \delta y, 1), B_2 = (1, -\delta y, 1)$ as the upper segment. The coefficients in f are precomputed from the coordinates of these points. Then, we sample $20 \times 20 \times 20$ points in a grid around the surface in the range $[-10, 10]^3$. We estimate the closest distance using the first order approximation

$$d_1(\mathbf{x}) = \frac{|f(\mathbf{x})|}{\|\nabla f(\mathbf{x})\|}$$

and the guaranteed underestimation $\hat{d}$. We compare these approximations to the real closest distance which we obtain by numerically solving the minimization problem

$$d(\mathbf{x}) = \min_{\mathbf{y} \in S} \|\mathbf{x} - \mathbf{y}\|.$$

We run this procedure for $\delta y = 0.02, 0.1, 0.5$. The corresponding k values are $0.08, 0.4, 2$. Figure 10 shows a visualization of the surfaces, while Figure 11 compares $\hat{d}$ and d_1 against d.

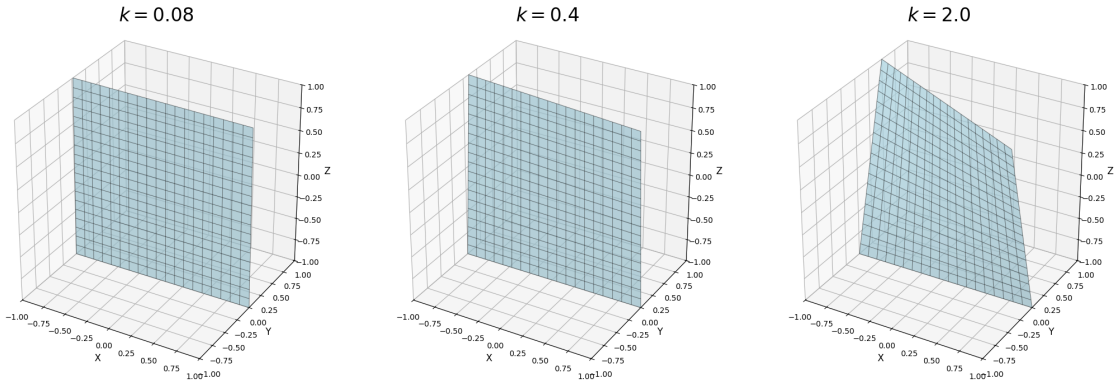

Figure 10: Visualization of the twisted surfaces with different twist levels.

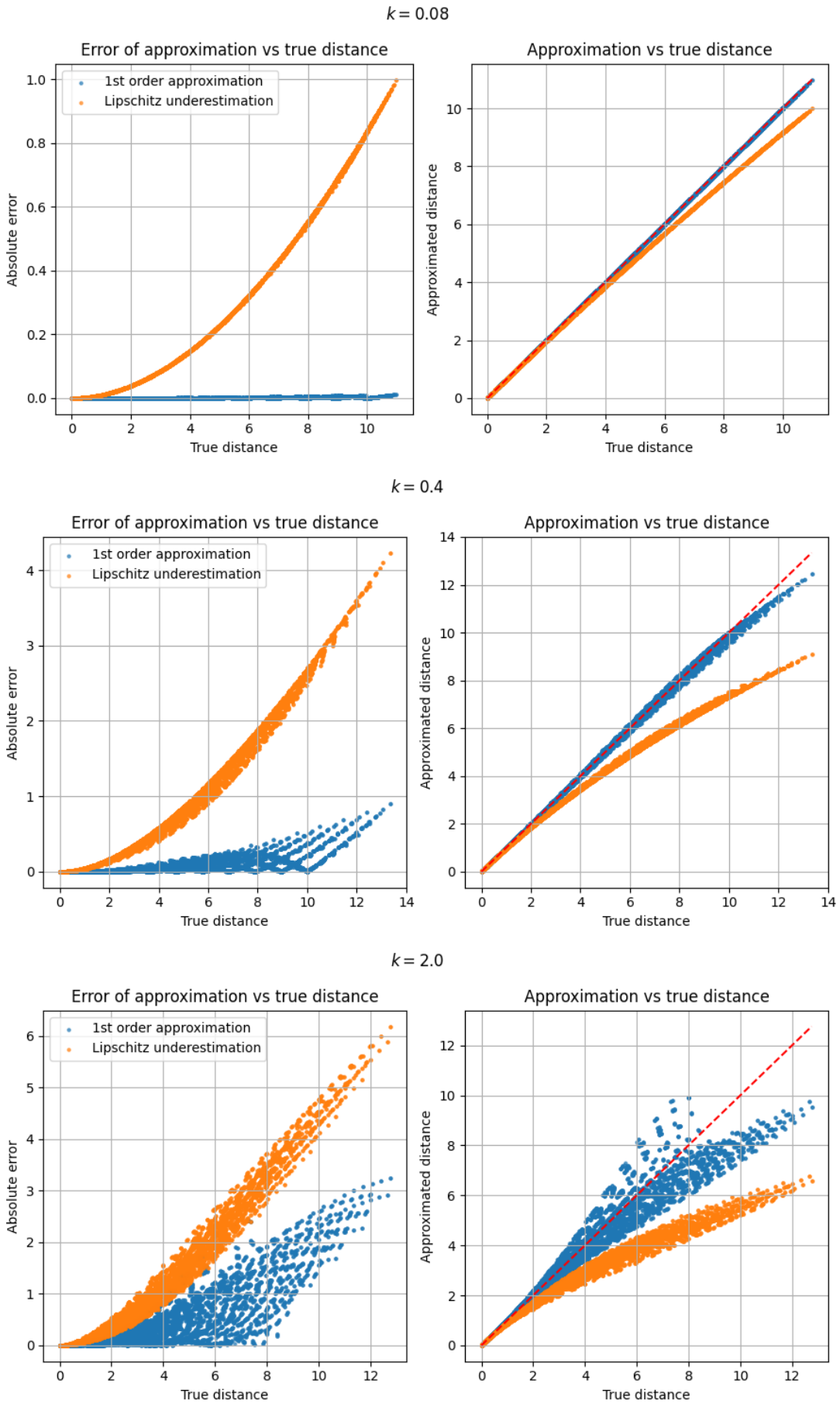

Figure 11: Approximation error on the generic trapezoid's twisted surfaces for different twist levels.

The current implementation of the safety for the generic trapezoid in Geant4 approximates side surfaces by two triangles that fully surround the surface. In actual runs, the absorbers' sides are only slightly twisted ($k \ll 1$). Moreover, Geant4 organizes geometry in a voxel grid, while AdePT organizes it in a bounding volume hierarchy tree. This means that calls to gentrap's safety are mostly done in the situation when the underestimation $\hat{d}$ is the most accurate (top most plot in Figure 11). In practice, using the proposed estimator $\hat{d}$ reduces the number of integration steps by **5%** in EMEC benchmarks. The speedup in a full ATLAS production run is expected to be of the order of 1-2%.

4 Conclusion

The increasing computational requirements of HL-LHC simulations require new algorithmic approaches that fully exploit modern parallel architecture. We've investigated potential optimizations for the integration of charged particles on GPUs within the AdePT framework as well as Geant4. Low precision arithmetic shows promising use on GPU because of its almost negligible increase in the integration error in comparison to large speedups. We also successfully derived a generic method for fast safety calculations in the complex geometries that are present in the current and upcoming HL-LHC runs.

A necessary continuation of this work is to validate the use of single precision integration in more complex and realistic magnetic field configurations, such as those actually measured in the detectors. Similarly, the safety estimation method could be generalized to other non-trivial detector geometries other than the EMEC. It was also noticed that some twisted solids available in Geant4 currently have computationally heavy implementations. While not in use in any CERN experiment at the moment, some of these more complex geometries may be required in future projects such as FCC. More generally, future efforts will need to address the transport of hadrons and heavy ions, where physics models are more complex and GPU parallelization even more challenging.

References

[1] Recent Developments in Geant4, J. Allison et al., Nucl. Instrum. Meth. A 835 (2016) 186-225

[2] Geant4 Developments and Applications, J. Allison et al., IEEE Trans. Nucl. Sci. 53 (2006) 270-278

[3] Geant4 - A Simulation Toolkit, S. Agostinelli et al., Nucl. Instrum. Meth. A 506 (2003) 250-303

[4] CMS NOTE – 2022/008, https://cds.cern.ch/record/2815292

[5] NVIDIA Ampere GA102 GPU Architecture whitepaper, https://www.nvidia.com/content/PDF/nvidia-ampere-ga-102-gpu-architecture-whitepaper-v2.pdf

[6] New EMEC description with constructed solids, Detector Description Session During S&C Workshop, June 2023, https://indico.cern.ch/event/1287543/contributions/5442708/attachments/2663884/4615608/2023.06.12-EMEC_ATLAS_SC_Week.pdf