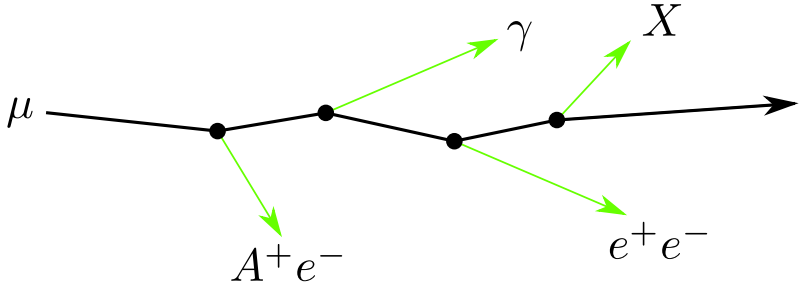
technische universität
dortmund



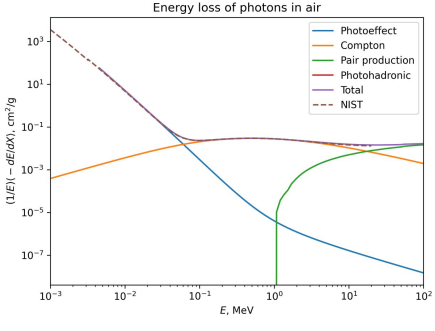
Updates from EM - Photoeffect and Runtime analysis

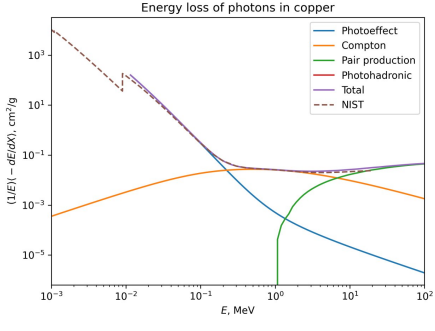
Jean-Marco Alameddine

23.06.2022

CORSIKA general call

Photoeffect

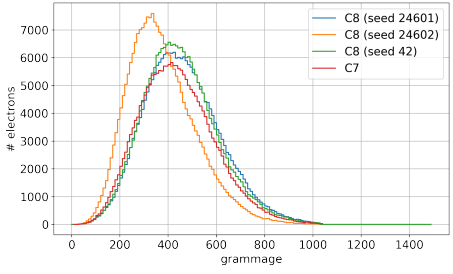


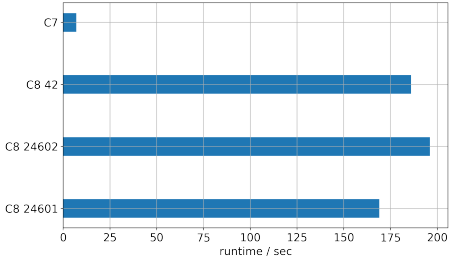


- Analytical formula to describe photoeffect has been implemented in PROPOSAL
  - → Implementation just needs to be finalized in PROPOSAL
  - → Will be included in the next PROPOSAL release
  - → Can directly be used in CORSIKA by updating the version

Runtime analysis - General information

- Looking at **10** TeV electron-induced showers, **2** MeV cuts
- Branch `502-examples-need-some-polishing`, commit `94875ed4`





→ Note: Shower profiles in both C7 and C8 are highly fluctuating

→ Runtime difference between C7 and C8 of an order of magnitude (≈factor 25)

1. Step: Runtime analysis

- Runtime profiling of a single **10** TeV shower
- Using `perf 5.4.0` with a sampling frequency of 1000 samples per second
  - → Count how much time is spent in which function
- Magnetic field is enabled
- TrackWriter has been disabled (otherwise, there is an extra factor of **1.5** in runtime!)

	Function name	runtime (% of total)
1.	`proposal::ContinuousProcess::doContinuous`	25.4 %
	→ `PROPOSAL::UtilityInterpolant::GetUpperLimit`	→ 9.8 %
	→ `SlidingPlanarExponential::getIntegratedGrammage`	→ 6.6 %
	→ `LeapFrogTrajectory::getPosition`	→ 4.6 %
	→ `PROPOSAL::multiple_scattering`	→ 2.5 %
2.	`PROPOSAL::Interaction::MeanFreePath`	19.5 %
	→ `cubic_splines::BicubicSplines::evaluate`	→ 5.9 %
	→ `PROPOSAL::CrossSectionDNDX::GetIntegrationLimits`	→ 8.1 %
3.	`SlidingPlanarExponential::getArclengthFromGrammage`	13.4 %
	→ `LeapFrogTrajectory::getPosition`	→ 9.1 %
	→ `LeapFrogTrajectory::getDirection`	→ 2.3 %
4.	`tracking_leapfrog_curved::Tracking::getTrack`	11.1 %
	→ `Intersect::nextIntersect`	→ 8.4 %
5.	`LeapFrogTrajectory::getPosition`	6.7 %
6.	`proposal::InteractionModel::doInteraction`	3.9 %
7.	`ParticleCut::checkCutParticle`	3.7 %
8.	`tracking_leapfrog_curved::Tracking::intersect`	2.1 %

	Function name	runtime (% of total)
1.	`proposal::ContinuousProcess::doContinuous`	25.4 %
	→ `PROPOSAL::UtilityInterpolant::GetUpperLimit`	→ 9.8 % ❓
	→ `SlidingPlanarExponential::getIntegratedGrammage`	→ 6.6 %
	→ `LeapFrogTrajectory::getPosition`	→ 4.6 %
	→ `PROPOSAL::multiple_scattering`	→ 2.5 %
2.	`PROPOSAL::Interaction::MeanFreePath`	19.5 %
	→ `cubic_splines::BicubicSplines::evaluate`	→ 5.9 %
	→ `PROPOSAL::CrossSectionDNDX::GetIntegrationLimits`	→ 8.1 % ❗
3.	`SlidingPlanarExponential::getArclengthFromGrammage`	13.4 %
	→ `LeapFrogTrajectory::getPosition`	→ 9.1 %
	→ `LeapFrogTrajectory::getDirection`	→ 2.3 %
4.	`tracking_leapfrog_curved::Tracking::getTrack`	11.1 %
	→ `Intersect::nextIntersect`	→ 8.4 %
5.	`LeapFrogTrajectory::getPosition`	6.7 %
6.	`proposal::InteractionModel::doInteraction`	3.9 %
7.	`ParticleCut::checkCutParticle`	3.7 %
8.	`tracking_leapfrog_curved::Tracking::intersect`	2.1 %

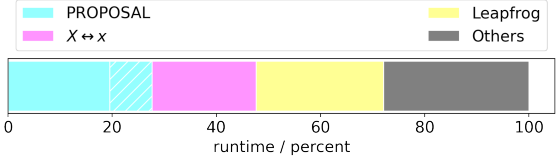
- **35.7 %** of runtime spent directly in PROPOSAL
- However, the **8.1 %** spent in `PROPOSAL::CrossSectionDNDX::GetIntegrationLimits` are unnecessary!
  - → This will be fixed in the next PROPOSAL release so this runtime can be saved! (see PROPOSAL PR #295)

	Function name	runtime (% of total)
1.	`proposal::ContinuousProcess::doContinuous`	25.4 %
	→ `PROPOSAL::UtilityInterpolant::GetUpperLimit`	→ 9.8 %
	→ `SlidingPlanarExponential::getIntegratedGrammage`	→ 6.6 % ❓
	→ `LeapFrogTrajectory::getPosition`	→ 4.6 %
	→ `PROPOSAL::multiple_scattering`	→ 2.5 %
2.	`PROPOSAL::Interaction::MeanFreePath`	19.5 %
	→ `cubic_splines::BicubicSplines::evaluate`	→ 5.9 %
	→ `PROPOSAL::CrossSectionDNDX::GetIntegrationLimits`	→ 8.1 %
3.	`SlidingPlanarExponential::getArclengthFromGrammage`	13.4 % ❓
	→ `LeapFrogTrajectory::getPosition`	→ 9.1 %
	→ `LeapFrogTrajectory::getDirection`	→ 2.3 %
4.	`tracking_leapfrog_curved::Tracking::getTrack`	11.1 %
	→ `Intersect::nextIntersect`	→ 8.4 %
5.	`LeapFrogTrajectory::getPosition`	6.7 %
6.	`proposal::InteractionModel::doInteraction`	3.9 %
7.	`ParticleCut::checkCutParticle`	3.7 %
8.	`tracking_leapfrog_curved::Tracking::intersect`	2.1 %

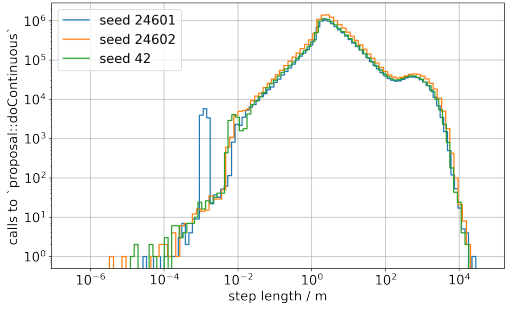
- **20 %** of runtime spent in functions transforming from grammage to distance (and vice versa)
  - → Maybe one can make assumptions (like local densities) to save runtime?
  - → However, this will probably bring (steplength) limitations...

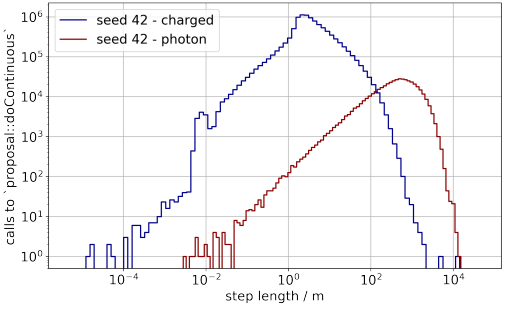
	Function name	runtime (% of total)
1.	`proposal::ContinuousProcess::doContinuous`	25.4 %
	→ `PROPOSAL::UtilityInterpolant::GetUpperLimit`	→ 9.8 %
	→ `SlidingPlanarExponential::getIntegratedGrammage`	→ 6.6 %
	→ `LeapFrogTrajectory::getPosition`	→ 4.6 %
	→ `PROPOSAL::multiple_scattering`	→ 2.5 %
2.	`PROPOSAL::Interaction::MeanFreePath`	19.5 %
	→ `cubic_splines::BicubicSplines::evaluate`	→ 5.9 %
	→ `PROPOSAL::CrossSectionDNDX::GetIntegrationLimits`	→ 8.1 %
3.	`SlidingPlanarExponential::getArclengthFromGrammage`	13.4 %
	→ `LeapFrogTrajectory::getPosition`	→ 9.1 %
	→ `LeapFrogTrajectory::getDirection`	→ 2.3 %
4.	`tracking_leapfrog_curved::Tracking::getTrack`	11.1 %
	→ `Intersect::nextIntersect`	→ 8.4 %
5.	`LeapFrogTrajectory::getPosition`	6.7 %
6.	`proposal::InteractionModel::doInteraction`	3.9 %
7.	`ParticleCut::checkCutParticle`	3.7 %
8.	`tracking_leapfrog_curved::Tracking::intersect`	2.1 %

- 24.5 % of runtime spent in functions dealing with `LeapFrog`
  - → Could this be optimized? Has this been optimized?



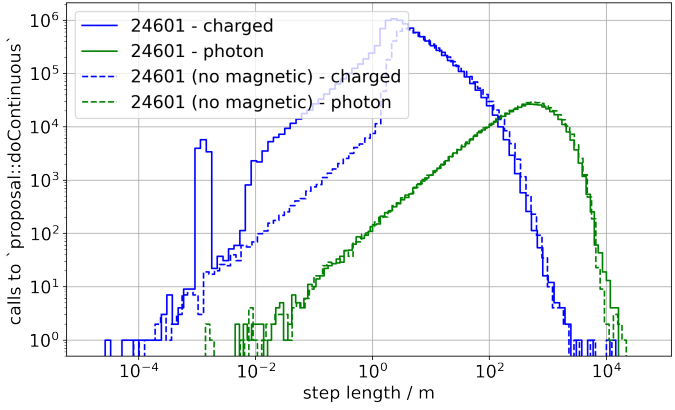
- First interpretation: No obvious, single source, where a lot of runtime is spent
- Idea: Maybe we make too many steps (compared to CORSIKA 7), which causes everything to be slow
  - → Idea: Look at the steplengths that are made within CORSIKA 8
  - → Make a histogram of all calls to `proposal::ContinuousProcess::doContinuous` with the associated steplengths!





- Two clearly separated structures can be seen
- Comparison with CORSIKA 7 would be very interesting
  - → Currently talking to Dominik B. to extract step length information from CORSIKA 7

Appendix



- Showers without magnetic field are about **30 %** faster
- There are less small steps made for charged particles