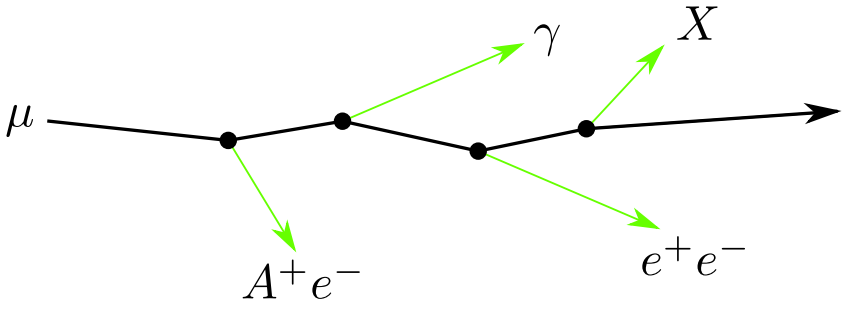
technische universität dortmund

e5 experimentelle physik 5
astroteilchenphysik

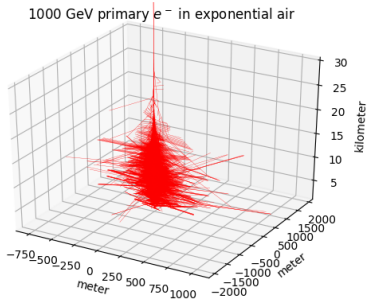


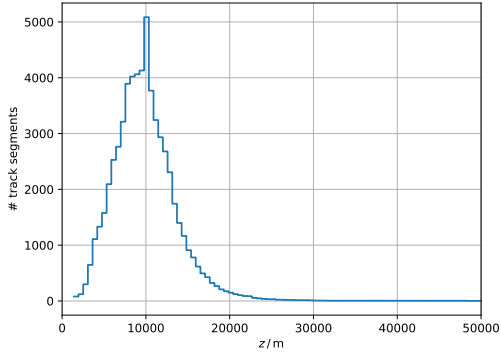
First em-Cascade of CORSIKA8 using PROPOSAL

Jean-Marco Alameddine, Maximilian Sackel, Jan Soedingrekso, Alexander Sandrock

July 16, 2020

Technische Universität Dortmund


1000 GeV primary e⁻ in exponential air
kilometer
meter
meter


track segments
z / m

General structure

```
template <typename Particle>
auto GetCalculator(Particle& vP) {
  auto& comp = vP.GetNode()->GetModelProperties().GetNuclearComposition();
  auto calc_it = calculators.find(make_pair(&comp, vP.GetPID()));
  if (calc_it != calculators.end()) return calc_it;
  return BuildCalculator(vP.GetPID(), comp);
}
```

- Make use of interpolation tables (set `PROPOSAL::InterpolationDef::path_to_tables`)
- calculated the first time it is required (take some time $O(10^2\,\mathrm{s})$)

General structure

```
template <typename Particle>
auto BuildCalculator(particles::Code code, Particle p_def, NuclearComposition const& comp) {
  auto cross = GetStdCrossSections(p_def, media.at(&comp),
      make_shared<const PROPOSAL::EnergyCutSettings>(cut.GetECut() / 1_MeV, 1, false),
      true);
  auto inter_types = PROPOSAL::CrossSectionVector::GetInteractionTypes(cross);
  auto [insert_it, success] = calculators.insert(
      {make_pair(&comp, code),
       make_tuple(PROPOSAL::make_secondaries(inter_types, p_def, media.at(&comp)),
                  PROPOSAL::make_interaction(cross, true))});
  return insert_it;
}
```

- Same energy cut assumed for all particles tracked by PROPOSAL
- Continuous energy losses are not randomized

Interaction

```
template <>
Interaction::Interaction(SetupEnvironment const& _env, CORSIKA_ParticleCut& _cut)
  : cut(_cut)
  , fRNG(corsika::random::RNGManager::GetInstance().GetRandomStream("proposal"))
{
    auto all_compositions = std::vector<const NuclearComposition*>();
    _env.GetUniverse()->walk([&](auto& vtn) {
      if (vtn.HasModelProperties())
        all_compositions.push_back(&vtn.GetModelProperties().GetNuclearComposition());
    });
    for (auto& ncarg : all_compositions) {
      auto comp_vec = std::vector<Component_PROPOSAL>();
      auto frac_iter = ncarg->GetFractions().cbegin();
      for (auto& pcode : ncarg->GetComponents()) {
        comp_vec.emplace_back(GetName(pcode), GetNucleusZ(pcode), GetNucleusA(pcode),
                              *frac_iter);
        ++frac_iter;
      }
      media[ncarg] = PROPOSAL::Medium(
          "Modified Air", 1., PROPOSAL::Air().GetI(), PROPOSAL::Air().GetC(),
          PROPOSAL::Air().GetA(), PROPOSAL::Air().GetM(), PROPOSAL::Air().GetX0(),
          PROPOSAL::Air().GetX1(), PROPOSAL::Air().GetD0(), 1.0, comp_vec);
    }
}
```

Interaction

```
template <>
corsika::units::si::GrammageType Interaction::GetInteractionLength(setup::Stack::StackIterator const& vP) {
    if (CanInteract(vP.GetPID())) {
      auto calc = GetCalculator(vP);
      return get<INTERACTION>(calc->second)->MeanFreePath(vP.GetEnergy() / 1_MeV) * 1_g / (1_cm * 1_cm);
    }
    return std::numeric_limits<double>::infinity() * 1_g / (1_cm * 1_cm);
}
```

$$\int_{x_0}^{x} \frac{dx}{\lambda(x)} = -\ln \xi$$

- leptons lose significant parts of their energy due to continuous losses

→ only small stepsizes are allowed (too many stochastic interactions)

Interaction

```
template <>
corsika::process::EProcessReturn Interaction::DoInteraction(setup::StackView::StackIterator& vP) {
  if (CanInteract(vP.GetPID())) {
    auto calc = GetCalculator(vP); // [CrossSections]@
    std::uniform_real_distribution<double> distr(0., 1.);
    auto [type, comp_ptr, v] = get<INTERACTION>(calc->second)->TypeInteraction(vP.GetEnergy() / 1_MeV, distr(fRNG));
    auto rnd = vector<double>(get<SECONDARIES>(calc->second)->RequiredRandomNumbers(type));
    for (auto& it : rnd) it = distr(fRNG);
    auto point = PROPOSAL::Vector3D(vP.GetPosition().GetX() / 1_cm, vP.GetPosition().GetY() / 1_cm, vP.GetPosition().GetZ() / 1_cm);
    auto d = vP.GetDirection().GetComponents();
    auto direction = PROPOSAL::Vector3D(d.GetX().magnitude(), d.GetY().magnitude(), d.GetZ().magnitude());
    auto loss = make_tuple(static_cast<int>(type), point, direction, v * vP.GetEnergy() / 1_MeV, 0.);
    auto sec = get<SECONDARIES>(calc->second)->CalculateSecondaries(vP.GetEnergy() / 1_MeV, loss, *comp_ptr, rnd);
    for (auto& s : sec) {
      auto E = get<PROPOSAL::Loss::ENERGY>(s) * 1_MeV;
      auto vec = corsika::geometry::QuantityVector( get<PROPOSAL::Loss::DIRECTION>(s).GetX() * E,
          get<PROPOSAL::Loss::DIRECTION>(s).GetY() * E, get<PROPOSAL::Loss::DIRECTION>(s).GetZ() * E);
      auto p = corsika::stack::MomentumVector(
          corsika::geometry::RootCoordinateSystem::GetInstance().GetRootCoordinateSystem(), vec);
      auto sec_code = corsika::particles::ConvertFromPDG(static_cast<particles::PDGCode>(get<PROPOSAL::Loss::TYPE>(s)));
      vP.AddSecondary(make_tuple(sec_code, E, p, vP.GetPosition(), vP.GetTime()));
    }
  }
  return process::EProcessReturn::eOk;
}
```

Continuous

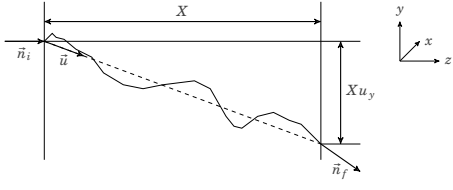
```
template <>
EProcessReturn ContinuousProcess::DoContinuous(SetupParticle& vP, SetupTrack const& vT) {
  if (vP.GetChargeNumber() == 0) return process::EProcessReturn::eOk;
  auto dX = vP.GetNode()->GetModelProperties().IntegratedGrammage(vT, vT.GetLength());
  auto energy_loss = TotalEnergyLoss(vP, dX);
  Scatter(vP, energy_loss, dX);
  vP.SetEnergy(vP.GetEnergy() - energy_loss);
  if (vP.GetEnergy() < cut.GetECut()) return process::EProcessReturn::eParticleAbsorbed;
  vP.SetMomentum(vP.GetMomentum() * vP.GetEnergy() / vP.GetMomentum().GetNorm());
  return process::EProcessReturn::eOk;
}
```

? particles that are below the energy threshold?

? energy conservation violation if particles are below the energy threshold (absorbed)

Continuous

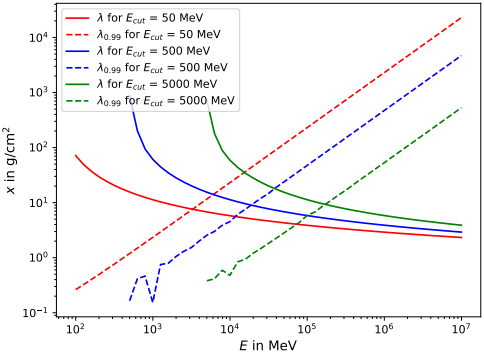
```
template <>
void ContinuousProcess::Scatter(SetupParticle& vP, HEPEnergyType const& loss, GrammageType const& grammage) {
  auto calc = GetCalculator(vP);
  auto d = vP.GetDirection().GetComponents();
  auto direction = PROPOSAL::Vector3D(d.GetX().magnitude(), d.GetY().magnitude(), d.GetZ().magnitude());
  auto E_f = vP.GetEnergy() - loss; // final energy@
  std::uniform_real_distribution<double> distr(0., 1.);
  auto rnd = array<double, 4>();
  for (auto& it : rnd) it = distr(fRNG);
  auto [mean_dir, final_dir] = get<SCATTERING>(calc->second)
          ->Scatter(grammage / 1_g * square(1_cm), vP.GetEnergy() / 1_MeV, E_f / 1_MeV, direction, rnd);
  auto vec = corsika::geometry::QuantityVector(final_dir.GetX() * E_f, final_dir.GetY() * E_f, final_dir.GetZ() * E_f);
  vP.SetMomentum(corsika::stack::MomentumVector(
      corsika::geometry::RootCoordinateSystem::GetInstance().GetRootCoordinateSystem(), vec));
}
```



- deflect particle momentum
- ? move particle position independent from `SetupTrack const&`

Continuous

```
template <>
units::si::LengthType ContinuousProcess::MaxStepLength(SetupParticle const& vP, SetupTrack const& vT) {
  auto energy_lim = 0.9 * vP.GetEnergy() / 1_MeV;
  auto calc = GetCalculator(vP);
  auto grammage = get<DISPLACEMENT>(calc->second) ->SolveTrackIntegral(vP.GetEnergy() / 1_MeV, energy_lim) *
                  1_g / square(1_cm);
  return vP.GetNode()->GetModelProperties().ArclengthFromGrammage(vT, grammage) * 1.0001;
}
```



- Current propagation method limits the maximal steplength ($\lambda_{0.99}$)
- For low energies, the optimal steplength could be larger (~ λ)

→ Loss of efficiency when $\lambda_{0.99} < \lambda$

→ Very slow propagation for very low energies

Current Installation

```
# install PROPOSAL
git clone --single-branch --depth=1 --recursive --branch restructure_parametrization
    https://github.com/tudo-astroparticlephysics/PROPOSAL.git
mkdir PROPOSAL/build
cd PROPOSAL/build
cmake install .. -DADD_TESTS=OFF -DADD_PYTHON=OFF -DADD_CPPEXAMPLE=OFF
make install -j


# install corsika
git clone --single-branch --branch include_proposal_process https://gitlab.ikp.kit.edu/AirShowerPhysics/corsika.git
mkdir corsika/build
cd corsika/build
export PROPOSAL_INCLUDE_DIR=...
export PROPOSAL_LIBRARY=...
cmake ..
make install
```

MR !222

- see WIP: Move third party dependencies into git submodules
- is this the way to include proposal into the installation process?
- who takes care of this?