Next-generation CORSIKA

Particle cascade simulation framework

There exist a large scientific community that depends on precise modelling of complex particle cascading processes in various types of matter. The most obviously related fields are cosmic ray physics, astrophysical neutrino physics, and gamma ray physics. This document summarizes the steps needed to ensure that the optimal simulation tools are kept available in the future. The main considerations and options are given, and the open questions are highlighted.

1 Introduction, History, Context

CORSIKA is the leading tool and of tremendous importance for the research field related to air shower processes. It has been used from calculating inclusive particle fluxes up to simulating ultra-high energy extensive air showers. It has supported and shaped reasearch during the last ≈20 years. Originally designed as a fortran 77 program as part of the detector simulation for the KASCADE experiment, it has evolved enourmously over time and is used now by essentially all cosmic ray experiments. The lack of flexibility in fortran has lead to a not very maintainable significant piece of highly complex software. While the performance is still excellent and the mainstream usecases are fequently tested as well as verified, it is more-and-more difficult to keep development up with requests and requirements. It is obvious that the complexity of the fortran code and the complexity of new developments are getting in conflic with each other. It is of paramount importance to make CORSIKA competitive for the future, requiring a major step in software technology. This will ensure that CORSIKA will continue to be the best and most comprehensive tool for air shower simulations and related tasks.

2 Purpose, Aim

The purpose of CORSIKA is a “particle transport code with stochastic and continuous processes”. A *next-generation CORSIKA* (ngC) will implement this core task in the most direct, flexible and efficient way. The ngC will provide a frameworke where all users can provide plugins and extensions for an unspecified number of scientific problems to come. CORSIKA will take a final step from being an air shower simulation program, to becoming the most powerful framework for particle cascade simulations available.

The ngC must support particle tracking, cascade equations, thinning, various particle interaction models, output options, parallel operations, GPU support, various possibilities for user routines to interact with the simulation process, full exposure of particles while they are tracked/simulated. Production of Cherenkov photons, radio signals, molecular bremsstrahlung should be much facilitated in this way and provided as standard output options.

ngC will be the necessary work horse for cosmic ray, neutrio and TeV/EeV gamma ray astronomy.

Millions to billions of CPU hours of high performance computing will be spend on air shower simulations for CTA, IceCUBE, HESS, Magic, Auger, TA. It is up to ngC to make sure this is done as efficiently as possible and to maximise the resulting physics output. In this respect ngC plays an important role in spending of valuable resources and is at the same time a fundamental cornerstone to support the physics output of many large experiments.

3 Main design components

The main steps of a particle tranport code with processes is illustrated in Fig. 1. The central loop involves a stack for temporary storage, a geometric transport code, and a list of processes that can lead to secondary particle production or also absorption.

3.1 Computational efficiency

Computational efficiency is not optional for ngC. The efficient use of expensive large-scale resources is a crucial requirement, and must be considered from the early planning.

The use of external code and libraries must be minimized to the absolute minimum in order to stay conflict-free and operational over a very extended period of time. Individual exceptions might be possible, but must be well motivated and discussed before use. For each functionality it should be evaluated if a basic re-implementation is feasible. In any case, whenever possible, appropriate wrappers in ngC should hide the implementation details of external packages from the users inside ngC.

A computer language offering high level of design flexibility and at the same time excellent compiler and optimization support is required. It is an advantage to chose a language that also has non-science relevance and thus assures long-term support, development and expertise.

For this purpose C++ will be chosen. The start of ngC will be based on the dialect C++11/14/17, but this can be revised in the future.

Run-time dynamic design patterns like virtual classes or dynamic libraries should be avoided.

Data copy operations must be minimized, or performed as late as possible. The use of “lazy” functionality, which is executed only delayed when actually, needed should be promoted.

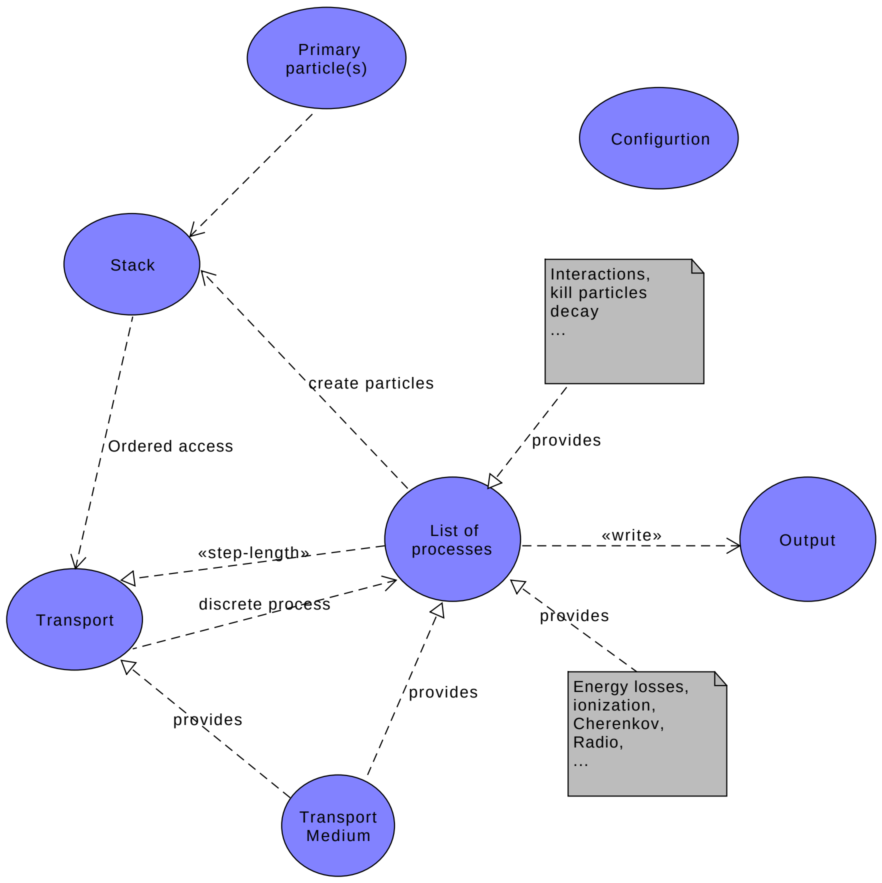

Figure 1: Scheme of particle transport code with processes.

Compiler and CPU optimization should be fully considered for ngC. Production versions of the code should take full benefit from all available optimizations. The execution of particular code on GPUs (or maybe even more custom hardware) must be transparently possible.

Parallel and multi-core computation are standard.

3.2 Related projects and previous work

ngC will heavily depend on expertise gained with the original CORSIKA program.

MCeq is a recent tool dedicated to the solution of cascade equation. It already offers GPU support and very high computational efficiency. CONEX is a cascade equation air shower simulation program that has been integrated in

CORSIKA and provides enourmous increase in computation speed.

dynstack is a recent extension of CORSIKA. Principle ideas of this should be placed inside the stack of ngC.

COAST has been developed in CORSIKA with the aim to offer plugin-opportunity for scientist. The functionality of COAST will be available in ngC.

3.3 Output

The output generation of ngC will be by far more flexible than in the current CORSIKA. The user will be able to decide for his specific case what kind of output is optimal, with the additional possibility to pipe the output directly into subsequent workflow steps without any need to store on disk.

There will be a new form of standard output file that is flexibile in content, and scales well from air showers with just a few particles on ground up to air showers with billion of particles on ground.

The old binary output format "DATxxxxxx" file can be one option but with all the known limitations.

HDF5 may be considered, but it has at least the disadvantage of being an external dependence.

File-system based solution with potentially *tar*ed or *zip*ed directory structures can offer similar performance and flexibility.

It might be interesting to investigate the advantage of compressed ASCII output versus pure binary output. Advantages are: human readability, complete architecture independence. Disadvantages can include: potential speed penalty, potential space penalty.

Since the output is very important, it might be worth to perform an open selection process considering a systematic benchmarking. To some part this has been done already in the past.

4 Tools and infrastructure

The basic development infrastructure for ngC will be provided at KIT. The most useful tool for collaborative development currently available is git, which allows to have a very dynamic and large base of contributors, and at the same time a well controlled access to the main code base via *pull requests*.

We will use a gitlab server for the hosting, the default choice for this is gitlab.kit.edu.

Unit test must yield a very high coverage of the ngC code. Unit tests are executed automatically by a jenkins (or equivalent) service on a regular basis.

Further validation and high-level functional tests will accompany the regular testing.

Automatic testing will provide a well defined list of supported environments, combined with a control over a specified set of different combinations of simulation options.

5 Main challenges

While there are many challenges, here is a list of topics that require particular dedicated attention. These topics are more-or-less directly linked to the underlying/internal physics of the cascading process and ask for very intelligent and likely highly complex solutions.

1. Efficient integration of electron-gamma cascades (previously EGS4)
2. Random-number generation in an inherent multi-core, parallel, CE environment.
3. Investigate the limits of equivalence between CE and detailed tranport (dE/dX, Cherenkov, lateral structure, radio production, ...)
4. GPU optimization.

6 Details

Taking these considerations into account a more detailed scheme of the simulation workflow can be produced. In Fig. 2 this is outlined. Many of the aspects shown in this diagram still need to be determined and optimized. Since the basic design is given here, alternative functionality and building blocks can be developed in parallel.

6.1 Main loop

A central part is the loop over all particles on the stack. Those particles are all transported, and processed. As part of this process also cascade equation tables can be filled. If the stack is empty (or maybe according to another trigger), the cascade equations are processed, which can once more also fill the particle stack. Thus, a double-loop is required here in order to process the full particle cascade.

```
 while (particleStack not empty) {
  while (particleStack not empty) {
   particle = particleStack.get();
   transport(particle);
  }
  cascade_equations_solve();
}
```

The transport function needs to handle geometric transport of neutral and charged particles, thus, magnetic and electric deflections are implemented. The distance of tracking steps is limited by the type of processes considered. The highest cross section process limits the maximum step size, the that is simulated is randomly selected accorind to its probability. For each step, continuous processes are calculated, which can be numerous: ionization, Cherenkov photons, radio emissions, etc. There is potentially a huge (and most straightforward) potential for GPU optimization is continous processes.

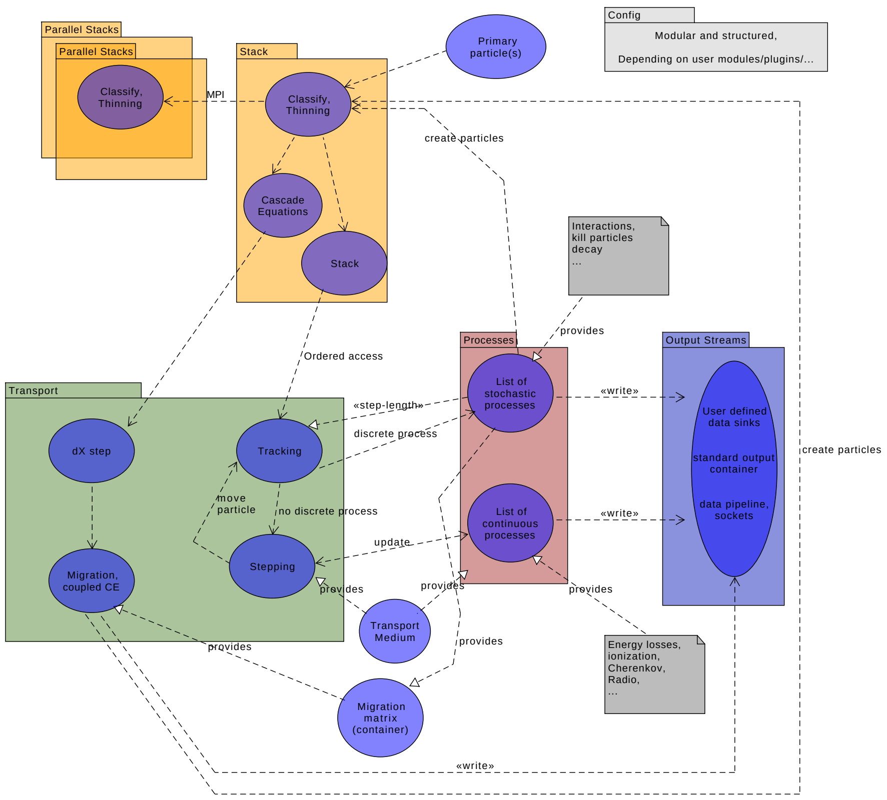

Figure 2: Main building blocks and workflow steps of ngC already highlighting the fundamental functionality and flexibility.

```
transport(particle) {
  while (particle.exists()) {
    step = next_step_length();
for (continuousProcesses cp){
      cp.update(particle, step);
    }
    particle.move(step);
    if (stochasticProcess(step)) {
      stochasticProcess(particle);
    }
  }
}
```

The corresponding cascade equations soltuion is performed as a fully equivalent functional block to normal tracking. A part of the processes is simplied to migration matrices, however, it is the aim to use the observed migration in energy, location and particle types also for the production of: ionization, Cherenkov light, radio emission, etc. The physical limits of the quality of this must be evaluated later. Processes like ionization are straightforward and must be handled in particular to record the longitudinal shower development, howerver, processes that depend a lot on geometry, like Cherenkov or radio emission, can be just an approximation. But it is up to detailed studied to evaluate this and adapt it to potential use cases.

```
cascade_equations_solve() {
  while (table.not_empty()) {
tableNew = (StochasticAndMove) * table;
    for (continuousProcesses cp) {
      cp.update(table, step);
    }
    if (tableNew.contains_entries_above_CE_threshold()) {
      create_particles_on_stack()
      remove_from_tableNew()
    }
    particle_output()
    particle_regenerate() // end of CE
    table = tableNew
  }
}
```

6.2 Atmosphere and environment and geometry

The basic geometry of ngC is spherical.

The center of the geometry for ngC is related to the most convenient access to environmental data. Since the basic geometry of

Tradidionally the medium of transport for CORSIKA was the atmosphere. It is one purpose of ngC to allow much more flexible combination of environments. This includes water, ice, mountains, the moon, etc. In this case also the interface between different media becomes a matter of significance.

The environment must at least provide

Interfaces between media can provide data about both adjacent media, e.g. refractive indices, densities, etc. Processes can use this information.

6.3 Particle data and stacks

There will be no dedicated class describing an individual particle. Instead particles are generally represented by a reference/proxy to data on a stack. Where stacks can be a fortran common block, dynamic C++ data, a file-based cache, or any other source/storage of particle data.

7 Summary

The steps towards ngC outlined here are crtical to ensure the future of scientific research in fields, where the simulation of particle transport cascades with stochastic and continuous processes are involved. The reach of the resulting framework will be far beyond the original CORSIKA program. It is up to the scientific community to decide in which concrete applications ngC will be used in the future. It is the aim to offer long-term support for the ngC program over a period of at least 20 years.

A much better access on the air shower physics simulation process will be one of the keys to resolve the main questions about cosmic ray physics, the universe at the highest energies, and related scientific questions.