cherenkov telescope array

Vectorization libraries

Vectorization techniques and libs

- Intrinsics
  - Low level SIMD instructions specific to each architecture
  - For arithmetics operations : +, -, /, x plus other types of operations (e.g. mask)
  - High performances but low portability
- Intrinsics through Vectorization libraries
  - Provide an abstraction of low level intrinsics
  - Intermediate performances and good portability
- Auto-vectorization (by compiler)
  - Lower performances and high portability
  - No extra-dependancy
- Optimized libs
  - E.g. For Linear Algebra, elementary functions, etc.
- Code generators: e.g. TensorFlow, Loopy (from Python language)
  - Transform programs written in C++, C, Python in highly optimized C++, CUDA code for a given architecture
  - Pros: Good perf and good portability
  - Cons: Add an extra dependancy

-> Vectorization libs and optimized libs seem the good approach for Corsika 8

Vectorization libs

- Vc
  - https://github.com/VcDevel/Vc
  - SSE4, AVX, AVX-2
- UME::SIMD used as backend of VecCore -> Tested
  - https://github.com/edanor/umesimd
  - SSE4, AVX, AVX-2, AVX-512
  - Support of some vectorized mathematical functions but not really optimized
- xsimd
  - https://github.com/QuantStack/xsimd
  - SSE4, AVX, AVX-2, AVX-512
  - Support of some vectorized mathematical functions
- VecCore (CERN)
  - https://github.com/root-project/veccore
  - Part the ROOT project on GitHub
  - Vc, UME::SIMD, or a scalar implementation

Libs for elementary functions

- **Implement the most common functions**
  - **exp, log, trigonometric functions, cbrt, ...**

- Intel's SVML:
  - https://software.intel.com/en-us/node/583201
- AMDs libm:
  - http://developer.amd.com/tools-and-sdks/archive/libm
- GNU's libmvec (open source) **-> Tested**
  - gcc > 4.9.0 and glibc > 2.26
  - Enabled by ffast-math flag -> No IEEE compliant
  - https://sourceware.org/glibc/wiki/
  - Based on auto-vectorization
  - No code transformation needed but it works only for simple scenarios

Libs for elementary functions

- CERN's VDT (backend of VecCore) (open source) **-> Tested**
  - https://github.com/dpiparo/vdt
  - http://iopscience.iop.org/article/10.1088/1742-6596/513/5/052027/pdf
  - Based on auto-vectorization
  - Functions require vector parameters
  - About 3-4 ulps accuracy
- SIMD vector libm (open source) **-> Tested and Used for Corsika 7 optimization**
  - https://gitlab.com/cquirin/vector-libm
  - https://hal.archives-ouvertes.fr/hal-01511131/document
  - Based on auto-vectorization
  - Functions require vector parameters
  - About 3 ulps measured accuracy (8 ulps guaranteed)

Libs for elementary functions

- VecMath (CERN)
  - https://github.com/root-project/vecmath
  - Elementary functions but also pseudorandom number generators, support of specific types, such as Lorentz vectors
  - Being extended to support vector operations for 2D and 3D vectors, and general-purpose vectorized algorithms
  - SIMD and SIMT (GPUs) support based on VecCore
  - Functions require vector parameters
  - Free of of external dependencies other than VecCore and usable by vector-aware software stacks

Libs for Linear Algebra

- Linear Algebra
  - Low-level libs: e.g. MKL, BLAS, ATLAS, LAPACK, LAPACK++
    - Pros: High perf. and portable
    - Cons: Not easy to use because of the high number of parameters for the functions
  - High-level libs (based on auto-vectorization): e.g. Eigen, Armadillo, HPX, Xtensor
    - Pros: Good perf and Easy to use. Allow to handle: Vector, Matrix, Tensor
    - Cons: They are specialized for different use cases but they are not compatible among them
      - Armadillo -> good for small matrix
      - Eigen -> good for large matrix
      - HPX -> good for very large matrix
    - Cons: Compilation can be very long and high memory consuming