Vectorization Plans in ROOT

L. Moneta

Vectorization in ROOT Math

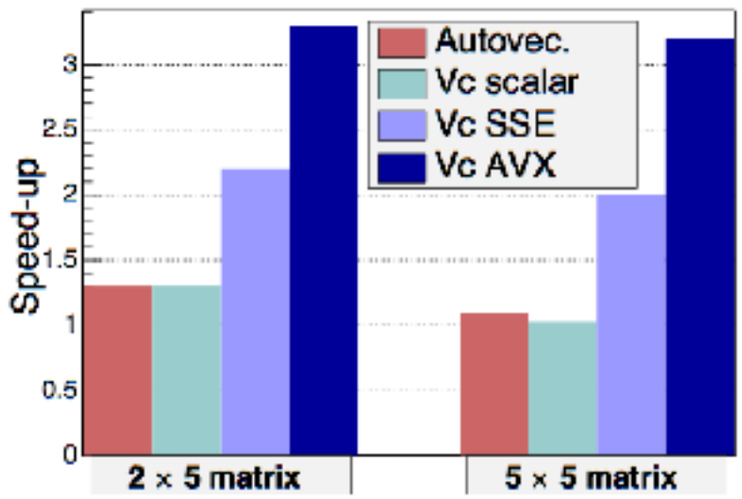
- ROOT depends now on VecCore.
  - Special types defined on the vector backends of VecCore (e.g. **Double_v**)
- VecCore types can be used for evaluating functions
- TF1 objects can be built by passing vectorised user functions built using VecCore
  - **TF1 f("f",[](ROOT::Double_v * x, double * p){ return sin(x[0]); },0,10,0)**
- TFormula function expressions can be automatically vectorised using VecCore when JIT'ing
  - **TF1 f("f","sin(x)");**
- ROOT provides a set of internal function interfaces which are used by numerical algorithms (e.g. minimisation and fitting)
  - These interfaces have been extended to support the template types of VecCore
- Fitting in ROOT is now vectorized
  - The likelihood evaluation can be performed using a vectorized model function

Future Plans for VecCore Usage

- Extend vectorisation to additional numerical algorithms
  - e.g. numerical integration and differentiation
- Re-implement most used mathematical and statistical functions of Math (e.g. TMath::Gaus) as templated functions which can be instantiated using the VecCore types
- The basic math functions (e.g. trigonometric and hyperbolic functions) could be implemented directly in VecCore
  - e.g. TMath::Sin(Double_v x) will be just a wrapper to VecCore
- Use new vectorised functions in critical code part of ROOT
  - e.g. activation functions in neural networks
  - probability density functions for fitting

Matrix and Vector Libraries

- ROOT provides a template vector and matrix classes (optimized for small sizes) which can be used in single and double precision
  - **SVector< double,N>**
  - **SMatrix< double,N,M>**
- Template classes for geometry and physics vectors with their transformations
  - **DisplacementVector3D< Cartesian3D<double>**
  - **LorentzVector<PxPyPzE4D<double> >**
- VecCore types (ROOT::Double_v) can be used as template parameters for vector and matrices classes
  - vectorisation for operations on a list of vectors/matrices (vertical vectorisation)
- Missing vectorisation for boost and rotation classes
- Plan to investigate using horizontal vectorisation for complex matrix operations
  - e.g. similarity $\mathbf{A\,B\,A}^t$
    implemented by LHCb using A. Fog library
    (*see presentation from G. Raven*)



Random Numbers

- New random classes added recently in ROOT
  - New interface wrapping also generators from std::random and gsl_random
- Added several types of MIXMAX random engines
- Plan to include a vectorised version of main generators
  - Either implement directly in ROOT or in a separate library (e.g. VecCore) and just wrap in the ROOT interface
- Provide vertical (external) vectorisation and eventually horizontal (internal) whenever possible
- Work in collaboration with Simulation team for having a common library

Outlook

- Desirable to develop tools for vectorisation in common
- Ideally developments should be provided in a independent library (e.g. VecCore)
  - Prefer to avoid having several of them in order to keep simple configurations
  - VecCore could be also integrated in ROOT
    - we already provide library in ROOT standalone library which can be built and use independently (e.g. Minuit)
    - other Math libraries of ROOT (SMatrix, GenVector) could be used already now as independent packages
- Problem with the distribution of vectorised libraries
  - we would like to select at run time the optimal vectorised Math library depending on the running architecture