

Vectorisation and GPUs extensions of ROOT::Math routines

Anca – Mihaela POPESCU
CERN openlab Summer Student 2015

Supervisors:
Lorenzo MONETA
Danilo PIPARO

26/08/2015

Background image: Shutterstock



The Project Challenge

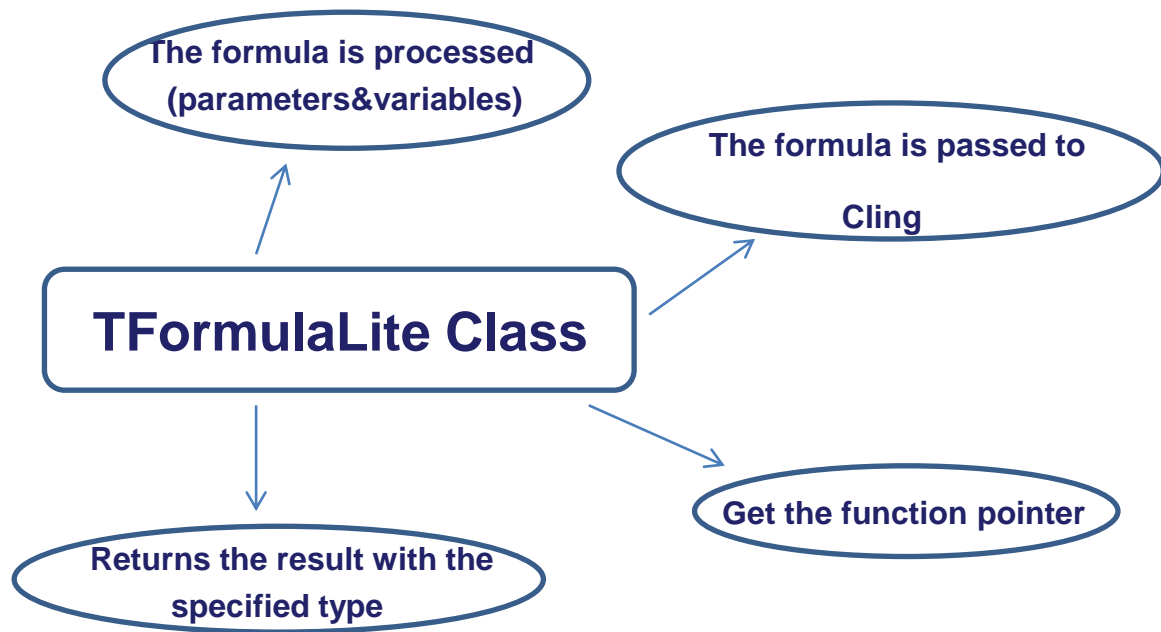
Project purpose: an accelerator/gpu ready implementation of the mathematical functions and statistical distributions offered by the ROOT::Math namespace

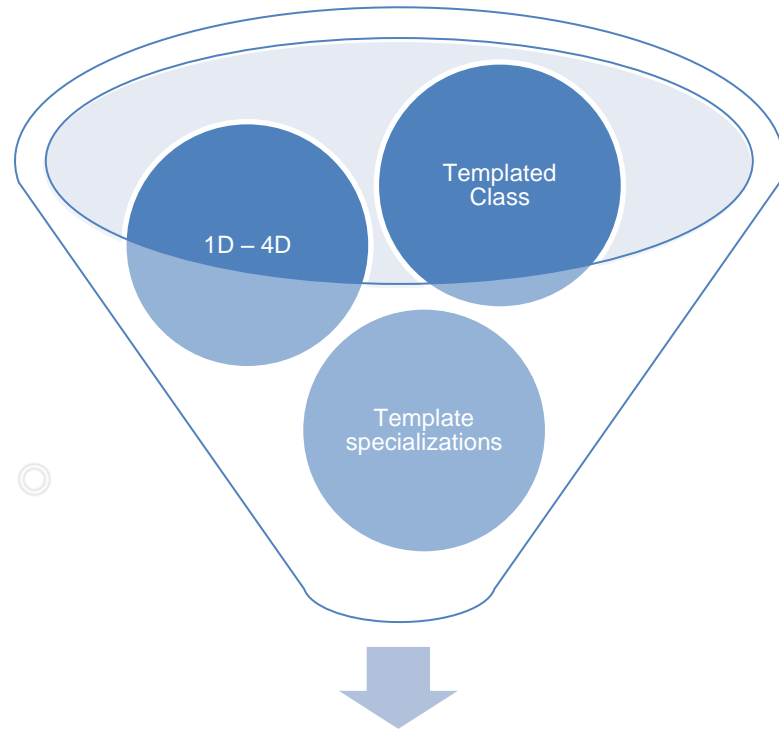
TFormula Class

- Class for evaluating mathematical functions provided as expression strings
- The created function is compiled using the JIT of Cling
- JIT compilation is done at initialization time
- Computes double values

=> necessity of a new evaluating method which computes vector types

An intermediate solution...





Eval() method



The main difference between TFormulaLite Class and TFormula Class



The role of computing any type of variable

Specialised Eval() (1)

Nested
Template
Class

```
template<class T>
class TFormulaLite
{
    template <class T2>
    class Evaluator
    {
        ...
    };
};
```

Static generic
Eval()
methodes

```
static T2 DoEval(const T2& x, TFormulaLite<T2> &fct)
{
    ...
    auto f = (T2 (*)(const T2&))fct.fcnptr;
    return f(x);
}
```

Template
specializations

➤ **double Eval()**

Specialised Eval() (2)

```
template <>  
template <class T2> class TFormulaLite<double>::Evaluator { ... };
```

➤ **float Eval()**

```
template <>  
template <class T2> class TFormulaLite<float>::Evaluator { ... };
```

➤ **Vc::double_v Eval()**

```
template <>  
template <class T2> class TFormulaLite<Vc::double_v>::Evaluator { ... };
```

➤ **Vc::float_v Eval()**

```
template <>  
template <class T2> class TFormulaLite<Vc::float_v>::Evaluator { ... };
```

➤ **std::vector<>**

```
template <>  
template <class T2> class TFormulaLite<std::vector<double>>::Evaluator { ... };
```



```
T f1(const T& x) {return sin(x)/x+x*x*x-2*x+cos(2*x);}
```

TFormula Eval()

Real time 0:00:10, CP time 10.130

TFormulaLite Eval() – double specialization

Real time 0:00:07, CP time 7.530

TFormulaLite Eval() – float specialization

Real time 0:00:03, CP time 3.360

TFormulaLite Eval() – Vc::double_v specialization

Real time 0:00:14, CP time 14.190

TFormulaLite Eval() – Vc::float_v specialization

Real time 0:00:04, CP time 4.590

TFormulaLite Eval() – std::vector<double> specialization


Real time 0:00:11, CP time 11.180

TFormulaLite Eval() – std::vector<float> specialization

Real time 0:00:07, CP time 7.240

Tests

The conclusion

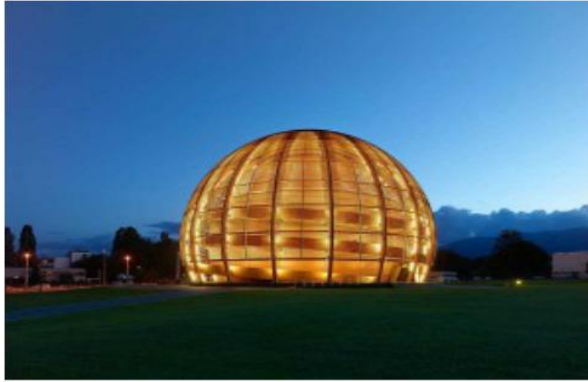


PODs & every other type
Vc library
Vectorised formula in Cling
Multidimensional functions

Pass the type of result when creating the formula
Problems compiling with Cling with maximum optimization

Future work

- X Analyze the Vc results
- X Integration in TFormula Class
- X Code optimization



THANK YOU!