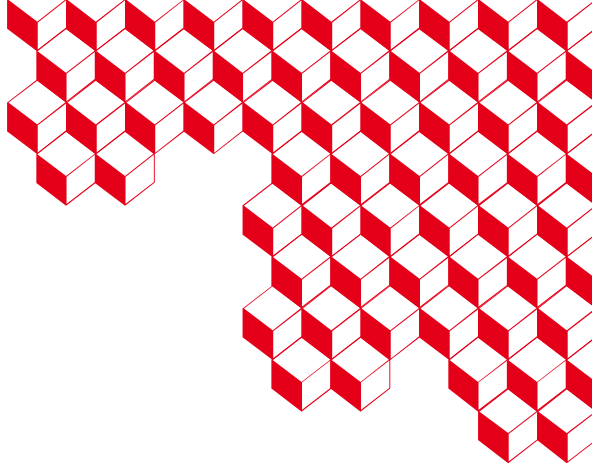


Direct Measurement of Numerical Accuracy

Cédric Chevalier, CEA, DAM, DIF

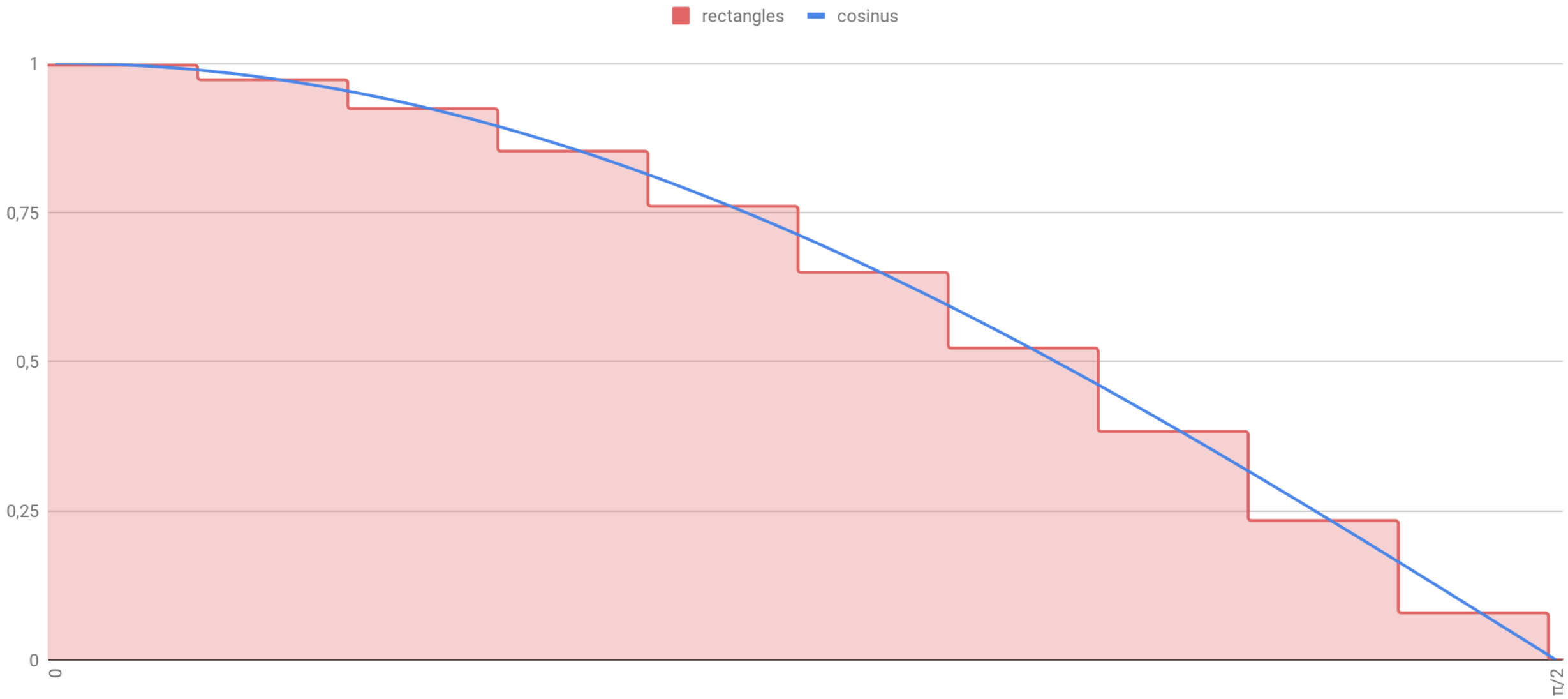
Floating point computations and correctness

- Most of Kokkos applications performs **floating point computations**
- And they care about their results

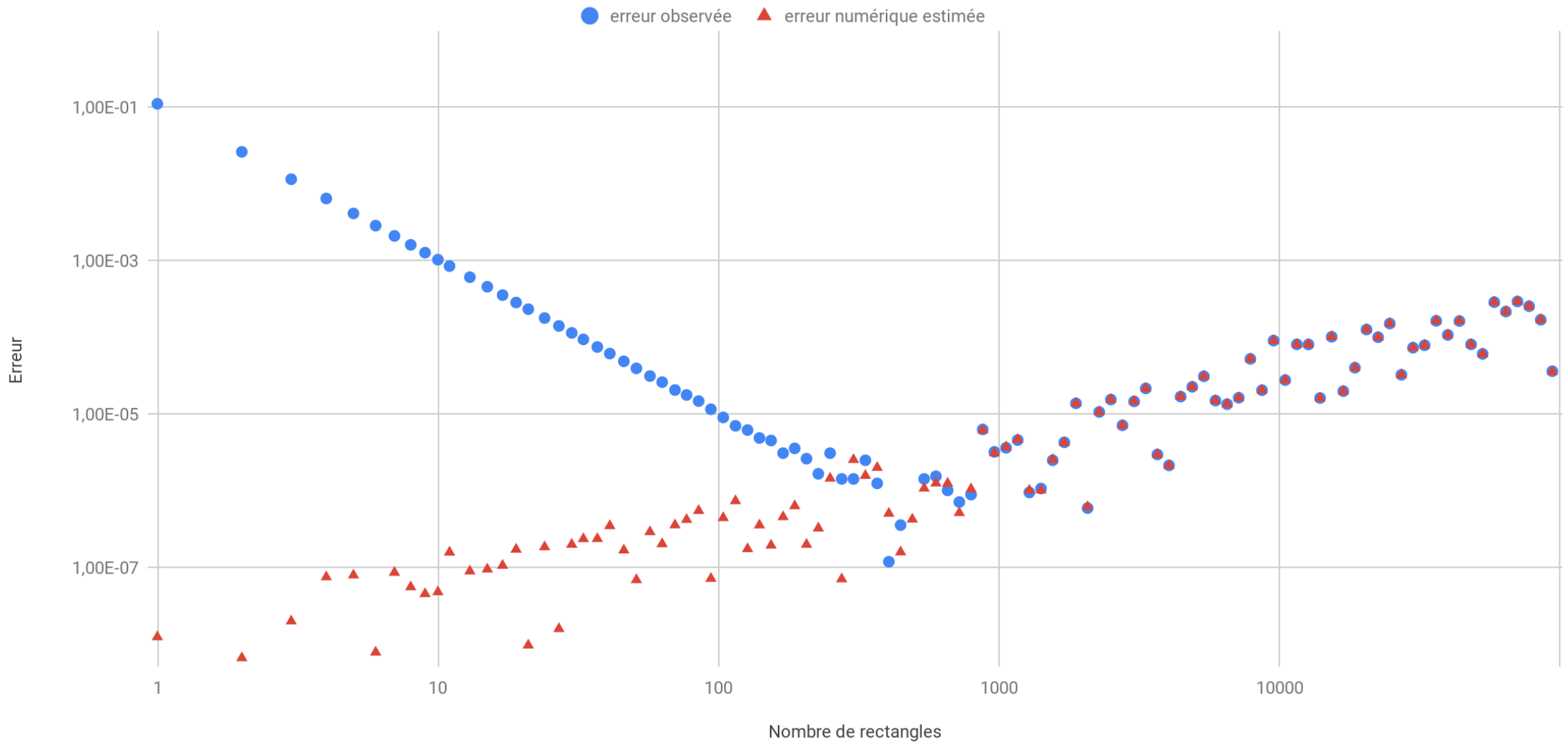
- Several related questions
  - How to unit tests numerical applications if outputs are allowed to change
    - New hardware
    - New software
    - Non deterministic computations
  - What floating point representation is the most efficient for my use case (float vs double vs AI)

- We (the CExA project) are currently porting Shaman (https://gitlab.com/numerical_shaman/shaman), a numerical profiler to Kokkos ecosystem

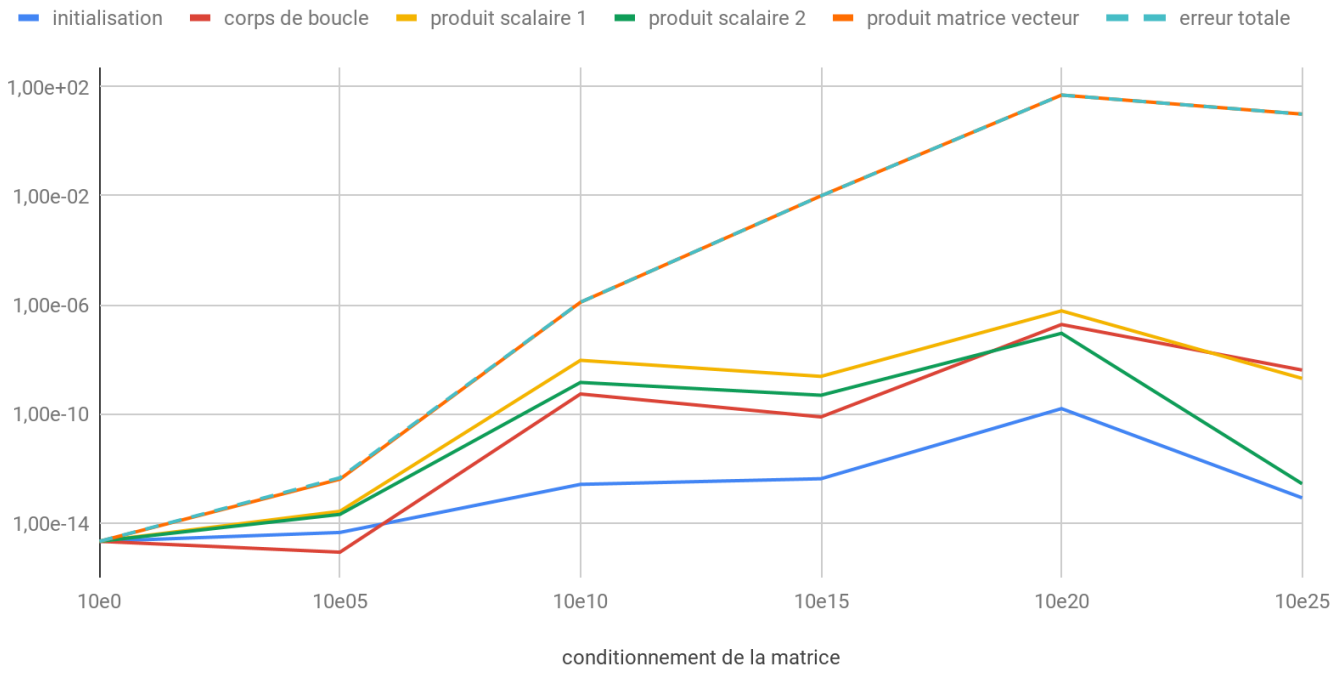
Numerical Integration: Rectangle Method



How accurate are we ?



CG Convergence vs Rounding Errors



Classical Algorithm (Trilinos/Belos):

Error distribution: [matrix_vector_product:99%…]

Iterations: **5000** Residual: **3400 Not converged**

Compensated matrix-vector product Error distribution: [dot_product1:94%…]

Iterations: **1494** Residual: **0.8**
« Converged »

The Idea Behind Shaman

We introduce a new datatype that stores the original value and operation errors

Errors are propagated and computed using **Error Free Transforms**

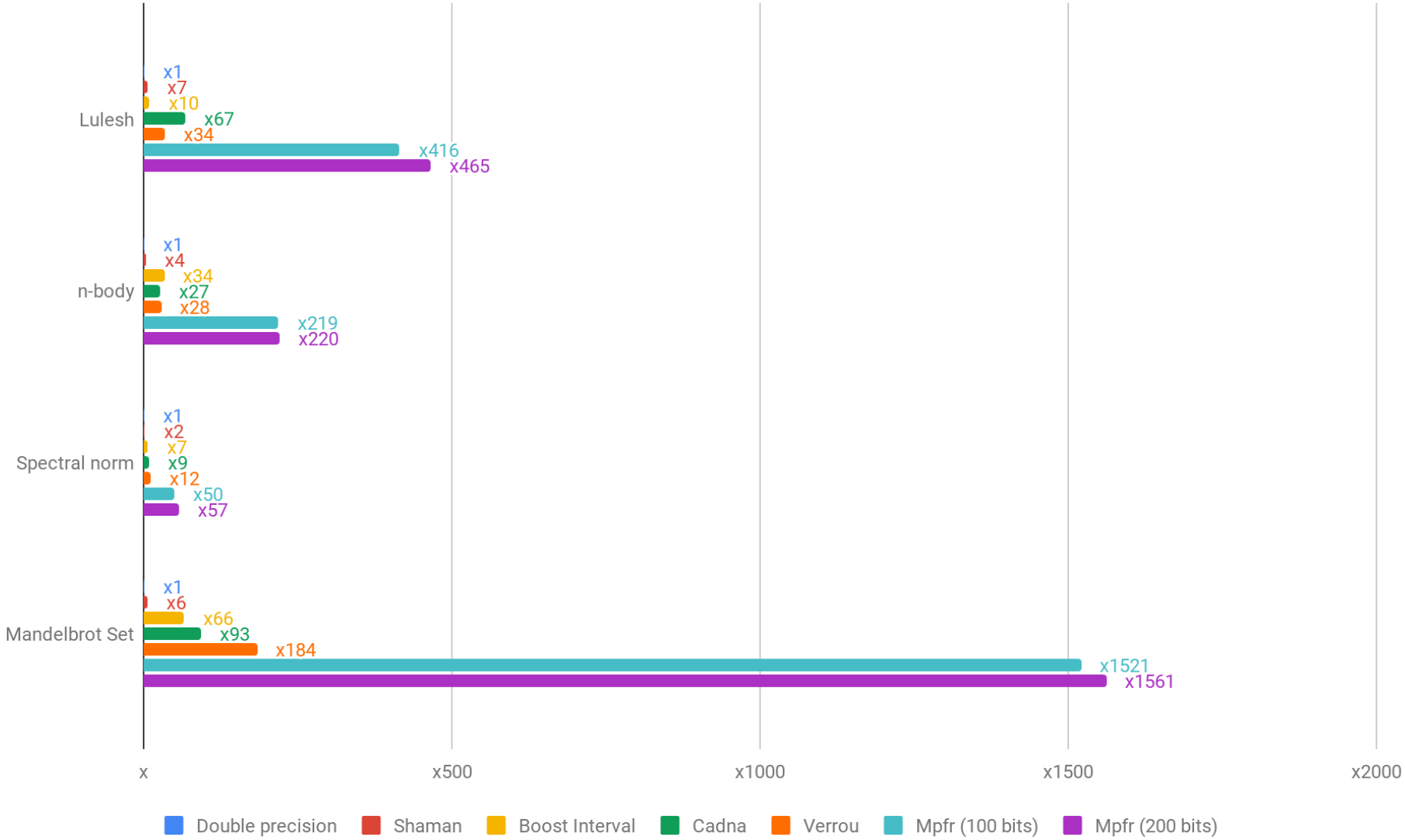
```
Snum operator+(Snum n1, Snum n2)
{
   numberType result = n1.number + n2.number;
   numberType errorOperator = EFT::TwoSum(n1.number, n2.number);
   errorType newError = errorOperator + (n1.error + n2.error);

   return Snum(result, newError);
};
```

```
double TwoSum(double x, double y)
{
      double result = x + y;
      double y2 = result - x;
      double x2 = result - y2;
      double epsilon1 = y - y2;   // ≠ 0
      double epsilon2 = x - x2;   // ≠ 0
      double error = epsilon1 + epsilon2;
      return error;
};
```

Main expected issue when going to Kokkos:

how to ensure that compilers will not optimize out useful computations.

CPU Performance



Thank you !

CEA, DAM, DIF
Arpajon
France

Phone. + 33 1 69 26 40 00