Performance benchmark: ROOT vectors and matrices vs Eigen3

- CERN
- any other collider experiment

Eigen

- TensorFlow
- OpenCV
- Open Babel

Daniil Yakovlev, 28.06.19

Why Eigen?

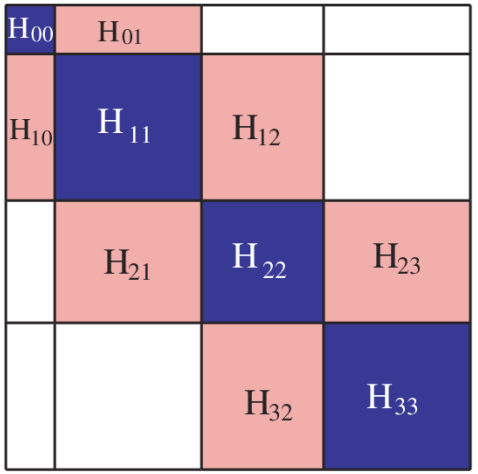
- Flexible
- Easy to use
- Full docs
- Faster (!)

```
#include <Eigen/Dense>

Matrix3f m;
m << 1, 2, 3,
     4, 5, 6,
     7, 8, 9;
std::cout << m;
```

```
Output:
1 2 3
4 5 6
7 8 9
```

- Block operations

```cpp
#include <Eigen/Dense>
#include <iostream>
using namespace std;
int main()
{
  Eigen::MatrixXf m(4,4);
  m <<  1, 2, 3, 4,
        5, 6, 7, 8,
        9,10,11,12,
       13,14,15,16;
  cout << "Block in the middle" << endl;
  cout << m.block<2,2>(1,1) << endl << endl;
  for (int i = 1; i <= 3; ++i)
  {
    cout << "Block of size " << i << "x" << i << endl;
    cout << m.block(0,0,i,i) << endl << endl;
  }
}
```

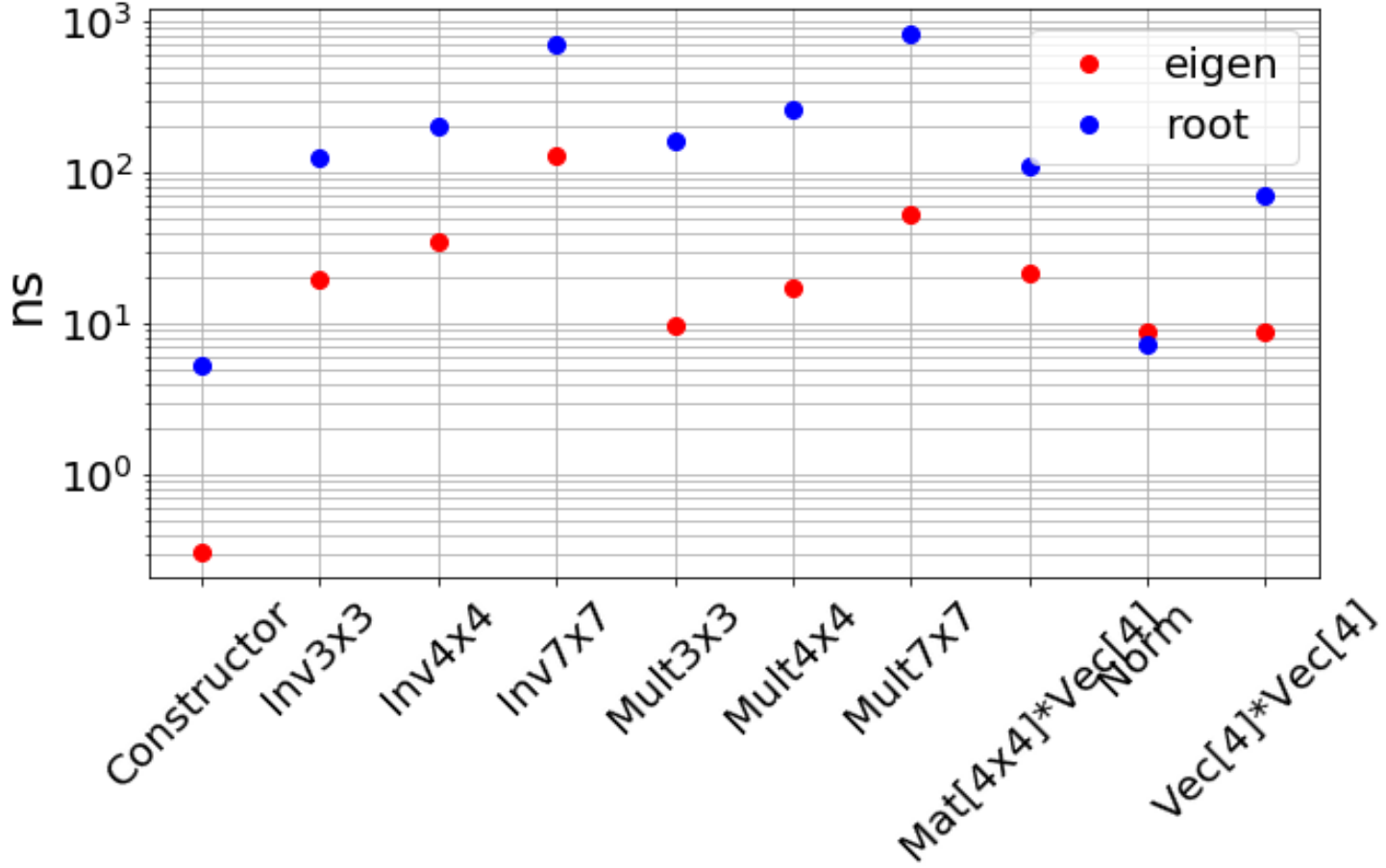


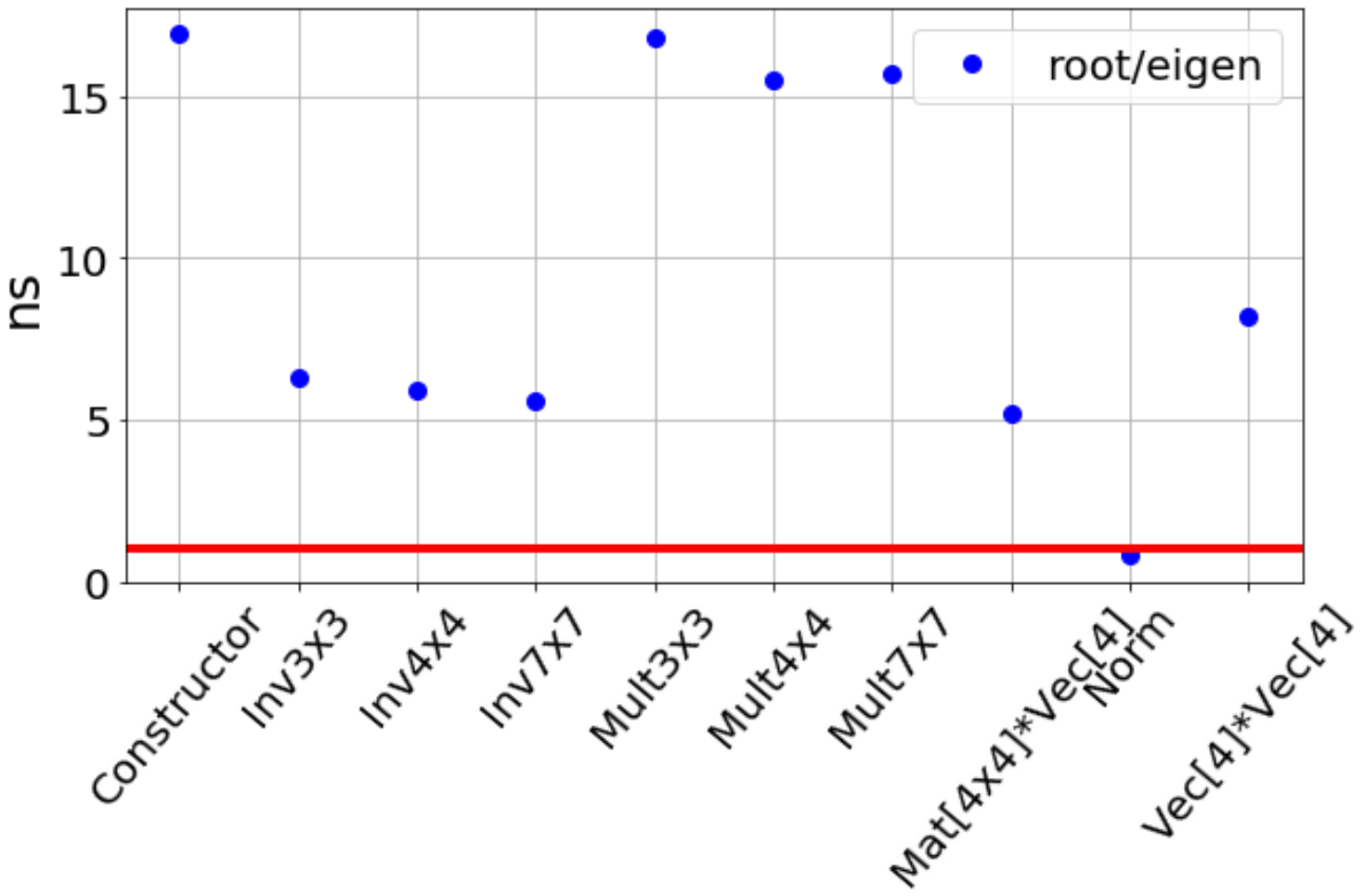

```
Output:
Block in the middle
 6  7
10 11

Block of size 1x1
1

Block of size 2x2
1 2
5 6

Block of size 3x3
 1  2  3
 5  6  7
 9 10 11
```

Google Benchmark [1]

```
1  #include <benchmark/benchmark.h>
2
3  // Define benchmark
4  static void BM_SomeFoo(benchmark::State& state) {
5      for (auto _ : state) {
6          foo(); // performance measured code
7      }
8  }
9  // Register the function as a benchmark
10 BENCHMARK(BM_StringCreation);
11
12 BENCHMARK_MAIN();
```

[1] GitHub Repository, https://github.com/google/benchmark


10^3
10^2
10^1
10^0
ns
eigen
root
Constructor
Inv3x3
Inv4x4
Inv7x7
Mult3x3
Mult4x4
Mult7x7
Mat[4x4]*Vec[4]
Nbrm
Vec[4]*Vec[4]


root/eigen
ns
0
5
10
15
Constructor
Inv3x3
Inv4x4
Inv7x7
Mult3x3
Mult4x4
Mult7x7
Mat[4x4]*Vec[4]
Norm
Vec[4]*Vec[4]