# Update on Matrix Algebra

Beomki Yeo

# Math/Backends of Algebra-plugins

|  |  | Backends |  |  |
|---|---|---|---|---|
|  |  | CPU | CUDA | SYCL |
| Math | cmath (home-made) | 🟩 | 🟩 | 🟩 |
|  | Eigen | 🟩 | 🟩 | 🟨 |
|  | SMatrix | 🟩 | ⬜ | ⬜ |
|  | VC | 🟩 | ⬜ | ⬜ |

🟩 Backend is supported

🟨 Supported but not tested

⬜ Backend is NOT supported

Devloped algebras before v0.8.0:

- ○ vector algebra
- ○ local ↔ global transform
- ○ matrix generation and element access
  - • matrix algebra had been missing

# Required Matrix Operations

- Matrix Creation
  - Zero
  - Identity
- Addition and Subtraction
- Multiplication
  - Normal multiplication
  - Blocked multiplication
- Transpose
- Inverse
  - 2x2 (KF updater)
  - 4x4 (local ↔ global transform)
  - 6x6 (KF smoother)
  - Maybe more?

# Eigen and SMatrix implementation

- Writing the matrix algebra for Eigen and SMatrix is just porting the existing functions.

```
/// "Matrix actor", assuming an Eigen matrix
template <typename scalar_t>
struct actor {

  /// 2D matrix type
  template <int ROWS, int COLS>
  using matrix_type = Eigen::Matrix<scalar_t, ROWS, COLS>;

  // Create transpose matrix
  template <int ROWS, int COLS>
  ALGEBRA_HOST_DEVICE inline matrix_type<COLS, ROWS>
transpose(
      const matrix_type<ROWS, COLS> &m) {
    return m.transpose();
  }

  // Create inverse matrix
  template <int N>
  ALGEBRA_HOST_DEVICE inline matrix_type<N, N> inverse(
      const matrix_type<N, N> &m) {
    return m.inverse();
  }
};
```

# cmath Implementation

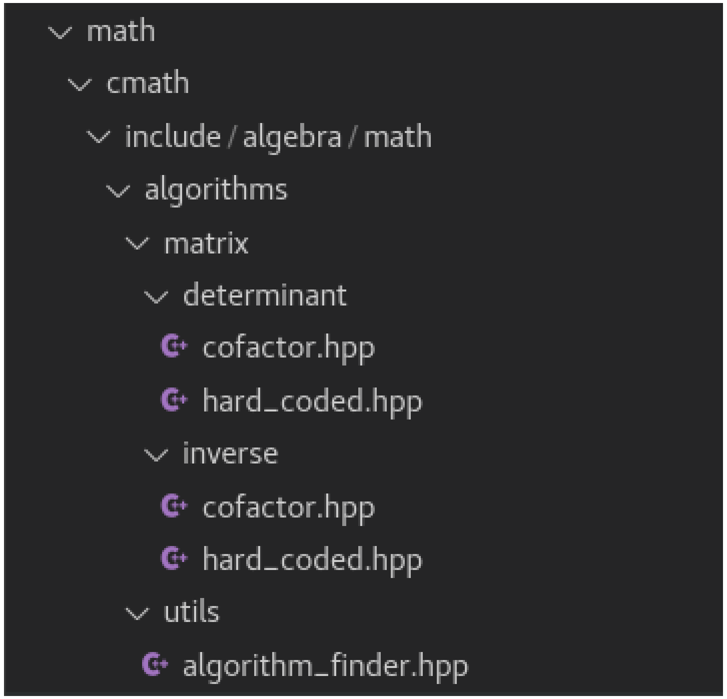
- For cmath, we can try something better
  - User can decide which specific algorithm will be used for which matrix dimensions in compile time
  - Various matrix_actor can be defined to test different aggregations

```
// Define inverse algorithm
// Base algorithm is cofactor method
// For 2x2 and 4x4 matrix, hard coded method is used
// For 3x3 and 5x5 matrix, LU decomposition is used
using inverse_actor = matrix::inverse::actor<cofactor<>,
                        hard_coded<2,4>, LU_decomposition<3,5>>

// Define matrix actor
using matrix_actor = matrix::actor<scalar, inverse_actor>

matrix<2,2> m22;
matrix<3,3> m33;
matrix<7,7> m77;

m22_inv = matrix_actor().inverse(m22) // hard-coded
m33_inv = matrix_actor().inverse(m33) // LU decomposition
m77_inv = matrix_actor().inverse(m77) // cofactor
```

# Customizable algorithms in cmath matrix

- Determinant
  - cofactor
  - hard_coded for 2x2 and 4x4

- Inverse
  - cofactor
  - hard_coded for 2x2 and 4x4



```
math
  cmath
    include / algebra / math
      algorithms
        matrix
          determinant
            cofactor.hpp
            hard_coded.hpp
          inverse
            cofactor.hpp
            hard_coded.hpp
        utils
          algorithm_finder.hpp
```

# Outlooks

- Similar concept of algorithm aggregation can be applied to Eigen and SMatrix implementation

- cmath needs 6x6 hard coded inversion or LU decomposition
  - cofactor is slow!

- cmath matrix multiplication relies on * operator which does standard multiplication
  - Needs to add customizable actor for multiplication as done for inverse and determinant