# The Eigen C++ template library for linear algebra

April 8, 2016

# Eigen

## Introduction to

Eigen 3

## Useful web links

- `http://eigen.tuxfamily.org`

# Eigen

## Eigen

The Eigen library is a linear algebra library that works with dense and sparse matrices. With the use of advanced programming techniques it performs algebraic operations with very high efficiency. It provides algorithms for the solution of linear systems, factorizations, and also interfaces to external libraries that perform linear algebra operations.

It is widely used as a library in many applications, i.e. at Google.

# Eigen

## Introduction

```
#include <iostream>
#include <Eigen/Dense>
int main()
{
 Eigen::MatrixXd m(2,2);
 m(0,0) = 3;
 m(1,0) = 2.5;
 m(0,1) = -1;
 m(1,1) = m(1,0) + m(0,1);
 std::cout << m << std::endl;
}
```

# Eigen

## Matricies

Generic matrix

```
Matrix<typename Scalar, int RowsAtCompileTime,
                        int ColsAtCompileTime>
```

## Special cases

- Matrix with fixed dimensions

  ```
  typedef Matrix<float, 4, 4> Matrix4f;
  ```

- Matrix with non-fixed dimensions

  ```
  typedef Matrix<double, Dynamic, Dynamic> MatrixXd;
  ```

- Matrix with only one fixed dimension

  ```
  typedef Matrix<int, Dynamic, 1> VectorXi;
  ```

# Eigen

## Initialization

- non-initialized matrix

```
MatrixXd m(2,2);
```

- initialized matrix

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

- initialized matrix

```
MatrixXf m = MatrixXf::Constant(3, 3, 1.2);
```

# Eigen

## Arithmetics

Eigen implements all the arithmetic operators, and also

- transpose `a.transpose()`
- conjugate `a.conjugate()`
- adjoint `a.adjoint()`
- dot product `a.dot(b)`
- cross product `a.cross(b)`
- ...

# Eigen

## Expression Templates

the code

```
VectorXf a(50), b(50), c(50), d(50);
...
a = 3*b + 4*c + 5*d;
```

is implemented in such a way that it is equivalent to the execution of

```
for(int i = 0; i < 50; ++i)
  a[i] = 3*b[i] + 4*c[i] + 5*d[i];
```

just like the most performant (and optimizable by the compiler) c code.

# Eigen

## Advantages & disadvantages

the efficiency of the code greatly improves, but expression of the type

```
a = a.transpose();
```

are not allowed. Furthermore, compile time errors and execution time errors become mostly unreadable.

# Eigen

## code

```
#include <iostream>
#include <Eigen/Dense>
int main() {
  Eigen::Matrix3f m = Eigen::Matrix3f::Random();
  std::cout << "m␣=" << std::endl << m ; }
```

## compilation

Since Eigen is a pure header file library, we need to add its path in the compilation phase with the parameter `-I`

```
g++ -I/path/to/eigen/ my_program.cpp -o my_program -O2
```

in the virtual machine eigen is available as a module

# Exercises

- Re-implement the fem1d code using Eigen/Dense