

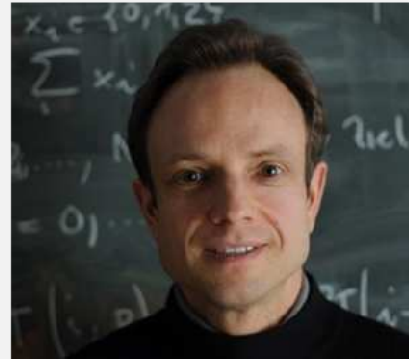
ETH Lecture 401-0663-00L Numerical Methods for CSE

Course Video Section 1.2.1: EIGEN

Prof. R. Hiptmair, SAM, ETH Zurich

Date: August 24, 2020

(C) Seminar für Angewandte Mathematik, ETH Zürich



Prerequisites.

- Basics of C++
- Classes and methods in C++
- Templates in C++

Dependencies. None

Duration: minutes

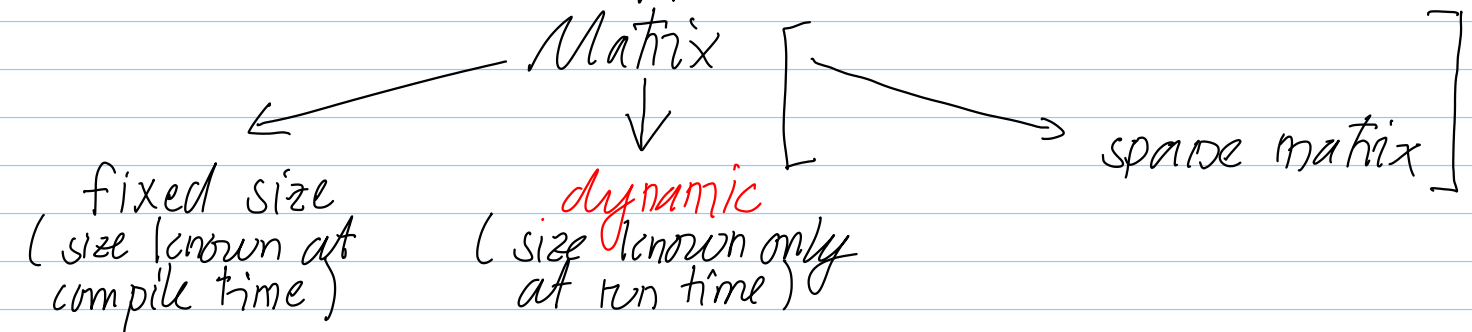
I. Computing with Matrices and Vectors

1.2. Software and Libraries

1.2.1. Eigen

↳ Header-only C++ template library
for numerical LA

Fundamental data type :



2

C++ code 1.2.1.2: Vector type and their use in EIGEN → GITLAB

```

1 #include <Eigen/Dense >
2
3 template<typename Scalar>
4 void eigenTypeDemo(unsigned int dim)
5 {
6     // General dynamic (variable size) matrices
7     using dynMat_t = Eigen::Matrix<Scalar, Eigen::Dynamic, Eigen::Dynamic>;
8     // Dynamic (variable size) column vectors
9     using dynColVec_t = Eigen::Matrix<Scalar, Eigen::Dynamic, 1>; → column vector
10    // Dynamic (variable size) row vectors
11    using dynRowVec_t = Eigen::Matrix<Scalar, 1, Eigen::Dynamic>; → row vector
12    using index_t = typename dynMat_t::Index;
13    using entry_t = typename dynMat_t::Scalar;
14
15    // Declare vectors of size 'dim'; not yet initialized
16    dynColVec_t colvec(dim);
17    dynRowVec_t rowvec(dim);
18    // Initialisation through component access ← accessing a vector comp.
19    for(index_t i=0; i < colvec.size(); ++i) colvec[i] = (Scalar)i;
20    for(index_t i=0; i < rowvec.size(); ++i) rowvec[i] = (Scalar)1/(i+1);
21    colvec[0] = (Scalar)3.14; rowvec[dim-1] = (Scalar)2.718;
22    // Form tensor product, a matrix, see Section 1.3.1
23    dynMat_t vecprod = colvec*rowvec; → matrix product
24    const int nrows = vecprod.rows();
25    const int ncols = vecprod.cols();
26 }

```

Line 23 :

$$\begin{bmatrix} \vdots \\ \vdots \end{bmatrix} * \begin{bmatrix} \vdots & \vdots \end{bmatrix} = \begin{bmatrix} \vdots \\ \vdots \end{bmatrix}$$

tensor product of vector

§ 1.2.1.5 Accessing sub-matrix

$M.\text{block}(i,j,p,q)$ method for matrix types
 $\rightarrow |M|_{i+1:i+p, j+1:j+q}$
 ! Eigen uses indexing from 0

C++ code 1.2.1.6: Demonstration code for access to matrix blocks in EIGEN → GITLAB

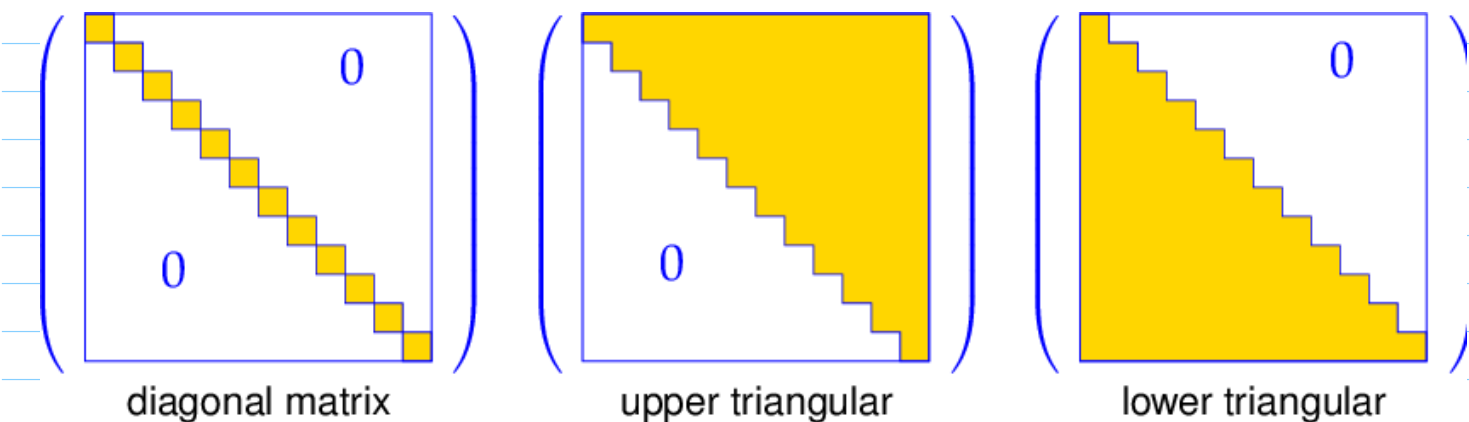
```

2 template<typename MatType>
3 void blockAccess(Eigen::MatrixBase<MatType> &M)
4 {
5     using index_t = typename Eigen::MatrixBase<MatType>::Index;
6     using entry_t = typename Eigen::MatrixBase<MatType>::Scalar;
7     const index_t nrows(M.rows()); // No. of rows
8     const index_t ncols(M.cols()); // No. of columns
9
10    cout << "Matrix M = " << endl << M << endl; // Print matrix
11    // Block size half the size of the matrix
12    index_t p = nrows/2, q = ncols/2;
13    // Output submatrix with left upper entry at position (i,i)
14    for(index_t i=0; i < min(p,q); i++)
15        cout << "Block (" << i << ', ' << i << ', ' << p << ', ' << q
16            << ") = " << M.block(i,i,p,q) << endl; → block read access
17    // l-value access: modify sub-matrix by adding a constant
18    M.block(1,1,p,q) += Eigen::MatrixBase<MatType>::Constant(p,q,1.0); → write access
19    cout << "M = " << endl << M << endl;
20    // r-value access: extract sub-matrix
21    MatrixXd B = M.block(1,1,p,q);
22    cout << "Isolated modified block = " << endl << B << endl;
23    // Special sub-matrices
24    cout << p << " top rows of m = " << M.topRows(p) << endl;
25    cout << p << " bottom rows of m = " << M.bottomRows(p) << endl;
26    cout << q << " left cols of m = " << M.leftCols(q) << endl;
27    cout << q << " right cols of m = " << M.rightCols(p) << endl;
28    // r-value access to upper triangular part
29    const MatrixXd T = M.template triangularView<Upper>(); //
30    cout << "Upper triangular part = " << endl << T << endl;
31    // l-value access to upper triangular part
32    M.template triangularView<Lower>() *= -1.5; //
33    cout << "Matrix M = " << endl << M << endl;
34 }

```

} special sub-matrix access operations

③



→ Review questions 1.2.1.14

Review questions 1.2.1.14

[tackle using only your memory and the Eigen documentation]

A:

Outline a C++ function

```
template <typename Matrix>
void replaceWithId(Eigen::DenseBase<Matrix> &M);
```

that checks whether the matrix is an $n \times n$ -matrix with even $n \in \mathbb{N}$ and then replaces its upper right $n/2 \times n/2$ -block with an identity matrix. Do not use any C++ loops.

B:

Given an **Eigen::VectorXd** object $\mathbf{v}$ ($\leftrightarrow \mathbf{v} \in \mathbb{R}^n$), sketch a C++ code snippet that replaces it with a vector $\tilde{\mathbf{v}}$ defined by

$$(\tilde{\mathbf{v}})_i := \begin{cases} (\mathbf{v})_n & \text{for } i = 1, \\ (\mathbf{v})_{i-1} & \text{for } i = 2, \dots, n. \end{cases}$$

Do not use C++ loops. Can you see a problem?

4

C:

Given a matrix $\mathbf{M} \in \mathbb{R}^{m,n}$ stored in an **Eigen::MatrixXd** object `M` write down a C++ code snippet that initializes another variable `Mext` of type **Eigen::MatrixXd** corresponding to

$$\widetilde{\mathbf{M}} := \begin{bmatrix} \mathbf{M} & \mathbf{0} \\ \mathbf{0}^\top & 1 \end{bmatrix} \in \mathbb{R}^{m+1,n+1}$$

using EIGEN's `<<` matrix construction operator 🐼 [EIGEN documentation](#).

D:

Learn about the methods `head()` and `tail()` from the EIGEN documentation and express them by means of the `block()` method applied to the same variable.

(You may find more review questions in the lecture document)

5

⑦