

C++ code 0.1.4: Evaluating a simple matrix-vector expression

```
2 Eigen::VectorXcd arithmetics(unsigned int n) {
3     // This declares dynamic size (signified by the letter 'X') matrices of
4     // complex numbers (signified by the letter c) with real and imaginary
5     // parts represented by doubles (signified by 'd').
6     Eigen::MatrixXcd A, C, I;
7
8     // We initialize the matrices arbitrarily:
9     // an n by n lower triangular matrix,
10    A = constantTriangular(n, 5.0).transpose().cast<std::complex<double>>();
11    // The n by n identity matrix,
12    I = Eigen::MatrixXcd::Identity(n, n);
13
14    // Declare the n by n matrix B.
15    Eigen::MatrixXcd B(n, n);
16
17    // TODO: (0-1.e) Fill in the matrix B such that  $B(k-1, l-1) = (k+l*i)/(k-l*i)$ ,
18    // where i is the imaginary unity,  $i^2 = -1$  and  $k, l=1, \dots, n$ .
19    // START
20    for (unsigned int k = 0; k < n; ++k) {
21        for (unsigned int l = 0; l < n; ++l) {
22            std::complex<double> tmp(k + 1, l + 1);
23            B(k, l) = tmp / std::conj(tmp);
24        }
25    }
26    // END
27
28    // We can perform arithmetics on matrices: +, -, *
29    // Note that for + and -, the left hand matrix and the right hand matrix have
30    // to be the same size, and that matrix multiplication * is only defined if
31    // the number of columns in the left hand matrix is the same as the number of
32    // rows in the right hand matrix.
33    C = B * (A - 5.0 * I);
34
35    // Eigen::VectorXcd is shorthand for
36    // Eigen::Matrix< std::complex<double>, Eigen::Dynamic, 1>.
37    // Hence, all the same arithmetics work for vectors.
38    Eigen::VectorXcd u, v;
39
40    // TODO: (0-1.e) Initialize the entries of u as the integers from 1 to n,
41    // multiply u by the matrix C, and store the result in v.
42    // START
43    u = Eigen::VectorXcd::LinSpaced(n, 1, n); // Or use a for-loop.
44    v = C * u;
45    // END
46
47    return v;
48 }
```

Get it on  GitLab (MatrixClass.hpp).