

C++ code 0.1.2: Initialization of upper triangular part of a matrix

```
2 Eigen::MatrixXd constantTriangular(unsigned int n, double val) {
3     // Now we do not know the size of our matrix at compile time.
4     // Hence, we use the special value Eigen::Dynamic to set the size of A.
5     Eigen::Matrix<double, Eigen::Dynamic, Eigen::Dynamic> A;
6
7     // Eigen::Matrix has a method Zero(n_rows,n_cols) which returns the n_rows by
8     // n_cols matrix whose entries are all equal to zero.
9     A = Eigen::Matrix<double, Eigen::Dynamic, Eigen::Dynamic>::Zero(n, n);
10
11    // To get a triangular matrix, we must set the values above the diagonal to
12    // val. This we can do by a nested for-loop.
13    for (unsigned int row = 0; row < n; ++row) {
14        // TODO: (0-1.c) Write the inner for-loop to initialize A.
15        // Hint: We can access and change the entries of A using parentheses, i.e.
16        // A(row,col) = val; Note that indexing starts at the value 0 (as usual),
17        // and not 1 (as in Matlab).
18
19        // START
20        for (unsigned int col = row; col < n; ++col) {
21            A(row, col) = val;
22        }
23        // END
24    }
25
26    // instead of the for-loop initialization, triangularView can be used
27    // A.triangularView<Eigen::Upper>() = Eigen::MatrixXd::Constant(n, n, val);
28
29    return A;
30 }
```

Get it on  GitLab (MatrixClass.hpp).