Pay attention to the efficient initialization of the sparse matrix in EIGEN. Since we know in advance that every row will have exactly one non-zero entry, we can reserve that space.

C++ code 11.5.9: Sub-problem (11-5.e): function `compBmat()` ↪ GitLab

```
2  Eigen::SparseMatrix<double> compBmat(int Ml, int Mr, double h) {
3    const int N = 2 * (Ml + Mr + 1);
4    Eigen::SparseMatrix<double> A(N, N);
5    // Tell Eigen that the matrix is diagonal, which allows efficient
6    // initialization
7    A.reserve(Eigen::VectorXi::Ones(N));
8    for (int i = 0; i < N; i += 2) {
9      A.insert(i, i) = h;
10     A.insert(i + 1, i + 1) = h * h * h / 12.0;
11   }
12   return A;
13 }
```