

C++ code 3.16.7: Method `solve()` of class `CompactStorageQR`

```
2 Eigen::MatrixXd CompactStorageQR::solve(const Eigen::MatrixXd &B) const {
3     assert((B.rows() == n_) && "Wrong size of right-hand side");
4     Eigen::MatrixXd X{B};
5
6     // TODO: (3-16.b) Solve the linear system of equations given by AX = B
7     with
8     // complexity O(n^2k).
9     // START
10    // Check invertibility by inspecting diagonal
11    // elements of R-factor
12    constexpr double atol = 1.0E-16;
13    constexpr double rtol = 1.0E-8;
14    double maxrii = std::abs(M_(0, 0));
15    for (unsigned int k = 1; k < n_; ++k) {
16        const double tmp = std::abs(M_(k, k));
17        if (maxrii < tmp) {
18            maxrii = tmp;
19        }
20    }
21    for (unsigned int k = 1; k < n_; ++k) {
22        const double rii = std::abs(M_(k, k));
23        if ((rii < rtol * maxrii) || (rii < atol)) {
24            throw std::runtime_error("CompactStorageQR::solve: matrix not regular");
25        }
26    }
27    // Multiplication with transposed Q-factors
28    Eigen::VectorXd u(n_);
29    for (unsigned int k = 0; k < n_ - 1; ++k) {
30        double sn{0};
31        for (unsigned int j = k + 1; j < n_; ++j) {
32            u[j] = M_(j, k);
33            sn += u[j] * u[j];
34        }
35        u[k] = std::sqrt(1.0 - sn);
36        // Multiplication with current Q-factor
37        X -= 2 * u * (u.transpose() * X);
38        u[k] = 0.0;
39    }
40    // Solve upper triangular linear system RX = Q^T B
41    X = M_.template triangularView<Eigen::Upper>().solve(X);
42    // END
43    return X;
44 }
```

Get it on  GitLab (compactstorageqr.hpp).