

```

2 Eigen::VectorXd solveExtendedNormalEquations(const Eigen::MatrixXd& D) {
3     Eigen::VectorXd x;
4     const unsigned int n = D.cols();
5     const unsigned int m = n * (n - 1) / 2;
6     // TODO: (3-10.c) Solve the extended normal equations with a sparse
7     solver
8     // that Eigen provides
9     // START
10    // initialize right hand side vector b as in (3.10.2)
11    Eigen::VectorXd b_ext = Eigen::VectorXd::Zero(m + n - 1);
12    unsigned int curr = 0;
13    for (unsigned int i = 0; i < n - 1; ++i) {
14        for (unsigned int k = i + 1; k < n; ++k) {
15            b_ext(curr) = D(i, k);
16            ++curr;
17        }
18    }
19    // initialize matrix A and convert to triplet format
20    Eigen::SparseMatrix<double> A = initA(n);
21    std::vector<Eigen::Triplet<double>> tripletsA = convertToTriplets(A);
22    // define triplets for extended normal equation
23    std::vector<Eigen::Triplet<double>> tripletsAext;
24    for (unsigned int k = 0; k < m; ++k) {
25        // - I block
26        tripletsAext.push_back(Eigen::Triplet<double>(k, k, -1.0));
27    }
28    for (auto t : tripletsA) {
29        // A block
30        tripletsAext.push_back(
31            Eigen::Triplet<double>(t.row(), m + t.col(), t.value()));
32        // A.transpose() block
33        tripletsAext.push_back(
34            Eigen::Triplet<double>(m + t.col(), t.row(), t.value()));
35    }
36
37    Eigen::SparseMatrix<double> A_ext(n - 1 + m, n - 1 + m);
38    A_ext.setFromTriplets(tripletsAext.begin(), tripletsAext.end());
39    // solve extended system and extract the last n-1 entries
40    // of the solution vector
41    Eigen::SparseLU<Eigen::SparseMatrix<double>> solver(A_ext);
42    x = solver.solve(b_ext).tail(n - 1);
43    // END
44    return x;
45 }

```