

C++11-code 2.9.4: LU-decomposition based solution

```

2  bool testSolveLSE(const Eigen::MatrixXd& R, const Eigen::VectorXd& v,
3                  const Eigen::VectorXd& u, const Eigen::VectorXd& b,
4                  Eigen::VectorXd& x) {
5      bool areTheSame = false;
6
7      // TODO: (2-9.e)
8      // i) Create the system matrix A and solve the LSE, using
9      // an Eigen LU-solver. Store the solution in x.
10     // ii) Solve the LSE with solvelse(), and calculate the
11     // difference between this solution and x. Return true
12     // if and only if the norm of this difference is close
13     // enough to zero.
14     // START
15     // size of R, which is size of u, v, and size of bb is n+1
16     const unsigned int n = R.rows();
17     assert(n == R.cols() && n == u.size() && n == v.size() && n + 1 == b.size() &&
18            "Size mismatch!");
19
20     Eigen::MatrixXd A(n + 1, n + 1);
21     // Build matrix using blocks
22     A << R, v, u.transpose(), 0;
23
24     // Solve system using LU decomposition
25     x = A.partialPivLu().solve(b);
26
27     // Solve system using solvelse()
28     Eigen::VectorXd y;
29     solvelse(R, v, u, b, y);
30
31     // We choose an acceptable tolerance.
32     constexpr double tolerance = 1e-8;
33     areTheSame = ((x - y).norm() < tolerance * b.norm());
34
35     // END
36     return areTheSame;
37 }

```

Get it on  GitLab (blockdecomp.hpp).