C++11 code 2.8.1: Sub-problem (2-8.c): Implementation of `solve()` ⇝ GitLab

```
template <lf::assemble::EntityMatrixProvider ELMAT_PROVIDER,
          lf::assemble::EntityMatrixProvider ELVEC_PROVIDER>
Eigen::VectorXd solve(ELMAT_PROVIDER &elmat_provider,
                      ELVEC_PROVIDER &elvec_provider) {
  // Use one of LehrFEM++'s default meshes.  Try different meshes by changing the
  // function index parameter.  See the documentation of that function for
  // details ablut the available meshes
  std::shared_ptr<const lf::mesh::Mesh> mesh_p =
      lf::mesh::test_utils::GenerateHybrid2DTestMesh(8, 1.0 / 3.0);
  // We use a linear Lagrangian FE space
  auto fe_space =
      std::make_shared<lf::uscalfe::FeSpaceLagrangeO1<double>>(mesh_p);
  // Reference to current mesh, obtained from the FE space
  const lf::mesh::Mesh &mesh{*(fe_space->Mesh())};
  // Obtain local->global index mapping for current finite element space
  const lf::assemble::DofHandler &dofh{fe_space->LocGlobMap()};
  // Dimension of finite element space`
  const lf::base::size_type N_dofs(dofh.NumDofs());

  // Matrix in triplet format holding Galerkin matrix, zero initially.
  lf::assemble::COOMatrix<double> A(N_dofs, N_dofs);
  // Invoke assembly on cells (co-dimension = 0).  The element matrix provider is
  // passed as an argument
  lf::assemble::AssembleMatrixLocally(0, dofh, dofh, elmat_provider, A);
  Eigen::SparseMatrix<double> A_crs = A.makeSparse();

  // Right-hand side vector; has to be set to zero initially
  Eigen::Matrix<double, Eigen::Dynamic, 1> phi(N_dofs);
  phi.setZero();
  // Invoke assembly on cells (codim == 0).  The element vector provider is
  // passed as an argument
  AssembleVectorLocally(0, dofh, elvec_provider, phi);

  // Define solution vector
  Eigen::VectorXd sol_vec = Eigen::VectorXd::Zero(N_dofs);

  // Solve linear system using Eigen's sparse direct elimination
  Eigen::SparseLU<Eigen::SparseMatrix<double>> solver;
  solver.compute(A_crs);
  // Make sure LU-decomposition could be computed
  if (solver.info() != Eigen::Success) {
    throw std::runtime_error("Could not decompose the matrix");
  }
  // Backward substitution
  sol_vec = solver.solve(phi);
```