C++ code 3.8.7: Function `solveImpedanceBVP`: Assembly of right-hand-side vector and solution of linear system of equations. ⇝ GITHUB

```
  // I.iii : Computing right-hand side vector
  // Right-hand side source function f
  auto mf_f = lf::mesh::utils::MeshFunctionGlobal(
      [](Eigen::Vector2d x) -> double { return 0.0; });
  lf::uscalfe::ScalarLoadElementVectorProvider<double, decltype(mf_f)>
      elvec_builder(fe_space_p, mf_f);
  // Invoke assembly on cells (codim == 0)
  AssembleVectorLocally(0, dofh, elvec_builder, phi);

  // I.iv : Imposing essential boundary conditions
  // Dirichlet data
  auto mf_g = lf::mesh::utils::MeshFunctionGlobal(
      [&g](Eigen::Vector2d x) -> double { return g.dot(x); });
  // Inspired by the example in the documentation of
  // InitEssentialConditionFromFunction()
  //
     https://craffael.github.io/lehrfempp/namespacelf_1_1uscalfe.html#a5afbd94919f0382cf
  // Creating a predicate that will guarantee that the computations are carried
  // only on the exterior boundary edges of the mesh using the boundary flags
  auto edges_predicate_Dirichlet =
      [&bd_flags](const lf::mesh::Entity &edge) -> bool {
    if (bd_flags(edge)) {
      auto endpoints = lf::geometry::Corners(*(edge.Geometry()));
      if (endpoints(0, 0) <= 0.05 || 0.95 <= endpoints(0, 0) ||
          endpoints(1, 0) <= 0.05 || 0.95 <= endpoints(1, 0)) {
        return true;
      }
    }
    return false;
  };
  // Determine the fixed dofs on the boundary and their values
  // Alternative: See lecturedemoDirichlet() in
  //
     https://github.com/craffael/lehrfempp/blob/master/examples/lecturedemos/lecturedemo
  auto edges_flag_values_Dirichlet{lf::fe::InitEssentialConditionFromFunction(
      *fe_space_p, edges_predicate_Dirichlet, mf_g)};
  // Eliminate Dirichlet dofs from the linear system
  lf::assemble::FixFlaggedSolutionCompAlt<double>(
      [&edges_flag_values_Dirichlet](lf::assemble::glb_idx_t gdof_idx) {
        return edges_flag_values_Dirichlet[gdof_idx];
      },
      A, phi);

  // Assembly completed! Convert COO matrix A into CRS format using Eigen's
  // internal conversion routines.
  Eigen::SparseMatrix<double> A_sparse = A.makeSparse();

// II : SOLVING  THE LINEAR SYSTEM
  // II.i : Setting up Eigen's sparse direct elimination
  Eigen::SparseLU<Eigen::SparseMatrix<double>> solver;
  solver.compute(A_sparse);
  LF_VERIFY_MSG(solver.info() == Eigen::Success, "LU decomposition failed");
  // II.ii : Solving
  discrete_solution = solver.solve(phi);
  LF_VERIFY_MSG(solver.info() == Eigen::Success, "Solving LSE failed");
```