C++ code 5.5.18: Incomplete listing of `newtonNextIt()`

```
void newtonNextIt(
    std::shared_ptr<const lf::uscalfe::FeSpaceLagrangeO1<double>> fes_p,
    Eigen::VectorXd &mu_vec, Eigen::VectorXd &rhs_vec) {
  lf::mesh::utils::MeshFunctionConstant mf_one(1.);
  lf::mesh::utils::MeshFunctionConstant mf_zero(0.);
  const lf::fe::MeshFunctionFE mf_uh_prev(fes_p, mu_vec);

  const FunctionMFWrapper mf_coeff1_uh_prev(
      [redacted], [redacted]);
  const FunctionMFWrapper mf_coeff2_uh_prev(
      [redacted], [redacted]);
  // We define some variables for easy access
  const lf::mesh::Mesh &mesh{*(fes_p->Mesh())};
  const lf::assemble::DofHandler &dofh{fes_p->LocGlobMap()};
  const std::size_t N_dofs(dofh.NumDofs());
  // Assemble matrix A
  lf::assemble::COOMatrix<double> A(N_dofs, N_dofs);
  lf::uscalfe::ReactionDiffusionElementMatrixProvider<double>
      elmat_provider(fes_p, [redacted], [redacted]);
  lf::assemble::AssembleMatrixLocally(0, dofh, dofh, elmat_provider, A);

  // Assemble Galerkin in Eigen::SparseMatrix format
  lf::assemble::COOMatrix<double> M(N_dofs, N_dofs);
  lf::uscalfe::ReactionDiffusionElementMatrixProvider<double>
    elmat_provider_M(fes_p, mf_one, mf_zero);
  lf::assemble::AssembleMatrixLocally(0, dofh, dofh, elmat_provider_M, M);
  Eigen::SparseMatrix<double> M_crs = M.makeSparse();

  // Construct the right-hand side of the system
  Eigen::VectorXd phi(N_dofs); phi.setZero();
  phi += [redacted];
  phi -= [redacted];
  lf::uscalfe::ScalarLoadElementVectorProvider<double>
      elvec_builder(fes_p, [redacted]);
  lf::assemble::AssembleVectorLocally(0, dofh, elvec_builder, phi);

  // Enforce homogeneous Dirichlet boundary conditions
  auto bd_flags{lf::mesh::utils::flagEntitiesOnBoundary(fes_p->Mesh(), 2)};
  // Elimination of degrees of freedom on the boundary.  Also sets the
  // corresponding entries of rhs_vec to zero.
  lf::assemble::FixFlaggedSolutionComponents<double>(
      [&bd_flags,
       &dofh](lf::assemble::glb_idx_t gdof_idx) -> std::pair<bool, double> {
        const lf::mesh::Entity &node{dofh.Entity(gdof_idx)};
        return (bd_flags(node) ? std::make_pair(true, 0.0)
                               : std::make_pair(false, 0.0));
      },
      A, phi);
  // Convert matrix A in COO format to A_crs in Eigen::SparseMatrix
     format
```