C++ code 9.9.13: Implementation of `assembleGalerkinMatrices` ➔ GitLab

```cpp
template <typename DIFF_COEFF>
std::pair<Eigen::SparseMatrix<double>, Eigen::SparseMatrix<double>>
assembleGalerkinMatrices(const lf::assemble::DofHandler &dofh, DIFF_COEFF &&c) {
  std::pair<Eigen::SparseMatrix<double>, Eigen::SparseMatrix<double>> A_M;
  // Obtain mesh and finite element space
  std::shared_ptr<const lf::mesh::Mesh> mesh = dofh.Mesh();
  const lf::uscalfe::size_type N_dofs(dofh.NumDofs());
  auto fe_space =
      std::make_shared<lf::uscalfe::FeSpaceLagrangeO1<double>>(mesh);

  // Matrix format for assembly
  lf::assemble::COOMatrix<double> A_COO(N_dofs, N_dofs);
  lf::assemble::COOMatrix<double> M_COO(N_dofs, N_dofs);

  // Function handles for Stiffness and Mass matrix
  auto one = [](Eigen::Vector2d x) -> double { return 1.0; };
  auto zero = [](Eigen::Vector2d x) -> double { return 0.0; };

  // Mesh functions
  lf::mesh::utils::MeshFunctionGlobal mf_c{c};
  lf::mesh::utils::MeshFunctionGlobal mf_one{one};
  lf::mesh::utils::MeshFunctionGlobal mf_zero{zero};

  // Element matrix provider
  lf::uscalfe::ReactionDiffusionElementMatrixProvider<double, decltype(mf_c),
                                                      decltype(mf_zero)>
      elMat_Stiff(fe_space, mf_c, mf_zero);
  lf::uscalfe::ReactionDiffusionElementMatrixProvider<double, decltype(mf_zero),
                                                      decltype(mf_one)>
      elMat_Mass(fe_space, mf_zero, mf_one);

  // Assembly
  lf::assemble::AssembleMatrixLocally(0, dofh, dofh, elMat_Stiff, A_COO);
  lf::assemble::AssembleMatrixLocally(0, dofh, dofh, elMat_Mass, M_COO);

  // Sparse matrix format
  Eigen::SparseMatrix<double> A = A_COO.makeSparse();
  Eigen::SparseMatrix<double> M = M_COO.makeSparse();

  A_M = std::make_pair(A, M);

  return A_M;
}
```