C++ code 9.10.22: Implementation of `computeMV()` ➙ GitLab

```
template <typename RHOFUNCTION>
Eigen::SparseMatrix<double> computeMV(
    std::shared_ptr<lf::uscalfe::FeSpaceLagrangeO1<double>> fe_space_V,
    RHOFUNCTION &&rho) {
  // TOOLS AND DATA
  std::shared_ptr<const lf::mesh::Mesh> mesh_p = fe_space_V->Mesh();
  const lf::assemble::DofHandler &dofh_V{fe_space_V->LocGlobMap()};
  const lf::uscalfe::size_type N_dofs_V(dofh_V.NumDofs());
  // For returning the matrix
  Eigen::SparseMatrix<double> M_V(N_dofs_V, N_dofs_V);
  auto zero_mf = lf::mesh::utils::MeshFunctionGlobal(
      [](Eigen::Vector2d x) -> double { return 0.0; });
  auto rho_mf = lf::mesh::utils::MeshFunctionGlobal(
      [rho](Eigen::Vector2d x) -> double { return rho(x); });

  // ASSEMBLY
  // Matrix in triplet format holding Galerkin matrix, zero initially.
  lf::assemble::COOMatrix<double> M_V_COO(N_dofs_V, N_dofs_V);
  // ReactionDiffusionElementMatrixProvider<SCALAR,DIFF_COEFF,REACTION_COEFF,
  // quadrule>
  // Tell this ENTITY_MATRIX_PROVIDER object to use the local trapezoidal rule
  // Lehrfempp requires the specification of a quadrature rule for
  // quadrilateral. A generic rule for QUAD that is NOT used in this problem is
  // passed. Alternatively : see computeMV_alt
  lf::uscalfe::ReactionDiffusionElementMatrixProvider<double, decltype(zero_mf),
                                                      decltype(rho_mf)>
      elmat_builder(fe_space_V, zero_mf, rho_mf,
                    {{lf::base::RefEl::kTria(), make_TriaQR_TrapezoidalRule()},
                     {lf::base::RefEl::kQuad(), lf::quad::make_QuadQR_P4O4()}});
  lf::assemble::AssembleMatrixLocally(0, dofh_V, dofh_V, elmat_builder,
                                      M_V_COO);
  M_V = M_V_COO.makeSparse();
  std::cout << "Assembly: M_V finished" << std::endl;

  return M_V;
}
```