C++ code 9.10.20: Alternative implementation of `computeMV()` ↪ GitLab

```
template <typename RHOFUNCTION>
Eigen::SparseMatrix<double> computeMV_alt(
    std::shared_ptr<lf::uscalfe::FeSpaceLagrangeO1<double>> fe_space_V,
    RHOFUNCTION &&rho) {
  const lf::mesh::Mesh &mesh{*(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());

  // Reserve space for a sparse diagonal matrix
  Eigen::SparseMatrix<double> M_V(N_dofs_V, N_dofs_V);
  M_V.reserve(Eigen::VectorXi::Constant(N_dofs_V, 1));

  // Determine area of mesh neighborhood of every node
  const auto areas{areasOfAdjacentCells(fe_space_V->Mesh())};
  // Loop over all nodes
  for (const lf::mesh::Entity *node : mesh.Entities(2)) {
    // Obtain position vector of node
    const Eigen::Vector2d node_pos{
        lf::geometry::Corners(*(node->Geometry())).col(0)};
    // Evaluate coefficient function at node
    const double rho_val = rho(node_pos);
    // Find global index of dof associated with the node
    const std::span<const lf::assemble::gdof_idx_t> node_dofs(
        dofh_V.GlobalDofIndices(*node));
    LF_ASSERT_MSG(node_dofs.size() == 1, "A node has to carry exatly one dof!");
    // Global index of tent function sitting on the node
    const lf::assemble::gdof_idx_t dof_no = node_dofs[0];
    M_V.insert(dof_no, dof_no) = areas(*node) * rho_val / 3.0;
  }
  return M_V;
}
```