C++ code 9.18.21: Implementation of `timestepDissipativeWaveEquation()`

```
Eigen::VectorXd timestepDissipativeWaveEquation(
    std::shared_ptr<const lf::uscalfe::FeSpaceLagrangeO2<double>> fe_space_p,
    double T, unsigned int M, Eigen::VectorXd mu0, Eigen::VectorXd nu0) {
  // Pointer and reference to current mesh
  std::shared_ptr<const lf::mesh::Mesh> mesh_p = fe_space_p->Mesh();
  const lf::mesh::Mesh &mesh{*mesh_p};
  // Obtain local->global index mapping
  // for current finite element space
  const lf::assemble::DofHandler &dofh{fe_space_p->LocGlobMap()};
  // Dimension of finite element space = number of nodes of the mesh
  const lf::base::size_type N_dofs(dofh.NumDofs());
  LF_VERIFY_MSG(mu0.size() == N_dofs, "Wrong length of mu0");
  LF_VERIFY_MSG(nu0.size() == N_dofs, "Wrong length of nu0");
  // The solution vector at time T
  Eigen::VectorXd mu = mu0;

  // Obtain matrices in triplet format
  const auto [M_COO, B_COO, A_COO] = computeGalerkinMatrices(fe_space_p);
  LF_VERIFY_MSG((M_COO.cols() == N_dofs) && (M_COO.rows() == N_dofs),
                "Wrong size of M");
  LF_VERIFY_MSG((B_COO.cols() == N_dofs) && (B_COO.rows() == N_dofs),
                "Wrong size of B");
  LF_VERIFY_MSG((A_COO.cols() == N_dofs) && (A_COO.rows() == N_dofs),
                "Wrong size of A");
  // For convenience convert in CRS format in order to be able to exploit
  // Eigen's linear-algebra operations.
  const Eigen::SparseMatrix<double> M_crs{M_COO.makeSparse()};
  const Eigen::SparseMatrix<double> B_crs{B_COO.makeSparse()};
  const Eigen::SparseMatrix<double> A_crs{A_COO.makeSparse()};
  // Build sparse matrices M+0.5*tau*B and M-0.5*tau*B
  // Timestep size
  const double tau = T / M;
  const std::vector<Eigen::Triplet<double>> &B_trp = B_COO.triplets();
  std::vector<Eigen::Triplet<double>> MBp_trp = M_COO.triplets();
  std::vector<Eigen::Triplet<double>> MBm_trp = M_COO.triplets();
  for (auto &triplet : B_trp) {
    MBp_trp.emplace_back(triplet.row(), triplet.col(),
                         0.5 * tau * triplet.value());
    MBm_trp.emplace_back(triplet.row(), triplet.col(),
                         -0.5 * tau * triplet.value());
  }
  Eigen::SparseMatrix<double> MBp(N_dofs, N_dofs);
  MBp.setFromTriplets(MBp_trp.begin(), MBp_trp.end());
  Eigen::SparseMatrix<double> MBm(N_dofs, N_dofs);
  MBm.setFromTriplets(MBm_trp.begin(), MBm_trp.end());
  // We have to solve a linear system with the matrix M+0.5*tau*B
  Eigen::SparseLU<Eigen::SparseMatrix<double>> solver_MBp;
  // LU decomposition, important: outside main timestepping loop
  solver_MBp.compute(MBp);
  LF_VERIFY_MSG(solver_MBp.info() == Eigen::Success,
                "LU decomposition of M+0.5*tau*B failed");
  // The auxiliary vector in the timestepping loop
  Eigen::VectorXd nu(N_dofs);
  // Special initial step
  {
    Eigen::SparseLU<Eigen::SparseMatrix<double>> solver_M;
```