C++11 code 9.1.22: Implementation of constructor of Radau3MOLTimestepper. ➝ GitLab

```
Radau3MOLTimestepper::Radau3MOLTimestepper(const lf::assemble::DofHandler &dofh)
    : dofh_(dofh) {
  std::cout << "\n>> Constructing SRadau3MOLTimestepper " << std::endl;
  auto mesh_p = dofh.Mesh();  // pointer to current mesh

  // Instantiating Galerkin matrices to be pre-computed
  // Dimension of finite element space
  const lf::uscalfe::size_type N_dofs(dofh.NumDofs());
  // Matrices in triplet format holding Galerkin matrices, zero initially.
  lf::assemble::COOMatrix<double> A_COO(N_dofs,
                                        N_dofs);  // element matrix Laplace
  lf::assemble::COOMatrix<double> M_COO(N_dofs,
                                        N_dofs);  // element mass matrix

  std::cout << "> Initializing the Galerking local matrices builders"
            << std::endl;
  // Initialize classes containing the information required for the
  // local computations of the Galerkin matrices. Simple implementations of
  // LinFEMassMatrixProvider and TrapRuleLinFEElemVecProvider adapted to this
  // particular problem was written to spare some of the overhead calculations
  // involved in the use of the more general LehrFEM++ matrices providers.
  lf::uscalfe::LinearFELaplaceElementMatrix elLapMat_builder;
  LinFEMassMatrixProvider elMassMat_builder;

  std::cout << "> Assembling Galerking matrices in COO format" << std::endl;
  // Compute the Galerkin matrices
  // Invoke assembly on cells (co-dimension = 0 as first argument)
  // Information about the mesh and the local-to-global map is passed through
  // a Dofhandler object, argument 'dofh'. This function call adds triplets to
  // the internal COO-format representation of the sparse matrices A and M.
  lf::assemble::AssembleMatrixLocally(0, dofh, dofh, elLapMat_builder, A_COO);
  lf::assemble::AssembleMatrixLocally(0, dofh, dofh, elMassMat_builder, M_COO);

  // Enforcing zero Dirichlet boundary conditions
  // Obtain an array of boolean flags for the vertices of the mesh: 'true'
  // indicates that the vertex lies on the boundary.
  auto bd_flags{lf::mesh::utils::flagEntitiesOnBoundary(mesh_p, 2)};
  // Index predicate for the selectvals FUNCTOR of dropMatrixRowsColumns
  auto bdy_vertices_selector = [&bd_flags, &dofh](unsigned int idx) -> bool {
    return bd_flags(dofh.Entity(idx));
  };
  dropMatrixRowsColumns(bdy_vertices_selector, A_COO);
  dropMatrixRowsColumns(bdy_vertices_selector, M_COO);

  std::cout << "> Converting triplets to sparse matrices" << std::endl;
  // Creating the private Galerkin stiffness and mass matrices
  A_ = A_COO.makeSparse();
  Eigen::SparseMatrix<double> M = M_COO.makeSparse();

  // Runge-Kutta matrices defining the 2-stage Radau timestepping. In the
  // Butcher tableau, this corresponds to c = (1/3 1)^T (top-left column
  // vector), b^T = (3/4 1/4) (bottom-right row vector), U_11 = 5/12, U_12 =
  // -1/12, U_21 = 3/4, U_22 = 1/4 (top-right block); values are fixed in
  // time
  // clang-format off
    U_ << 5.0/12.0, -1.0/12.0,
```