**C++ code 9.7.17: Implementation of the method `solveWaveABC2D()` of WaveABC2DTimestepper ➙ GitLab**

```cpp
template <typename FUNC_RHO, typename FUNC_MU0, typename FUNC_NU0>
Eigen::VectorXd WaveABC2DTimestepper<FUNC_RHO, FUNC_MU0,
                                     FUNC_NU0>::solveWaveABC2D(FUNC_MU0 mu0,
                                                               FUNC_NU0 nu0) {
  std::cout << "\nSolving variational problem of WaveABC2D." << std::endl;
  Eigen::VectorXd sol;

  // Initial conditions
  auto mf_mu0 = lf::mesh::utils::MeshFunctionGlobal(mu0);
  auto mf_nu0 = lf::mesh::utils::MeshFunctionGlobal(nu0);
  Eigen::VectorXd nu0_nodal = lf::fe::NodalProjection(*fe_space_p_, mf_nu0);
  Eigen::VectorXd mu0_nodal = lf::fe::NodalProjection(*fe_space_p_, mf_mu0);

  // Setup loop and tools
  Eigen::VectorXd cur_step_vec(2 * N_dofs_);
  Eigen::VectorXd next_step_vec(2 * N_dofs_);
  cur_step_vec.head(N_dofs_) = nu0_nodal;
  cur_step_vec.tail(N_dofs_) = mu0_nodal;
  std::cout << "Performing discrete evolution..." << std::endl;
  progress_bar progress{std::clog, 55u, "Timestepping"};
  double progress_pourcentage;

  // Performing timesteps
  for (int i = 1; i < M_; i++) {
    next_step_vec = solver_.solve(R_ * cur_step_vec);
    cur_step_vec = next_step_vec;

    // Display progress
    progress_pourcentage = ((double)i + 1.0) / M_ * 100.0;
    progress.write(progress_pourcentage / 100.0);
  }

  full_sol_ = cur_step_vec;
  sol = full_sol_.tail(N_dofs_);
  // Full solution is computed
  timestepping_performed_ = true;

  return sol;
}  // solveWaveABC2D
```