C++ code 2.4.20: Implementation of `solveC()` ➝ GitLab

```
template <typename FUNCTOR1, typename FUNCTOR2>
Eigen::VectorXd solveC(const Eigen::VectorXd &mesh, FUNCTOR1 &&alpha,
                       FUNCTOR2 &&gamma) {
  // Nodes are indexed as 0=x_0 < x_1 < ... < x_N = 1
  unsigned N = mesh.size() - 1;
  // Initializations (notice initialization with zeros here)
  Eigen::VectorXd u(N + 1);                          // solution vec
  Eigen::SparseMatrix<double> A(N + 1, N + 1);  // laplacian galerkin mat
  Eigen::SparseMatrix<double> M(N + 1, N + 1);  // mass galerkin mat
  Eigen::SparseMatrix<double> L(N + 1, N + 1);  // full galerkin mat

  // I. Build the (full) Galerkin matrix L for the lin. sys.
  // I.i Compute the entries of the Laplace Galerkin matrix A
  std::vector<Eigen::Triplet<double>> triplets_A = computeA(mesh, alpha);
  // I.ii Compute the entries of the mass matrix M
  std::vector<Eigen::Triplet<double>> triplets_M = computeM(mesh, gamma);
  // I.iii Assemble the sparse matrices
  A.setFromTriplets(triplets_A.begin(), triplets_A.end());
  M.setFromTriplets(triplets_M.begin(), triplets_M.end());
  L = A + M;  // Full Galerkin matrix of the LSE

  // II. Build the right hand side source vector
  Eigen::VectorXd rhs_vec =
      computeRHS(mesh, [](double) -> double { return 1.0; });

  // IV. Solve the LSE A*u = rhs_vec using an Eigen solver
  Eigen::SimplicialLDLT<Eigen::SparseMatrix<double>> solver;
  solver.compute(L);
  if (solver.info() != Eigen::Success) {
    throw std::runtime_error("Could not decompose the matrix");
  }
  u = solver.solve(rhs_vec);

  return u;
}  // solveC
```