C++ code 2.4.25: Implementation of `solveB()` ➝ GitLab

```
2  template <typename FUNCTOR1, typename FUNCTOR2>
3  Eigen::VectorXd solveB(const Eigen::VectorXd &mesh, FUNCTOR1 &&alpha,
4                         FUNCTOR2 &&f, double u0, double u1) {
5    // Nodes are indexed as 0=x_0 < x_1 < ... < x_N = 1
6    unsigned N = mesh.size() - 1;
7    // Initializations
8    Eigen::VectorXd u(N + 1);                       // solution vec
9    Eigen::SparseMatrix<double> A(N + 1, N + 1);  // laplacian galerkin mat
10   // Some tool variables
11   double dx_left, dx_right;  // cell widths
12
13   // I. Build the Galerkin matrix A
14   std::vector<Eigen::Triplet<double>> triplets = computeA(mesh, alpha);
15   A.setFromTriplets(triplets.begin(), triplets.end());
16
17   // II. Build the right hand side source vector
18   Eigen::VectorXd rhs_vec = computeRHS(mesh, f);
19
20   // III. Enforce dirichlet boundary conditions
21   // Proceeding as in solveA, we begin by dropping the first and last rows
22   // and columns of the galerkin matrix. The rhs_vec needs to be modified to
23   // account for the non-homegenous Dirichlet boundary conditions. It is
24   // modified using an offset function.
25   Eigen::SparseMatrix<double> A_reduced = A.block(1, 1, N - 1, N - 1);
26   Eigen::VectorXd rhs_vec_reduced = rhs_vec.segment(1, N - 1) -
27                                     A.block(1, 0, N - 1, 1) * u0 -
28                                     A.block(1, N, N - 1, 1) * u1;
29
30   // IV. Solve the LSE A*u = rhs_vec using an Eigen solver
31   Eigen::SimplicialLDLT<Eigen::SparseMatrix<double>> solver;
32   solver.compute(A_reduced);
33   if (solver.info() != Eigen::Success) {
34     throw std::runtime_error("Could not decompose the matrix");
35   }
36   u.segment(1, N - 1) = solver.solve(rhs_vec_reduced);
37
38   // The solution vector still needs to be supplemented with the known
39   // boundary values
40   u(0) = u0;  // left boundary node
41   u(N) = u1;  // right boundary node
42   return u;
43 }  // solveB
```