C++ code 9.10.26: Building the matrix $\widetilde{B}$: `computeB` ↪ GitLab

```
Eigen::SparseMatrix<double> computeB(const lf::assemble::DofHandler &dofh_V,
                                     const lf::assemble::DofHandler &dofh_Q) {
  // TOOLS AND DATA
  auto mesh_p = dofh_V.Mesh();
  // Dimension of finite element space
  const lf::uscalfe::size_type N_dofs_Q(dofh_Q.NumDofs());
  const lf::uscalfe::size_type N_dofs_V(dofh_V.NumDofs());
  // Sparse matrix to be returned
  Eigen::SparseMatrix<double> B(N_dofs_Q, N_dofs_V);

  // ASSEMBLY
  lf::assemble::COOMatrix<double> B_COO(N_dofs_Q, N_dofs_V);
  BElemMatProvider BElemMat_builder;
  lf::assemble::AssembleMatrixLocally(0, dofh_V, dofh_Q, BElemMat_builder,
                                      B_COO);

  // Node-index array of flags marking boundary nodes
  auto bd_flags{lf::mesh::utils::flagEntitiesOnBoundary(mesh_p, 2)};
  // Set columns of the matrix to zero using a predicate that evaluates
     to true,
  // if the column index is that of a d.o.f.  associated with a node on
     the
  // boundary
  B_COO.setZero([&bd_flags, &dofh_V](lf::assemble::gdof_idx_t /*i*/,
                                     lf::assemble::gdof_idx_t j) -> bool {
    return bd_flags(dofh_V.Entity(j));
  });

  // Convert from triplet to CRS format
  B = B_COO.makeSparse();

  return B;
}
```