C++ code 10.7.17: Implementation of `Eval()` method, which computes the element matrices $\mathbf{B}_K$

```
template <typename VELOCITY>
typename SUFEM::AdvectionElementMatrixProvider<VELOCITY>::ElemMat
AdvectionElementMatrixProvider<VELOCITY>::Eval(const lf::mesh::Entity &cell) {
  // Topological type of the cell
  const lf::base::RefEl ref_el{cell.RefEl()};
  // Obtain precomputed information about values of local shape
     functions
  // and their gradients at quadrature points.
  lf::uscalfe::PrecomputedScalarReferenceFiniteElement<double> &pfe =
      fe_precomp_[ref_el.Id()];
  if (!pfe.isInitialized()) {
    // Accident:  cell is of a type not covered by finite element
    // specifications or there is no quadrature rule available for this
    // reference element type
    std::stringstream temp;
    temp << "No local shape function information or no quadrature rule for "
            "reference element type "
         << ref_el;
    throw lf::base::LfException(temp.str());
  }

  // Query the shape of the cell
  const lf::geometry::Geometry *geo_ptr = cell.Geometry();
  LF_ASSERT_MSG(geo_ptr != nullptr, "Invalid geometry!");
  LF_ASSERT_MSG((geo_ptr->DimLocal() == 2),
                "Only 2D implementation available!");

  // Physical dimension of the cell (must be 2)
  const lf::base::dim_t world_dim = geo_ptr->DimGlobal();
  LF_ASSERT_MSG_CONSTEXPR(world_dim == 2, "Only available for flat domains");
  // Gram determinant at quadrature points
  const Eigen::VectorXd determinants(
      geo_ptr->IntegrationElement(pfe.Qr().Points()));
  LF_ASSERT_MSG(
      determinants.size() == pfe.Qr().NumPoints(),
      "Mismatch " << determinants.size() << " <-> " << pfe.Qr().NumPoints());
  // Fetch the transformation matrices for the gradients
  const Eigen::MatrixXd JinvT(
      geo_ptr->JacobianInverseGramian(pfe.Qr().Points()));
  LF_ASSERT_MSG(
      JinvT.cols() == 2 * pfe.Qr().NumPoints(),
      "Mismatch " << JinvT.cols() << " <-> " << 2 * pfe.Qr().NumPoints());
  LF_ASSERT_MSG(JinvT.rows() == world_dim,
                "Mismatch " << JinvT.rows() << " <-> " << world_dim);

  // Get the velocity vectors at quadrature points in the cell
  auto veloval = velo_(cell, pfe.Qr().Points());

  // Element matrix
  ElemMat mat(pfe.NumRefShapeFunctions(), pfe.NumRefShapeFunctions());
  mat.setZero();

  // Loop over quadrature points
  for (lf::base::size_type k = 0; k < pfe.Qr().NumPoints(); ++k) {
    const double w = pfe.Qr().Weights()[k] * determinants[k];
```

```
  // Matrix G whose columns contain transformed gradients of all local
```