C++ code 3.9.20: Implementation of `computeL2Deviation()` ↬ GITHUB

```
double computeL2Deviation(const lf::assemble::DofHandler &scal_dofh,
                          const Eigen::VectorXd &eta,
                          const lf::assemble::DofHandler &vec_dofh,
                          const Eigen::VectorXd &gamma) {
  double deviation_norm_value = 0.0;  // For retrurning the result
  // Obtain shared_ptr to mesh
  auto mesh_p = scal_dofh.Mesh();
  // Cell-oriented computation of deviation norm (squared)
  for (const lf::mesh::Entity *cell : mesh_p->Entities(0)) {
    LF_VERIFY_MSG(cell->RefEl() == lf::base::RefEl::kTria(),
                  "Unsupported cell type " << cell->RefEl());
    // Obtain area of the triangular cell
    const double area = lf::geometry::Volume(*(cell->Geometry()));

    // Obtain global scalar-FE indices of the vertices
    auto scal_dof_idx_vec = scal_dofh.GlobalDofIndices(*cell);
    // Obtain the gradients of the barycentric coordinates functions
    Eigen::Matrix<double, 2, 3> elgrad_Mat = gradbarycoordinates(*cell);
// Compute the gradient of the passed coefficient vector eta
    const Eigen::Vector2d grad_eta =
        elgrad_Mat.col(0) * eta(scal_dof_idx_vec[0]) +
        elgrad_Mat.col(1) * eta(scal_dof_idx_vec[1]) +
        elgrad_Mat.col(2) * eta(scal_dof_idx_vec[2]);

    // Obtaining the values of the passed vector at each node
    std::vector<Eigen::Vector2d> r_vec_values;
    for (const lf::mesh::Entity *node : cell->SubEntities(2)) {
      LF_VERIFY_MSG(node->RefEl() == lf::base::RefEl::kPoint(),
                    "Expected kPoint type!" << node->RefEl());
      auto vec_dofh_idx = vec_dofh.GlobalDofIndices(*node);
      Eigen::Vector2d r_vec_at_node;
      r_vec_at_node[0] = gamma[vec_dofh_idx[0]];
      r_vec_at_node[1] = gamma[vec_dofh_idx[1]];
      r_vec_values.push_back(r_vec_at_node);
    }

    // Computing local contribution of the cell to the deviation norm
    double local_norm_value =
        (0.5 * (r_vec_values.at(0) + r_vec_values.at(1)) - grad_eta)
            .squaredNorm() +
        (0.5 * (r_vec_values.at(1) + r_vec_values.at(2)) - grad_eta)
            .squaredNorm() +
        (0.5 * (r_vec_values.at(2) + r_vec_values.at(0)) - grad_eta)
            .squaredNorm();
    local_norm_value *= area / 3.0;

    // Adding local contribution to the value of the deviation norm
    deviation_norm_value += local_norm_value;
  }
  return std::sqrt(deviation_norm_value);
};  // computeL2Deviation
```