C++ code 3.13.10: Alternative Implementation of `contactFlux()` ⇝ GitLab

```cpp
template <typename SIGMAFUNCTION>
double contactFluxMF(
    std::shared_ptr<const lf::uscalfe::FeSpaceLagrangeO1<double>> fe_space,
    const Eigen::VectorXd &sol_vec, SIGMAFUNCTION &&sigma,
    const lf::mesh::utils::CodimMeshDataSet<int> &edgeids, int contact_id = 0) {
  // The underlying finite element mesh
  const lf::mesh::Mesh &mesh{*(fe_space->Mesh())};
  // Variable for summing boundary flux
  double s = 0.0;
  // Counter for edges on selected contact
  unsigned int ed_cnt = 0;
  // Compute exterior edge-weighted normals
  lf::mesh::utils::CodimMeshDataSet<Eigen::Vector2d> normals{
      exteriorEdgeWeightedNormals(fe_space->Mesh())};
  // Build a MeshFunction representing the gradient of the finite element
  // solution
  const lf::fe::MeshFunctionGradFE mf_grad(fe_space, sol_vec);
  // Reference coordinates of edge midpoints of a triangle
  const Eigen::MatrixXd mp_refc{
      (Eigen::Matrix<double, 2, 3>() << 0.5, 0.5, 0.0, 0.0, 0.5, 0.5)
          .finished()};
  // Loop over all cells
  for (const lf::mesh::Entity *cell : mesh.Entities(0)) {
    const lf::base::RefEl ref_el_type{cell->RefEl()};
    LF_ASSERT_MSG(ref_el_type == lf::base::RefEl::kTria(),
                  "contactFlux: implemented for triangles only");
    // Obtain array of edge pointers (sub-entities of co-dimension 1)
    std::span<const lf::mesh::Entity *const> sub_ent_range{
        cell->SubEntities(1)};
    // Must be three edges
    LF_ASSERT_MSG(sub_ent_range.size() == 3, "Triangle must have three edges!");
    // Obtain gradient values at midpoints of edges
    const std::vector<Eigen::VectorXd> grad_at_mp{mf_grad(*cell, mp_refc)};
    // Visit edges, check flags, and add contribution to flux integral
    for (lf::base::sub_idx_t j = 0; j < ref_el_type.NumSubEntities(1); ++j) {
      const lf::mesh::Entity &edge{*sub_ent_range[j]};
      LF_ASSERT_MSG(edge.RefEl() == lf::base::RefEl::kSegment(),
                    "Not an edge!");
      if (edgeids(edge) == contact_id) {
        // Exterior normal vector
        LF_ASSERT_MSG(normals.DefinedOn(edge),
                      "Normal vector not available for " << edge);
        const Eigen::Vector2d n{normals(edge)};
        LF_ASSERT_MSG(n.norm() > 0, " zero normal vector ?");
        const auto sigma_val{sigma((cell->Geometry())->Global(mp_refc.col(j)))};
        s += n.dot(sigma_val * grad_at_mp[j]);
        ed_cnt++;
      }
    }  // end loop over edges
  }    // end loop over cells
  std::cout << "Summed flux for " << ed_cnt << " edges." << std::endl;
  return s;
}  // end contactFluxMF
```