C++ code 11.8.22: Sub-problem (11-8.m): function `solveAdvection2D()` ⇢ GitLab

```cpp
std::shared_ptr<
    lf::mesh::utils::CodimMeshDataSet<Eigen::Matrix<double, 2, Eigen::Dynamic>>>
computeCellNormals(std::shared_ptr<const lf::mesh::Mesh> mesh_p) {
  // Appears only in mastersolution

  // Initialize datastruture for the result
  lf::mesh::utils::CodimMeshDataSet<Eigen::Matrix<double, 2, Eigen::Dynamic>>
      result(mesh_p, 0);

  // Compute normal vectors
  for (const lf::mesh::Entity *cell : mesh_p->Entities(0)) {
    const lf::geometry::Geometry *geo_p = cell->Geometry();
    const Eigen::MatrixXd corners = lf::geometry::Corners(*geo_p);

    if (corners.cols() == 3) {
      Eigen::Matrix<double, 2, Eigen::Dynamic> normal_vectors(2, 3);

      // Compute normal vectors
      Eigen::Matrix<double, 2, 3> bary_coord = gradbarycoordinates(corners);

      // Reorder computed normal vectors
      // normal_vectors[0] -> normal_vector of edge[0]
      normal_vectors.col(0) = -(bary_coord.col(2)).normalized();
      normal_vectors.col(1) = -(bary_coord.col(0)).normalized();
      normal_vectors.col(2) = -(bary_coord.col(1)).normalized();

      // Save the matrix in our datastructure
      result(*cell) = normal_vectors;
    } else {  // corners.cols() == 4
      Eigen::Matrix<double, 2, Eigen::Dynamic> normal_vectors(2, 4);

      // Split the quadrilateral into two triangles (1,2,3) and (1,3,4)
      Eigen::Matrix<double, 2, 3> tria_1;
      Eigen::Matrix<double, 2, 3> tria_2;
      tria_1 << corners.col(0), corners.col(1), corners.col(2);
      tria_2 << corners.col(0), corners.col(2), corners.col(3);

      // Compute normal vectors
      Eigen::Matrix<double, 2, 3> bary_coord_1 = gradbarycoordinates(tria_1);
      Eigen::Matrix<double, 2, 3> bary_coord_2 = gradbarycoordinates(tria_2);

      // Reorder computed normal vectors
      // normal_vectors[0] -> normal_vector of edge[0]
      normal_vectors.col(0) = -(bary_coord_1.col(2)).normalized();
      normal_vectors.col(1) = -(bary_coord_1.col(0)).normalized();
      normal_vectors.col(2) = -(bary_coord_2.col(0)).normalized();
      normal_vectors.col(3) = -(bary_coord_2.col(1)).normalized();

      // Save the matrix in our datastructure
      result(*cell) = normal_vectors;
    }
  }
  return std::make_shared<lf::mesh::utils::CodimMeshDataSet<
      Eigen::Matrix<double, 2, Eigen::Dynamic>>>(result);

}
```