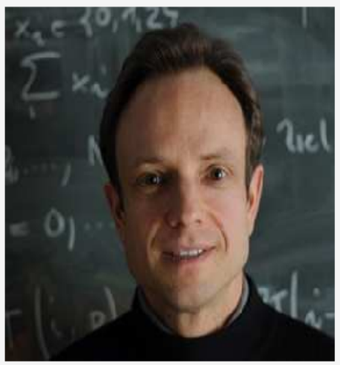
ETH Lecture 401-0674-00L Numerical Methods for Partial Differential Equations

Homework Problem Video Tutorial

Problem 2-7: Computing the length of the boundary in LEHRFEM++

Prof. R. Hiptmair, SAM, ETH Zurich

Date: March 2, 2019



▷ Relies on Section 2.7.2.

a, Compute volume of meshed domain

→ 

C++11 code 2.7.1: Computation of edge-node incidence matrix G ⇝ GITLAB

```
double volumeOfDomain(const std::shared_ptr<lf::mesh::Mesh> mesh) {
  double volume = 0.0;
  /* BEGIN_SOLUTION */
  // iterate over all cells (co-dimension = 0)
  for (const lf::mesh::Entity &ent : mesh->Entities(0)) {
    lf::geometry::Geometry *geo_ptr = ent.Geometry();
    volume += lf::geometry::Volume(*geo_ptr);
  }
  /* END_SOLUTION */
  return volume;
}
```

b, LF++ - way of flagging entities on $\partial\Omega$

edge $\in \partial\Omega$ ⇔ 1 adjacent cell

node $\in \partial\Omega$ ⇔ endpoint of an edge $\in \partial\Omega$

(2)

Study the description and the implementation of the LEHRFEM++ function (↪ GITHUB)

```
CodimMeshDataSet< bool >
lf::mesh::utils::flagEntitiesOnBoundary(const std::shared_ptr< const
   Mesh > & mesh_p,
  lf::base::dim_t codim);
```

and explain how edges on the boundary are identified in this function.

C++11 code 2.7.2: (Skeleton) ilmplementation of the LEHRFEM++ function `flagEntitiesOnBoundary()`

```
CodimMeshDataSet<bool> flagEntitiesOnBoundary(
    const std::shared_ptr<const Mesh>& mesh_p, lf::base::dim_t codim) {
  // count cells adjacent to entities of co-dimension 1
  CodimMeshDataSet<lf::base::size_type> no_adjacent_cells{
      countNoSuperEntities(mesh_p, 1, 1)};
  // flag array
  CodimMeshDataSet<bool> bd_flags{mesh_p, codim, false};
  // relative codimension with respect to faces (entities of
     co-dimension 1)
  const lf::base::dim_t rel_codim = codim - 1;
  // Run through faces and flag sub-entities
  for (const lf::mesh::Entity& edge : mesh_p->Entities(1)) {
    if (no_adjacent_cells(edge) == 1) {
      // Boundary face detected!
      // Traverse all sub-entities of a specific relative co-dimension
      for (const lf::mesh::Entity& subent : edge.SubEntities(rel_codim)) {
        bd_flags(subent) = true;
      }
    }
  }
  return bd_flags;
}
```

→ creates an array storing the number of adjacent cells for edge

If codim == 0 → nodes

C++11 code 2.7.3: (Skeleton) Implementation of the LEHRFEM++ function `countNoSuperEntities()`

```
CodimMeshDataSet<lf::base::size_type> countNoSuperEntities(
    const std::shared_ptr<const Mesh>& mesh_p, lf::base::dim_t codim_sub,
    lf::base::dim_t codim_super) {
  // Declare and initialize the data set
  CodimMeshDataSet<lf::base::size_type> sup_ent_cnt{mesh_p, codim_sub, 0};

  const lf::base::dim_t super_codim = codim_sub - codim_super;
  // Run through all super entities
  for (const lf::mesh::Entity& e : mesh_p->Entities(super_codim)) {
    // Traverse all sub-entities of a specific relative co-dimension
    for (const lf::mesh::Entity& subent : e.SubEntities(codim_super)) {
      sup_ent_cnt(subent) += 1;
    }
  }
  return sup_ent_cnt;
}
```

(c)

Implement a function

```
double lengthOfBoundary(const lf::mesh::Mesh &mesh);
```

that calculates the length of the boundary of the domain covered by the mesh passed through the `mesh` object.

③

C++11 code 2.7.4: Computation of edge-node incidence matrix G ↪ GITLAB

```
double lengthOfBoundary(const std::shared_ptr<lf::mesh::Mesh> mesh) {
  double length = 0.0;
  /* BEGIN_SOLUTION */
  // This function returns an array of flags
  // a flag is true if an edge is part of the boundary
  auto edge_marker = lf::mesh::utils::flagEntitiesOnBoundary(mesh, 1);

  // iterate over all edges (co-dimension = 1)
  for (const lf::mesh::Entity &ent : mesh->Entities(1)) {
    // check if edge is part of the boundary
    if (edge_marker(ent)) {
      lf::geometry::Geometry *geo_ptr = ent.Geometry();
      length += lf::geometry::Volume(*geo_ptr);
    }
  }
  /* END_SOLUTION */
  return length;
}
```

d, Reading mesh & computing area, 12.21