SOLUTION of (12-1.q):

We already know the normal vector at those parts of the boundary, where the velocity does not vanish:

$$\text{On} \quad \Gamma_{\text{in}} \quad \boldsymbol{n}(\boldsymbol{x}) = \begin{bmatrix} -1 \\ 0 \end{bmatrix} \quad , \quad \text{on} \quad \Gamma_{\text{out}} \quad \boldsymbol{n}(\boldsymbol{x}) = \begin{bmatrix} 1 \\ 0 \end{bmatrix} .$$

So we have to add up integrals

$$\pm \int_e (\mathbf{v}_h)_1(\boldsymbol{x}) p_h(\boldsymbol{x}) \, \mathrm{d}S(\boldsymbol{x}) \, , \quad e \mathrel{\hat{=}} \text{mesh edge on } \Gamma_* \, .$$

If the two values of p_h in the endpoints of the edge e are denoted by p_0, p_1, and the values of $(\mathbf{v}_h)_1$ at the endpoints and in the midpoint are v_0, v_1, v_m, then

$$\int_e (\mathbf{v}_h)_1(\boldsymbol{x}) p_h(\boldsymbol{x}) \, \mathrm{d}S(\boldsymbol{x}) = \begin{bmatrix} p_0 & p_1 \end{bmatrix} \begin{bmatrix} \frac{1}{6} & \frac{1}{3} & 0 \\ 0 & \frac{1}{3} & \frac{1}{6} \end{bmatrix} \begin{bmatrix} v_0 \\ v_m \\ v_1 \end{bmatrix} . \tag{12.1.41}$$

The values p_0, p_1, v_0, v_1, v_m can directly be accessed as basis expansion coefficients through the global indices of the local shape functions *covering* the edge e.

C++ code 12.1.42: Sub-problem (12-1.q): Boundary integral formula for dissipated power
➥ GitLab

```
2  double compDissPowBd(const lf::assemble::DofHandler& dofh,
3                       const Eigen::VectorXd& muvec, bool print) {
4    // First fetch underlying mesh
5    const lf::mesh::Mesh& mesh{*dofh.Mesh()};
6    // Summation variable
7    double p_diss{0.0};
8  // Loop over the edges and check whether they are located on the inlet
       x_1 = 0
9  // or outlet x_1 = 1.
10   const Eigen::Matrix<double, 2, 3> M{
11       (Eigen::Matrix<double, 2, 3>(2, 3) << 1.0 / 6.0, 1.0 / 3.0, 0.0, 0.0,
12        1.0 / 3.0, 1.0 / 6.0)
13           .finished()};
14   for (const lf::mesh::Entity* edge : mesh.Entities(1)) {
15     // Length of edge
16     const double length = lf::geometry::Volume(*(edge->Geometry()));
17     const Eigen::MatrixXd endpoints{Corners(*(edge->Geometry()))};
18     const Eigen::Vector2d mp{0.5 * (endpoints.col(0) + endpoints.col(1))};
19     // Check by location whether the edge is on the inlet or outlet
20     int locflag = 0;
21     if (mp[0] < 1E-8) {
22       // On inlet boundary
23       locflag = -1;
24     } else if (mp[0] > 1 - 1E-8) {
25       locflag = 1;
26     }
27     if (locflag != 0) {
28       // Obtain indices of global shape functions associated with the
           edge
29       std::span<const lf::assemble::gdof_idx_t> dof_idx{
30           dofh.GlobalDofIndices(*edge)};
```

```
31        LF_ASSERT_MSG(dof_idx.size() == 8,
32                      "For TH FEM an edge should be covered by 8 GSFs");
33        // The pressure dofs sit in the nodes as third node dof.
34        // These are edge-local shape functions #2 and #5 (C++ indexing)
35        Eigen::Vector2d p_ldof{muvec[dof_idx[2]], muvec[dof_idx[5]]};
36        // The velocity x-component dofs sit in nodes and edges and are
              numbered
37        // first there.  Their edge-local numbers are #0, #3, and #6
38        Eigen::Vector3d vx_ldof{muvec[dof_idx[0]], muvec[dof_idx[6]],
39                                muvec[dof_idx[3]]};
40        // Evaluate the integral; can be done exactly, because the
              integrand is
41        // polynomial
42        const double loc_diss = locflag * length * p_ldof.dot(M * vx_ldof);
43        if (print) {
44          std::cout << "DPB: Edge with midpoint [" << mp.transpose()
45                    << "] : lf = " << locflag
46                    << ", * pvals = " << p_ldof.transpose()
47                    << ", vx_ldof = " << vx_ldof.transpose()
48                    << ", loc_diss = " << loc_diss << std::endl;
49        }
50        p_diss += loc_diss;
51      }
52    }
53    return p_diss;
54  }
```