

SOLUTION of (13-1.q):

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

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

So we have to add up integrals

$$\pm \int_e (\mathbf{v}_h)_1(\mathbf{x}) p_h(\mathbf{x}) \, dS(\mathbf{x}), \quad e \triangleq \text{mesh edge on } \Gamma_*.$$

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

$$\int_e (\mathbf{v}_h)_1(\mathbf{x}) p_h(\mathbf{x}) \, dS(\mathbf{x}) = [p_0 \quad p_1] \begin{bmatrix} \frac{1}{6} & \frac{1}{3} & 0 \\ 0 & \frac{1}{3} & \frac{1}{6} \end{bmatrix} \begin{bmatrix} v_0 \\ v_m \\ v_1 \end{bmatrix}. \quad (13.1.41)$$

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 13.1.42: Sub-problem (13-1.q): Boundary integral formula for dissipated power

→ [GitLab](#)

```
2 double compDissPowBd(const If::assemble::DofHandler& dofh,
3                     const Eigen::VectorXd& muvec, bool print) {
4     // First fetch underlying mesh
5     const If::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
9     // or outlet  $x_1 = 0$ 
10    // or outlet  $x_1 = 1$ .
11    const Eigen::Matrix<double, 2, 3> M{
12        (Eigen::Matrix<double, 2, 3>(2, 3) << 1.0 / 6.0, 1.0 / 3.0, 0.0, 0.0,
13         1.0 / 3.0, 1.0 / 6.0)
14        .finished()};
15    for (const If::mesh::Entity* edge : mesh.Entities(1)) {
16        // Length of edge
17        const double length = If::geometry::Volume(*(edge->Geometry()));
18        const Eigen::MatrixXd endpoints{Corners(*(edge->Geometry()))};
19        const Eigen::Vector2d mp{0.5 * (endpoints.col(0) + endpoints.col(1))};
20        // Check by location whether the edge is on the inlet or outlet
21        int locflag = 0;
22        if (mp[0] < 1E-8) {
23            // On inlet boundary
24            locflag = -1;
25        } else if (mp[0] > 1 - 1E-8) {
26            locflag = 1;
27        }
28        if (locflag != 0) {
29            // Obtain indices of global shape functions associated with the
30            // edge
31            std::span<const If::assemble::gdof_idx_t> dof_idx{
32                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
37 // numbered
38 // first there. Their edge-local numbers are #0, #3, and #6
39 Eigen::Vector3d vx_ldof{muvec[dof_idx[0]], muvec[dof_idx[6]],
40                        muvec[dof_idx[3]]};
41 // Evaluate the integral; can be done exactly, because the
42 // integrand is
43 // polynomial
44 const double loc_diss = locflag * length * p_ldof.dot(M * vx_ldof);
45 if (print) {
46     std::cout << "DPB: Edge with midpoint [" << mp.transpose()
47               << "]" : lf = " << locflag
48               << ", * pvals = " << p_ldof.transpose()
49               << ", vx_ldof = " << vx_ldof.transpose()
50               << ", loc_diss = " << loc_diss << std::endl;
51 }
52 p_diss += loc_diss;
53 }
54 return p_diss;
55 }

```