ETH Lecture 401-0663-00L Numerical Methods for PDEs

Midterm Exam

Spring Term 2019

April 1, 2019, 15:15, HG F 1

Don't panic!

Family Name		%
First Name		
Department		
Legi Nr.		
Date	April 1, 2019	

Points:

	1	2	3	4	Total
max	5	6	6	8	25
achvd					

- This is a **closed-book exam.**
- Keep only writing material and Legi on the table.
- Keep mobile phones, tablets, smartwatches, etc. turned off in your bag.
- Fill in this cover sheet first.
- Turn the cover sheet only when instructed to do so.
- Then write your name and Legi Nr. on each page.
- **Write your answers in the appropriate fields on these problem sheets.**
- **Wrong ticks in multiple-choice boxes will lead to points being subtracted.**
- **Anything written outside the answer boxes will not be taken into account.**
- Do not write with red/green color or with pencil.
- Make sure to hand in every sheet.
- Two blank pages at the end of the exam: space for notes
- **Duration: 30 minutes.**

Problem 0-1: Deducing the strong form of second-order elliptic boundary value problem

In [Lecture → Section 1.5] we have seen how to extract the strong (PDE) form of a boundary value problem from its variational (weak) form.

This is a purely theoretical problem. ▷ problem code folder: FromWeakToStrongFormOfBVP

On the unit disk $\Omega := \{x \in \mathbb{R}^2 : \|x\| < 1\}$ (with exterior unit normal vector field $\boldsymbol{n} : \partial\Omega \to \mathbb{R}^2$) we consider the variational problem: seek

$$u \in H^1(\Omega)\colon \quad \int_\Omega \mathbf{grad}\, u(x) \cdot \mathbf{grad}\, v(x)\, \mathrm{d}x = \int_\Omega v(x)\, \mathrm{d}x + \int_{\partial\Omega} v(x)\, \mathrm{d}S(x) \quad \forall v \in H^1(\Omega)\,. \tag{0.1.1}$$

(0-1.a) ⚀ (2 pts.) Write down the partial differential equation occurring in the strong form of (0.1.1)

$$\boxed{} = \boxed{} \quad \text{in} \quad \Omega\,.$$

Solution of (0-1.a):

Testing with $v \in C_0^\infty(\Omega)$ and integrating by parts we get

$$-\Delta u \ \left[= -\operatorname{div}\mathbf{grad}\, u = -\frac{\partial^2 u}{\partial {x_1}^2} - \frac{\partial^2 u}{\partial {x_2}^2} \right] = 1 \quad \text{in} \quad \Omega\,. \tag{0.1.2}$$

▲

(0-1.b) ⚁ (3 pts.) State the boundary conditions satisfied by a sufficiently smooth solution u of (0.1.1).

$$\boxed{} = \boxed{} \quad \text{on} \quad \partial\Omega\,.$$

Solution of (0-1.b):

Testing with $v \in C^\infty(\overline{\Omega})$, integrating by parts, and using the PDE (0.1.2) we get

$$\mathbf{grad}\, u \cdot \boldsymbol{n} = 1 \quad \text{on} \quad \partial\Omega\,, \tag{0.1.3}$$

where $\boldsymbol{n}$ is the exterior unit normal vector field at $\partial\Omega$. In fact, since Ω is the unit disk, we have $\boldsymbol{n}(x) = x$ for all $x \in \partial\Omega$, which allows to express (0.1.3) differently.

▲

End Problem 0-1 , 5 pts.

Problem 0-2: Cubic Lagrangian finite element space on 2D hybrid meshes

In [Lecture → Section 2.6] we learned about Lagrangian finite element spaces of piecewise polynomial continuous functions. We saw how the finite element basis functions (global shape functions) could be constructed as cardinal basis functions belonging to suitable sets of interpolation nodes.

This is a purely theoretical problem. ▷ problem code folder: CubicLagrangianFEM

We consider the cubic Lagrangian finite element space $\mathcal{S}_3^0(\mathcal{M})$ on a 2D hybrid mesh $\mathcal{M}$. The **local interpolation nodes** for $\mathcal{S}_3^0(\mathcal{M})$ on a triangle and a quadrilateral are drawn in Fig. 1 (left figure) for triangular cells and in Fig. 2 (right figure) for quadrilateral cells, respectively:

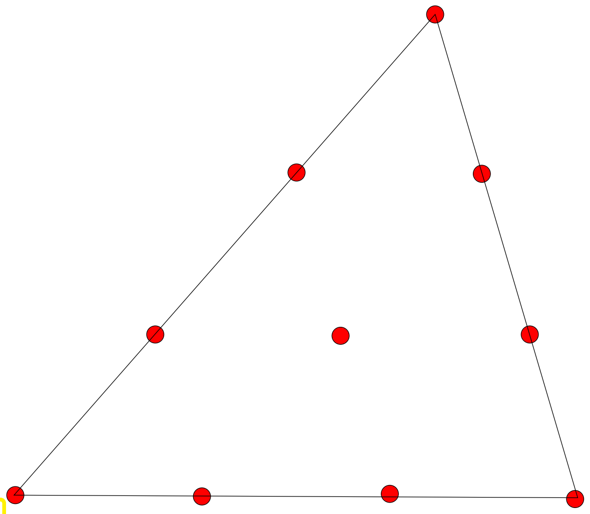


Fig. 1

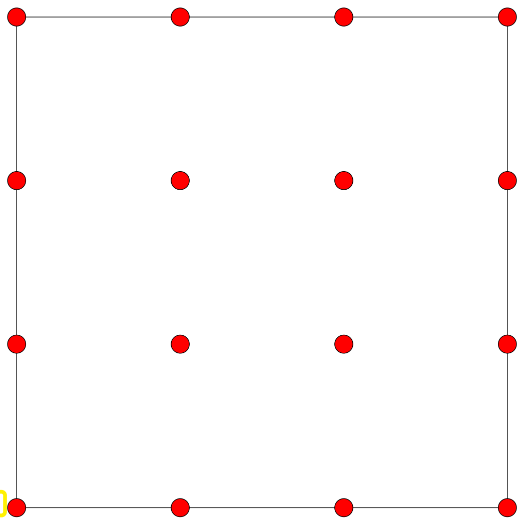


Fig. 2

(0-2.a) ⚁ (4 pts.) Complete the following LEHRFEM++-based C++ code for the initialization of a **lf::assemble::DofHandler**-type appropriate for $\mathcal{S}_3^0(\mathcal{M})$ on a 2D hybrid mesh.

HINT 1 for (0-2.a): The constructor of **lf::assemble::UniformFEDofHandler** has to be given information about how many finite element basis function are associated with each type of entity. ⌟

```
lf::assemble::UniformFEDofHandler dof_handler(
      mesh_p, {{lf::base::RefEl::kPoint(), [ ]},
               {lf::base::RefEl::kSegment(), [ ]},
               {lf::base::RefEl::kTria(), [ ]},
               {lf::base::RefEl::kQuad(), [ ]}});
```

SOLUTION of (0-2.a):

From the number of interpolation points in the *interior* of the respective entities we can read off the number of global/local shape functions associated with them:

```
lf::assemble::UniformFEDofHandler dof_handler(
      mesh_p, {{lf::base::RefEl::kPoint(), 1},
               {lf::base::RefEl::kSegment(), 2},
               {lf::base::RefEl::kTria(), 1},
               {lf::base::RefEl::kQuad(), 4}});
```

▲

(0-2.b) ⚁ (2 pts.) Give a formula for $\dim \mathcal{S}_3^0(\mathcal{M})$ in terms of the numbers N_V, N_E, N_T, and N_Q of NODEs, EDGEs, TRIAs, and QUADs of a 2D hybrid mesh $\mathcal{M}$:

$$\dim \mathcal{S}_3^0(\mathcal{M}) = \boxed{} .$$

SOLUTION of (0-2.b):

We just have to add up the number of global shape functions associated with the respective entity types:

$$\dim \mathcal{S}_3^0(\mathcal{M}) = N_V + 2N_E + N_T + 4N_Q .$$

▲

End Problem 0-2 , 6 pts.

Problem 0-3: Operating locally on Galerkin matrices

In this problem we connect assembly with local operations on Galerkin matrices.

A purely theoretical problem based on the material of [Lecture → Section 2.7.4] ▷ problem code folder: GalerkinMatVecMult

We examine the following LEHRFEM++-based C++ function:

C++11 code 0.3.1: A function for a local operation of Galerkin matrices

```
1  template <typename SCALAR, typename ENTITY_MATRIX_PROVIDER>
2  SCALAR multVecAssMat(lf::assemble::dim_t codim,
3                       const lf::assemble::DofHandler &dofh,
4                       ENTITY_MATRIX_PROVIDER &entity_matrix_provider,
5                       Eigen::Matrix<SCALAR, Eigen::Dynamic, 1> &vec) {
6    LF_ASSERT_MSG(dofh.NoDofs() == vec.size(),
7                  "NoDof mismatch " << dofh.NoDofs() << " <-> " << vec.size());
8    auto mesh = dofh.Mesh();
9    SCALAR s{};
10   for (const lf::mesh::Entity &entity : mesh->Entities(codim)) {
11     if (entity_matrix_provider.isActive(entity)) {
12       const lf::assemble::size_type elmat_dim = dofh.NoLocalDofs(entity);
13       lf::base::RandomAccessRange<const gdof_idx_t> global_idx(
14           dofh.GlobalDofIndices(entity));
15       const auto elem_mat{entity_matrix_provider.Eval(entity)};
16       Eigen::Matrix<SCALAR, Eigen::Dynamic, 1> locvec(elmat_dim);
17       for (int l = 0; l < elmat_dim; ++l) {
18         locvec[l] = vec[global_idx[l]];
19       }
20       s += locvec.dot(elem_mat * locvec);
21     }
22   }
23   return s;
24 }
```

The argument `dofh` passes a **lf::assemble::DofHandler** object providing information about the mesh and the local-to-global index mapping for the finite element space. The other argument `vec` contains a finite element basis expansion coefficient vector.

(0-3.a) (6 pts.) Supplement Line 11 of the code of the following C++ function so that it can be used as a unit test for assembly in LEHRFEM++:

C++11 code 0.3.2: Auxiliary function for unit testing assembly in LEHRFEM++

```
1  void testAssembly(const lf::mesh::Mesh &mesh,
2                    const lf::assemble::DofHandler &dof_handler) {
3    const lf::assemble::size_type N_dofs(dof_handler.NoDofs());
4    Eigen::VectorXd vec = Eigen::VectorXd::Random(N_dofs);
5    // Create object compliant with ENTITY_MATRIX_PROVIDER
6    TestEntityMatrixProvider assembler{mesh};
7    lf::assemble::COOMatrix<double> mat(N_dofs, N_dofs);
8    mat = lf::assemble::AssembleMatrixLocally<lf::assemble::COOMatrix<double>>(
9        0, dof_handler, assembler);
10   Eigen::SparseMatrix<double> A = mat.makeSparse();
11   double s1 =                                            ;
12   double s2 = multVecAssMat<double, decltype(assembler)>(0, dof_handler,
                                                   assembler, vec);
   EXPECT_NEAR(s1, s2, 1.0E−9);
}
```

SOLUTION of (0-3.a):

The function `multVecAssMat()` multiplies the square finite element Galerkin matrix $\mathbf{A}$ encoded by `dofh` and `entity_matrix_provider` with the vector $\vec{v}$ passed in `vec` from left and right:

$$\texttt{multVecAssMat(...)} = \vec{v}^{\top}\mathbf{A}\vec{v}\,.$$

Hence the missing line can be

```
double s1 = vec.dot(A * vec);
```

or (because we compute in $\mathbb{R}$)

```
double s1 = vec.transpose()*A * vec;
```

▲

End Problem 0-3 , 6 pts.

Problem 0-4: Linear finite element Galerkin discretization of 1D transport equation

This exercise practises the computation of local element matrices and global Galerkin matrices for linear Lagrangian finite elements in 1D, for a "non-standard" bilinear form, however.

This is a purely theoretical problem related to [Lecture → Section 2.3] and [Lecture → Section 2.5.3].

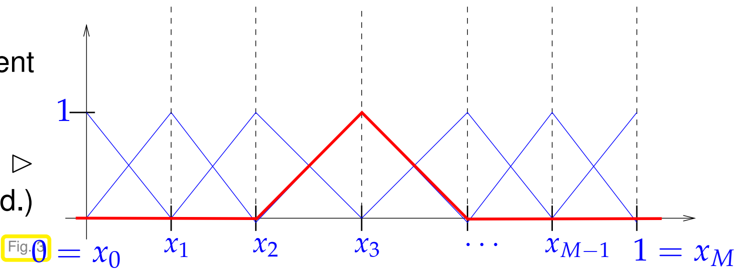
We consider the bilinear form

$$\mathsf{b}(u,v) := \int_0^1 \frac{du}{dx}(x)\, v(x)\, \mathrm{d}x\,, \quad u,v \in H^1(]0,1[)\,, \tag{0.4.1}$$

and its Galerkin finite element discretization based on the space $\mathcal{S}_1^0(\mathcal{M})$ of $\mathcal{M}$-piecewise linear continuous functions on an *equidistant* mesh $\mathcal{M}$ of $]0,1[$ with M cells, $M \in \mathbb{N}$.

The standard finite element basis functions ("tent functions") are used.

The basis functions of $\mathcal{S}_1^0(\mathcal{M})$ in 1D ▷
(The basis function associated with x_3 is highlighted.)



(0-4.a) ⚃ (4 pts.) Compute the entries of the element matrix $\mathbf{B}_K$ for the cell $K :=]0, 1/M[$ of the mesh and write their values in the boxes:

$$\mathbf{B}_K = \begin{bmatrix} \boxed{} & \boxed{} \\ \boxed{} & \boxed{} \end{bmatrix}.$$

SOLUTION of (0-4.a):

The element matrices are the same for all cells of the mesh. On $K :=]0,h[$, $h := M^{-1}$, the local shape functions are

$$b_K^1(x) = 1 - \frac{x}{h} \quad , \quad b_K^2(x) = \frac{x}{h}\,.$$

Their derivatives are constant:

$$\frac{db_K^1}{dx}(x) = -\frac{1}{h} \quad , \quad \frac{db_K^2}{dx}(x) = \frac{1}{h}\,.$$

The element matrix is given by

$$\mathbf{B}_K \left[\int_0^h \frac{db_K^j}{dx}(x) b_K^i(x)\right]_{i,j=1}^2 = \begin{bmatrix} -\frac{1}{2} & \frac{1}{2} \\ -\frac{1}{2} & \frac{1}{2} \end{bmatrix},$$

by direct computation. The row sums of the element matrix must vanish, because $\mathsf{b}(x \mapsto 1, v) = 0$ for all $v \in H^1(]0,1[)$.

▲

(0-4.b) (4 pts.) For $M = 3$ write down the full $\mathcal{S}_1^0(\mathcal{M})$-Galerkin matrix $\mathbf{B}$ for $\mathsf{b}(\cdot,\cdot)$ by filling the boxes:

$$\mathbf{B} = \begin{bmatrix} \boxed{} & \boxed{} & \boxed{} & \boxed{} \\ \boxed{} & \boxed{} & \boxed{} & \boxed{} \\ \boxed{} & \boxed{} & \boxed{} & \boxed{} \\ \boxed{} & \boxed{} & \boxed{} & \boxed{} \end{bmatrix}.$$

SOLUTION of (0-4.b):

By direct assembly from the element matrices, which are the same for all the cells:

$$\mathbf{A} = \begin{bmatrix} -\frac{1}{2} & \frac{1}{2} & 0 & 0 \\ -\frac{1}{2} & 0 & \frac{1}{2} & 0 \\ 0 & -\frac{1}{2} & 0 & \frac{1}{2} \\ 0 & 0 & -\frac{1}{2} & \frac{1}{2} \end{bmatrix}.$$

Note that all row sums must vanish. Moreover, the zero off-diagonal entries correspond to cases where the supports of the tent functions have no overlap.

▲

End Problem 0-4 , 8 pts.

Scratch space (will not be evaluated)

Scratch space (will not be evaluated)