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

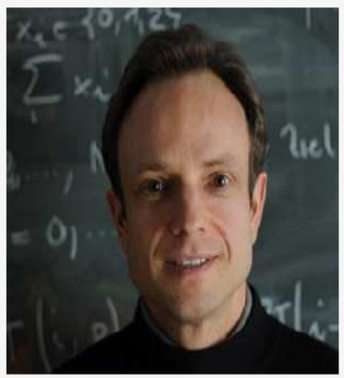
Homework Problem Video Tutorial

Problem 2-8: Introduction to local assembly in LEHRFEM++

Prof. R. Hiptmair, SAM, ETH Zurich

Date: March 2, 2019

(C) Seminar für Angewandte Mathematik, ETH Zürich



Based on Section 2.7

a,

b,

LF++ ENTITY_*_PROVIDERS :

Eval(const Entity &)

c,

C++11 code 2.8.1: Sub-problem (2-8.c): Implementation of `solve()` → GITLAB

```
template <class ELMAT_BUILDER, class ELVEC_BUILDER>
Eigen::VectorXd solve(ELMAT_BUILDER& elmat_builder,
                      ELVEC_BUILDER& elvec_builder) {
  // Use one of LehrFEM++'s default meshes.  Try different meshes by
     changing the
  // function index parameter
  /* std::shared_ptr<lf::mesh::Mesh> mesh_p =
   *    lf::mesh::test_utils::GenerateHybrid2DTestMesh(0, 1.0 / 3.0); */
  auto mesh_p = Generate2DTestMesh();
  // We use a linear Lagrangian FE space
  auto fe_space =
      std::make_shared<lf::uscalfe::FeSpaceLagrangeO1<double>>(mesh_p);
  // Reference to current mesh, obtained from the FE space
  const lf::mesh::Mesh& mesh{*(fe_space->Mesh())};
  // Obtain local->global index mapping for current finite element
     space
  const lf::assemble::DofHandler& dofh{fe_space->LocGlobMap()};
  // Dimension of finite element space`
  const lf::base::size_type N_dofs(dofh.NoDofs());

  // Matrix in triplet format holding Galerkin matrix, zero initially.
  lf::assemble::COOMatrix<double> A(N_dofs, N_dofs);
  // Invoke assembly on cells (co-dimension = 0).  The element matrix
     builder is
  // passed as an argument
  lf::assemble::AssembleMatrixLocally(0, dofh, dofh, elmat_builder, A);
  Eigen::SparseMatrix<double> A_crs = A.makeSparse();

  // Right-hand side vector; has to be set to zero initially
  Eigen::Matrix<double, Eigen::Dynamic, 1> phi(N_dofs);
  phi.setZero();
  // Invoke assembly on cells (codim == 0).  The element vector builder
     is passed
  // as an argument
  AssembleVectorLocally(0, dofh, elvec_builder, phi);

  // Define solution vector
  Eigen::VectorXd sol_vec = Eigen::VectorXd::Zero(N_dofs);

  /* BEGIN_SOLUTION */
  // Solve linear system using Eigen's sparse direct elimination
  Eigen::SparseLU<Eigen::SparseMatrix<double>> solver;
  solver.compute(A_crs);
  if (solver.info() != Eigen::Success) {
    throw std::runtime_error("Could not decompose the matrix");
  }
  sol_vec = solver.solve(phi);
  /* END_SOLUTION */
```

(2)

d, Neumann BVP : $-\Delta u = f$ in Ω, $\operatorname{grad} u \cdot n = 0$

C++11 code 2.8.3: Sub-problem (2-8.d): Implementation of `solvePoissonBVP()`
➙ GITLAB

```
Eigen::VectorXd solvePoissonBVP() {
  // Convert tPoissonda function f to a LehrFEM++ mesh function object
  lf::uscalfe::MeshFunctionGlobal mf_f{f};

  // Define the solution vector
  Eigen::VectorXd solution = Eigen::VectorXd::Zero(1);

  /* BEGIN_SOLUTION */
  // Define the element matrix and element vector builders and solve
     the system
  lf::uscalfe::LinearFELaplaceElementMatrix elmat_builder;
  lf::uscalfe::LinearFELocalLoadVector<double, decltype(mf_f)> elvec_builder(
      mf_f);

  std::cout << "==================" << std::endl;
  std::cout << "solvePoissonBVP" << std::endl;

  solution = solve(elmat_builder, elvec_builder);
  /* END_SOLUTION */

  return solution;
}
```

```
==================
solvePoissonBVP
Error of solver: Absolute error = 31.6288, relative error = 1.24895
Norms of FE solution: L2-norm = 9.58266e+16, H1-seminorm = 25.229
```

Gal. Matrix is singular for Neumann BVP!

kernel $= \begin{bmatrix} 1 \\ \vdots \\ 1 \end{bmatrix} \cdot \mathbb{R}$

f,

The problem haunting any finite element Galerkin discretization of (2.8.2) does not arise for the rather similar boundary value problem

$$-\Delta u + u = f \quad \text{in} \quad \Omega \quad , \quad \mathbf{grad}\, u \cdot \boldsymbol{n} = 0 \quad \text{on} \quad \partial\Omega \, , \tag{2.8.4}$$

also posed on the unit square $\Omega =]0,1[^2$. Explain why based on the weak (variational) formulation that you should derive first.

Weak form : $u \in H^1(\Omega)$

$$\underbrace{\int_\Omega \operatorname{grad} u \cdot \operatorname{grad} v + uv \, dx}_{\text{s.p.d.}} = \int f v \, dx \quad \forall v \in H^1(\Omega)$$

s.p.d. $\Rightarrow$ Gal. Mat. s.p.d.

g, For a triangle K deduce the formula for the 3×3 element matrix $\mathbf{M}_K$ related to the bilinear form

$$\mathrm{m}(u,v) := \int_\Omega u(\boldsymbol{x}) v(\boldsymbol{x}) \, \mathrm{d}\boldsymbol{x} \, , \quad u, v \in L^2(\Omega) \, , \tag{2.8.6}$$

and the finite element space $\mathcal{S}_1^0(\mathcal{M})$. This element matrix is also called the **element mass matrix**. Only the area $|K|$ of K will enter your formula.

③

$M_K = \left[\int_K \lambda_i \lambda_j \, dx \right]_{i,j=1}^{3}$, K triangle

Lemma 2.7.5.5. Integration of powers of barycentric coordinate functions

For any non-degenerate d-simplex K with barycentric coordinate functions $\lambda_1, \ldots, \lambda_{d+1}$ and exponents $\alpha_j \in \mathbb{N}$, $j = 1, \ldots, d+1$,

$$\int_K \lambda_1^{\alpha_1} \cdot \cdots \cdot \lambda_{d+1}^{\alpha_{d+1}} \, \mathrm{d}\boldsymbol{x} = d!|K| \, \frac{\alpha_1! \alpha_2! \cdot \cdots \cdot \alpha_{d+1}!}{(\alpha_1 + \alpha_2 + \cdots + \alpha_{d+1} + d)!} \quad \forall \boldsymbol{\alpha} \in \mathbb{N}_0^{d+1} . \tag{2.7.5.6}$$

$d = 2$,

$$\mathbf{M}_K = \frac{|K|}{12} \begin{bmatrix} 2 & 1 & 1 \\ 1 & 2 & 1 \\ 1 & 1 & 2 \end{bmatrix} , \tag{2.8.7}$$

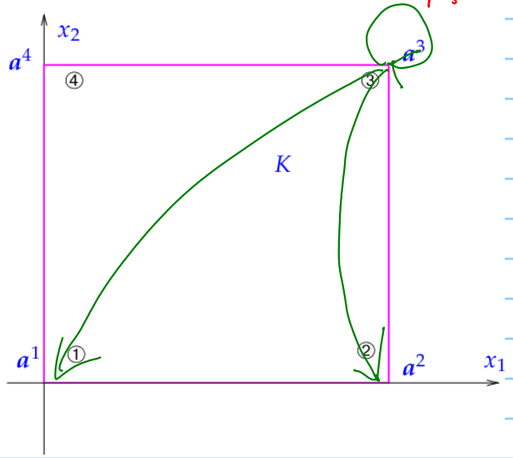
b) Element mass matrix for K

Again: $M_K = |K| \cdot M_{\hat{K}}$

$$b_K^1(\boldsymbol{x}) = (1 - x_1)(1 - x_2) ,$$
$$b_K^2(\boldsymbol{x}) = x_1(1 - x_2) ,$$
$$b_K^3(\boldsymbol{x}) = x_1 x_2 ,$$
$$b_K^4(\boldsymbol{x}) = (1 - x_1)x_2 .$$

(2.6.2.3)

► $b_K^i(\boldsymbol{a}^j) = \delta_{ij}$, $1 \le i, j \le 4$,



$$\mathbf{M}_K = \frac{|K|}{36} \begin{bmatrix} 4 & 2 & 1 & 2 \\ 2 & 4 & 2 & 1 \\ 1 & 2 & 4 & 2 \\ 2 & 1 & 2 & 4 \end{bmatrix} . \tag{2.8.8}$$

Trick: Use symmetry

a)

The problem haunting any finite element Galerkin discretization of (2.8.2) does not arise for the rather similar boundary value problem

$$-\Delta u + u = f \quad \text{in} \quad \Omega \quad , \quad \mathbf{grad}\, u \cdot \boldsymbol{n} = 0 \quad \text{on} \quad \partial\Omega , \tag{2.8.4}$$

also posed on the unit square $\Omega =]0,1[^2$. Explain why based on the weak (variational) formulation that you should derive first.

done

additional contribution

$$a(u,v) = \int_\Omega \mathrm{grad}\, u \cdot \mathrm{grad}\, u + u v \, dx$$

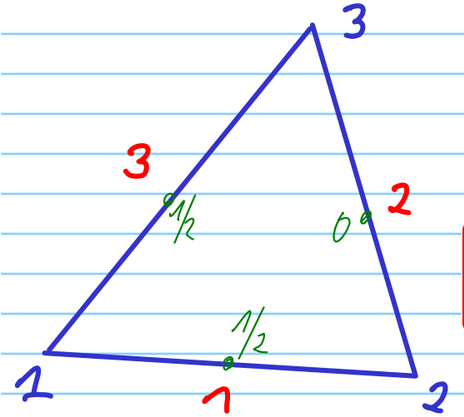
Idea: Add mass matrices to the element matrices for $-\Delta$.

1) Element-rhs-vector $\vec{\varphi}_K \leftrightarrow \int_K f v \, dx$

$$\int_K \varphi \, dx \approx \frac{|K|}{\sharp\mathcal{V}(K)} \sum_{\ell=1}^{\sharp\mathcal{V}(K)} \varphi(m_\ell) . \qquad (2.8.10)$$

m_ℓ: midpoint of edge ℓ

$K \hat{=}$ triangle



$$(\vec{\varphi}_K)_1 = \int_K f(x) \lambda_1(x) \, dx$$

$$= \frac{|K|}{3} \sum_{\ell=1}^{3} f(m_\ell) \lambda_1(m_\ell)$$

$$= \tfrac{1}{3}|K| \, \tfrac{1}{2}(f(m_1) + f(m_3))$$