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

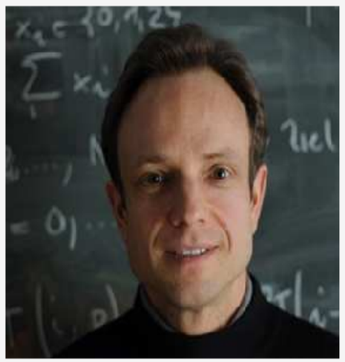
# Homework Problem Video Tutorial

# Problem 2-5: Triangular linear FEM for 2D reaction-diffusion BVP

Prof. R. Hiptmair, SAM, ETH Zurich

Date: March 2, 2019





Essential prerequisite: Sect 2.4.



$$u \in H^1(\Omega): \quad \underbrace{\int_\Omega \mathbf{grad}\, u \cdot \mathbf{grad}\, v + u\, v \,\mathrm{dx}}_{:=a(u,v)} = \underbrace{\int_\Omega f v \,\mathrm{dx}}_{:=\ell(v)} \quad \forall v \in H^1(\Omega). \tag{2.5.1}$$

"reaction term"

a) Strong form of (2.5.1):

l.b.p. (Greens formula)

$$\int_\Omega \underbrace{(-\operatorname{div}\operatorname{grad} u + u)}_{=0 \text{ by testing with } v \in C_0^\infty(\Omega)} v \,dx + \int_{\partial\Omega} \underbrace{\operatorname{grad} u \cdot n}_{=0 \text{ by testing with } v \in C^\infty(\bar\Omega)} v \,dS = \int f v \,dx \quad \forall v \in H^1(\Omega)$$

$$\triangleright \qquad -\Delta u + u = f \quad \text{in } \Omega$$
$$\operatorname{grad} u \cdot n = 0 \quad \text{on } \partial\Omega$$

b) Element matrix $M_K$ for $(u,v) \to \int_\Omega u v \,dx$ for general triangle $K$.

[ Setting: FE space $S_1^0(\mathcal{M})$ ]

$$(M_K)_{i,j} = \int_K \lambda_i \lambda_j \,dx$$

↑ barycentric coordinate fncts.

(2)

**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 $\alpha_j \in \mathbb{N}$, $j = 1, \ldots, d+1$,*

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

$d = 2$ : $(M_K)_{ii} \overset{?}{=} \alpha_1 = 2, \alpha_2 = \alpha_3 = 0$

$\triangleright = 1/6 \, |K|$

$(M_K)_{ij} \overset{?}{=} \alpha_1 = \alpha_2 = 1, \alpha_3 = 0 \quad (i \neq j)$

$\triangleright = 1/12 \, |K|$

$$M_K = \frac{1}{12}|K| \begin{bmatrix} 2 & 1 & 1 \\ 1 & 2 & 1 \\ 1 & 1 & 2 \end{bmatrix}$$

$$\mathbb{1}^T M_K \mathbb{1} = \int_K 1 \cdot dx = |K|$$

c) Computation of $M_K \in \mathbb{R}^{3,3}$

d)

Implement the function

```
double L2Error(const TriangularMesh2D& mesh,
               const Eigen::VectorXd& uFEM,
               const std::function<double(double, double)> exact);
```

to compute the error $\|u_h - u\|_{L^2(\Omega)}$, where $u$ is the exact solution to (2.5.1), passed through the function handle `exact`, and $u_h \in \mathcal{S}_1^0(\mathcal{M})$ is the finite element Galerkin solution passed through its coefficient vector `uFEM` with respect to the customary basis of tent functions. The argument `mesh` contains the mesh data structure.

A function $\in H^1(\Omega) \subset L^2(\Omega)$

Take into account : $u$ in procedural form

$\triangleright$ Numerical quadrature mandatory

$\triangleright$ Cell-oriented computation

$$\|u_h - u\|_{L^2(\Omega)}^2 \approx \sum_{K \in \mathcal{M}} \frac{1}{3}|K| \left( (u_h - u)^2(\boldsymbol{a}_K^1) + (u_h - u)^2(\boldsymbol{a}_K^2) + (u_h - u)^2(\boldsymbol{a}_K^3) \right),$$

$u_h(a_k^i) \overset{?}{=}$ entries of uFEM

e)

Implement the function

```
double H1Serror(const TriangularMesh2D& mesh, const Eigen::VectorXd&
    uFEM,
                const std::function<Eigen::Vector2d(double, double)>
                  exact);
```

to compute the error norm $|u_h - u|_{H^1(\Omega)}$, where $u$ is the exact solution of (2.5.1), whose gradient is passed in the function handle `exact` (that returns a column vector), and $u_h \in \mathcal{S}_1^0(\mathcal{M})$ is the finite element Galerkin solution, passed through the coefficient vector `U`. The input argument `Mesh` contains the mesh data structure.

③ Again: Numerical quadrature by 2D local trapezoidal rule

$$|u_h - u|^2_{H^1(\Omega)} \approx \sum_{K\in\mathcal{M}} \frac{1}{3}|K| \Big( \big\| \mathbf{grad}\, u_h(a_K^1) - \mathbf{grad}\, u(a_K^1) \big\|^2 + \big\| \mathbf{grad}\, u_h(a_K^2) - \mathbf{grad}\, u(a_K^2) \big\|^2 + \big\| \mathbf{grad}\, u_h(a_K^3) - \mathbf{grad}\, u(a_K^3) \big\|^2 \Big) .$$

↑ the same vector $\forall a_K^\ell$

Observe: $\mathbf{grad}\, u_h$ is $\mathcal{M}$-p.w. constant

$$\mathbf{grad}\, u_h|_K = \sum_{\ell=1}^{3} u_h(a_K^\ell)\, \mathbf{grad}\, \lambda_\ell$$

↑ entry of uFEM

**C++ code 2.4.5.11: Computation of gradients of barycentric coordinate functions on a triangle → GITLAB**

```
Eigen::Matrix<double, 2, 3> gradbarycoordinates(const t_TriGeo& Vertices)
{
  // Argument Vertices passes the vertex positions of the triangle
  // as the rows of a 3×2-matrix, , see Code 2.4.1.3..
  // The function returns the components of the gradients as the
  // columns of a 2×3-matrix

  // Computation based on (2.4.5.10), solving for the
  // coefficients of the barycentric coordinate functions.
  Eigen::Matrix<double, 3, 3> X; // Temporary matrix
  X.block<3, 1>(0, 0) = Eigen::Vector3d::Ones();
  X.block<3, 2>(0, 1) = Vertices.transpose();
  return X.inverse().block<2, 3>(1, 0);
}
```

f)

Implement a function

```
std::tuple<Eigen::VectorXd, double, double>
  solve(const TriangularMesh2D& mesh);
```

that, given in input a mesh data structure `mesh`, computes the discrete solution $u_h \in \mathcal{S}_1^0(\mathcal{M})$ to (2.5.1) in the case that the exact solution is $u(x) = \cos(2\pi x_1)\cos(2\pi x_2)$, plots the mesh and $u_h$. The function returns the coefficient vector `U` of $u_h$, the $L^2$-norm and the (approximate, due to quadrature) $H^1$-seminorm of the discretization error $u - u_h$

You can rely on the functions `assemLoad_LFE()` and `GalerkinAssembly()` given in the file `simple_linear_finite_elements.cc`. Those implement cell-oriented assembly according to the "distribute scheme", see [Lecture → Code 2.4.6.8] and [Lecture → Code 2.4.5.23].

How to make a given $u$ a solution of the BVP

- Use strong form → $f := -\Delta u + u$
- Verify b.c.

"method of manufactured solutions"