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

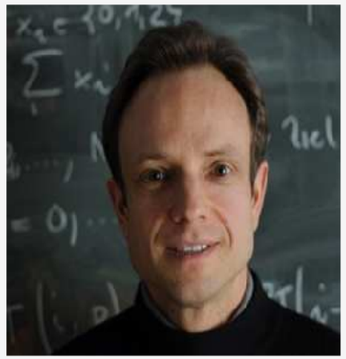
# Course Video

# Section 2.7.6: Treatment of Essential Boundary Conditions

Prof. R. Hiptmair, SAM, ETH Zurich

Date: February 24, 2019

(C) Seminar für Angewandte Mathematik, ETH Zürich



**Prerequisites.**

- Calculus with matrices and vectors

**Dependency.** Requires knowledge about offset techniques [Lecture → § 1.4.1.9], [Lecture → Rem. 2.3.3.15], Galerkin discretization [Lecture → Section 2.2], and linear finite elements [Lecture → Section 2.4].

Note: Possible minor *mismatch of video and tablet notes*!

[Corrections and updates can be incorporated into tablet notes only]



# II. Finite Element Methods (FEM)

## 2.7. Implementation of FEM

### 2.7.6. Treatment of Essential Boundary Conditions

↳ Dirichlet BVPs

$$u \in H^1(\Omega),\ u = g \text{ on } \partial\Omega: \quad \int_\Omega \kappa(x)\,\mathbf{grad}\,u \cdot \mathbf{grad}\,v\,\mathrm{d}x = \int_\Omega f v\,\mathrm{d}x \quad \forall v \in H^1_0(\Omega)\,. \tag{1.8.0.5}$$

$$\Downarrow$$

$$-\operatorname{div}(\kappa(x)\,\mathbf{grad}\,u) = f \text{ in } \Omega\,,\quad u = g \text{ on } \partial\Omega\,,$$

affine space

Abstract LVP in $V$ on affine space

$u \in \hat{V} := u_0 + V_0$ : $a(u,v) = \ell(v)$ $\forall v \in V_0$

↑ offset function

$w \in V_0$ : $a(w,v) = \underbrace{\ell(v) - a(u_0,v)}_{\text{modified r.h.s. functional } \hat{\ell}}$ $\forall v \in V_0$

$\Rightarrow u = u_0 + w$

→ Rem 2.2.1.7.

(2) FE context, 1D linear FE first: → Rem 2.3.3.15

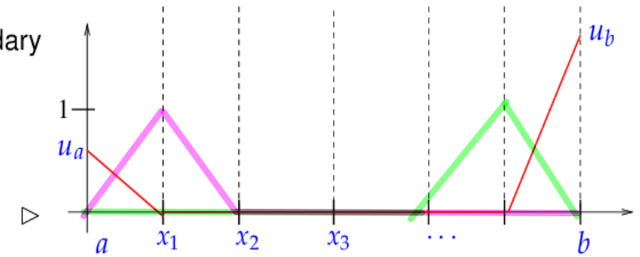
2pt-BVP: $-u'' = f$ in $]a,b[$, $u(a) = u_a$, $u(b) = u_b$

$V_{0,h} = S^0_{1,0}(\mathcal{M})$, $V_h = S^0_1(\mathcal{M}) \subset H^1(]a,b[)$

To deal with the case of general Dirichlet boundary conditions

$$u(a) = u_a\,, \quad u(b) = u_b$$

we use a *piecewise linear* offset function




$$u_{0,h}(x) = \begin{cases} u_a\left(1 - \frac{x-a}{h_1}\right) & \text{, if } a \le x \le x_1\,, \\ u_b\left(1 - \frac{b-x}{h_M}\right) & \text{, if } x_{M-1} \le x \le b\,, \\ 0 & \text{elsewhere,} \end{cases} \quad , \quad \begin{array}{l} u_{0,h} \in H^1(]a,b[)\,, \\ u_{0,h}(a) = u_a,\ u_{0,h}(b) = u_b\,. \end{array} \tag{2.3.3.16}$$

$u_{0,h} \in V_h$

▷ minimal support

$a(u_0, b_h^i) \neq 0$ only if $i = 1, M-1$

↳ $\vec{\varphi} \to \tilde{\vec{\varphi}}$ ≙ only first & last component of $\vec{\varphi}$ change

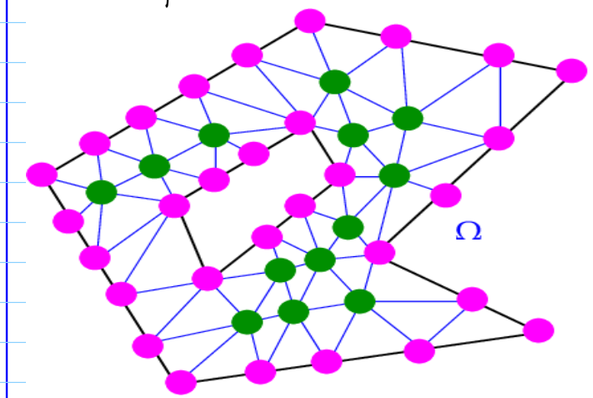
Lesson: Essential BDC can be taken into account by modifying r.h.s. vector

2D: $V_{0,h} = S^0_p(\mathcal{M}) \cap H^1_0(\Omega)$

$V_h = S^0_p(\mathcal{M})$

Drop all GSF associated with entities $\subset \partial\Omega$

Focus $p = 1$: [→ Linear Lagrangian FE]




◁ "Location" of nodal basis functions: (mesh $\mathcal{M}$ → Fig. 49)

●, ● → nodal basis functions of $\mathcal{S}^0_1(\mathcal{M})$

● → nodal basis functions of $\mathcal{S}^0_{1,0}(\mathcal{M})$

Offset function approach: $a(w,v)$

(1.8.0.5) $\Leftrightarrow$ $u = u_0 + w$, $w \in H^1_0(\Omega)$: $\int_\Omega \kappa(\boldsymbol{x})\,\mathbf{grad}\, w \cdot \mathbf{grad}\, v\,\mathrm{d}\boldsymbol{x}$

$$= \int_\Omega -\kappa(\boldsymbol{x})\,\mathbf{grad}\, u_0 \cdot \mathbf{grad}\, v + fv\,\mathrm{d}\boldsymbol{x} \quad \forall v \in H^1_0(\Omega)\,, \tag{2.7.6.2}$$

$\ell(v)$

with offset function $u_0 \in H^1(\Omega)$ satisfying $u_0 = g$ on $\partial\Omega$

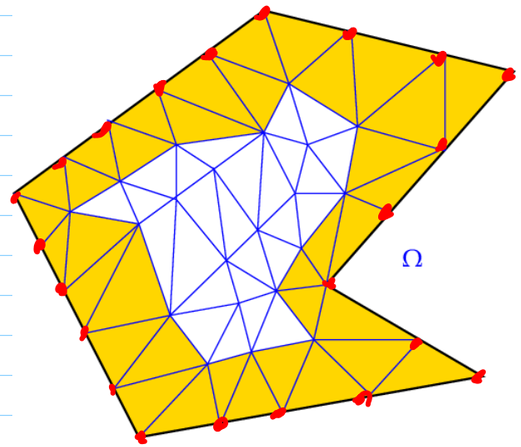
$\tilde{\ell}$

## Finite element offset functions

Idea (inspired by choice in 1D), Rem. 2.3.3.15):
use offset function $u_0 \in V_h := \mathcal{S}^0_p(\mathcal{M})$
*locally supported near the boundary*:
⇕
use offset function in the span of global basis functions associated with geometric entities on $\partial\Omega$

$$\operatorname{supp}(u_0) \subset \bigcup\{K \in \mathcal{M}: \overline{K} \cap \partial\Omega \neq \emptyset\}\,. \tag{2.7.6.4}$$

$p=1$:



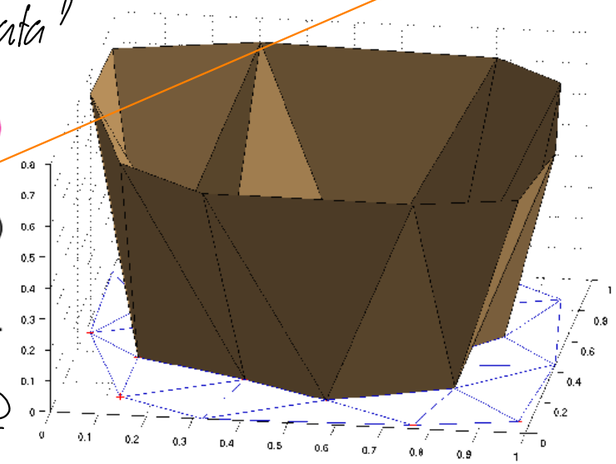

◁ Largest possible support of $u_0$ on a triangular mesh.

"Sampling of Dirichlet data"

For $V_h = \mathcal{S}^0_1(\mathcal{M})$ and Dirichlet data $g \in C^0(\partial\Omega)$ use

$$u_0 = \sum_{x \in \mathcal{V}(\mathcal{M}) \cap \partial\Omega} g(x)\, b^x_h \tag{2.7.6.6}$$

$b^x_h \triangleq$ tent function associated with node $x \in \mathcal{V}(\mathcal{M})$.



Usually: $u_0 \neq g$ on $\partial\Omega$

§2.7.6.8: Algebraic level:

$\mathcal{B}_{0,h} = \{b^1_h, \dots, b^N_h\} \doteq$ basis $\mathcal{S}^0_{1,0}(\mathcal{M}) =: V_{0,h}$

$\mathcal{B}_h = \mathcal{B}_0 \cup \{b^{N+1}_h, \dots, b^M_h\} \doteq$ basis of $\mathcal{S}^0_1(\mathcal{M})$

$\mathbf{A}_0 \in \mathbb{R}^{N,N} \triangleq$ Galerkin matrix for discrete trial/test space $\mathcal{S}^0_{1,0}(\mathcal{M})$,
$\mathbf{A} \in \mathbb{R}^{M,M} \triangleq$ Galerkin matrix for discrete trial/test space $\mathcal{S}^0_1(\mathcal{M})$.

This gives rise to a block-partitioning of the Galerkin matrix $\mathbf{A}$,

$$\mathbf{A} = \left[\begin{array}{c|c} \mathbf{A}_0 & \mathbf{A}_{0\partial} \\ \hline \mathbf{A}_{0\partial}^T & \mathbf{A}_{\partial\partial} \end{array}\right], \quad \mathbf{A}_{0\partial} := \left(\mathsf{a}(b^j_h, b^i_h)\right)_{\substack{i=1,\dots,N \\ j=N+1,\dots,M-N}} \in \mathbb{R}^{N,M-N}, \quad \mathbf{A}_{\partial\partial} := \left(\mathsf{a}(b^j_h, b^i_h)\right)_{\substack{i=N+1,\dots,M-N \\ j=N+1,\dots,M-N}} \in \mathbb{R}^{M-N,M-N}. \tag{2.7.6.9}$$

$u_0 = \sum_{j=N+1}^{M} g_j\, b^j_h$, $g_j = g(x^j)$

(1.8.0.5) $\Leftrightarrow$ $u = u_0 + w$, $w \in H^1_0(\Omega)$:

$$\underbrace{\int_\Omega \kappa(x)\,\mathbf{grad}\, w \cdot \mathbf{grad}\, v\,\mathrm{d}x}_{= a(w,v)} = \int_\Omega -\kappa(x)\,\mathbf{grad}\, u_0 \cdot \mathbf{grad}\, v + f v\,\mathrm{d}x \quad \forall v \in H^1_0(\Omega)\,, \tag{2.7.6.2}$$

with offset function $u_0 \in H^1(\Omega)$ satisfying $u_0 = g$ on $\partial\Omega$

"Offset BVP":

$w_h \in V_{0,h}$ : $a(w_h, v_h) = \ell(v_h) - a(u_0, v_h)$ $\forall v_h \in V_{0,h}$

(4)

Expand: $u_h = \sum_{j=1}^{N} v_j b_h^j$, $v_h \leftrightarrow b_h^i$, $i = 1,\dots,N$

Plug into 0-BVP:

$$\sum_{j=1}^{N} a(b_h^j, b_h^i) v_j = \ell(b_h^i) - \sum_{\ell=N+1}^{M} a(b_h^\ell, b_h^i) g_\ell \quad \Leftrightarrow$$

Modified LSE: $A_0 \vec{v} = \vec{\varphi} - A_{02} \vec{g}$

$$u_h = \sum_{j=1}^{N} v_j b_h^j + \sum_{\ell=N+1}^{M} g_\ell b_h^\ell$$

Essential BDC in LF++: [§ 2.7.6.13]

$$\begin{bmatrix} A_0 & A_{02} \\ 0 & I \end{bmatrix} \vec{v} = \begin{bmatrix} \vec{\varphi} \\ \vec{g} \end{bmatrix}$$

by modification of $A$

Note: No sorting of GSF must be assumed!

```
template <typename SCALAR, typename SELECTOR,
          typename RHSVECTOR>
void FixFlaggedSolutionCompAlt(SELECTOR &&selectvals,
  lf::assemble::COOMatrix<SCALAR> &A, RHSVECTOR &b);
```

passes: • flags boundary dofs
• $g_\ell$

**C++11 code 2.7.6.17: Transformation function** (2.7.6.14) → (2.7.6.16) ↪ GITHUB

```
template <typename SCALAR, typename SELECTOR, typename RHSVECTOR>
void FixFlaggedSolutionCompAlt(SELECTOR &&selectvals, COOMatrix<SCALAR> &A,
                               RHSVECTOR &b) {
  const lf::assemble::size_type N(A.cols());
  LF_ASSERT_MSG(A.rows() == N, "Matrix must be square!");
  LF_ASSERT_MSG(N == b.size(),"Mismatch N = " << N << " <-> b.size() = " <<
      b.size());
  // I: Set components of right-hand-side vector to prescribed values
  for (lf::assemble::gdof_idx_t k = 0; k < N; ++k) {
    const auto selval{selectvals(k)};
    if (selval.first) b[k] = selval.second;
  }
  // II: Set rows of the sparse matrix corresponding
  // to the fixed solution components to zero
  typename lf::assemble::COOMatrix<SCALAR>::TripletVec::iterator new_last =
      std::remove_if(
          A.triplets().begin(), A.triplets().end(),
          [&fixed_comp_flags](
              typename lf::assemble::COOMatrix<SCALAR>::Triplet &triplet) {
            return (fixed_comp_flags[triplet.row()]);
          });
  // Adjust size of triplet vector
  A.triplets().erase(new_last, A.triplets().end());
  // III: Add Unit diagonal entries corrresponding to fixed components
  for (lf::assemble::gdof_idx_t dofnum = 0; dofnum < N; ++dofnum) {
    if (selectvals(dofnum).first) A.AddToEntry(dofnum, dofnum, 1.0);
  }}
```

→ modification of rhs vector

Works on triplet format

⑤ Review questions 2.7.6.21 :

A: What is the offset function technique for handling essential boundary conditions ?

B: What is a convenient offset function when discretizing a pure Dirichlet boundary value problem by means of quadratic Lagrangian finite elements, space $S_2^0(\mathcal{M})$?

C:

On a polyognally bounded domain $\Omega \subset \mathbb{R}^2$ we solve the Dirichlet problem

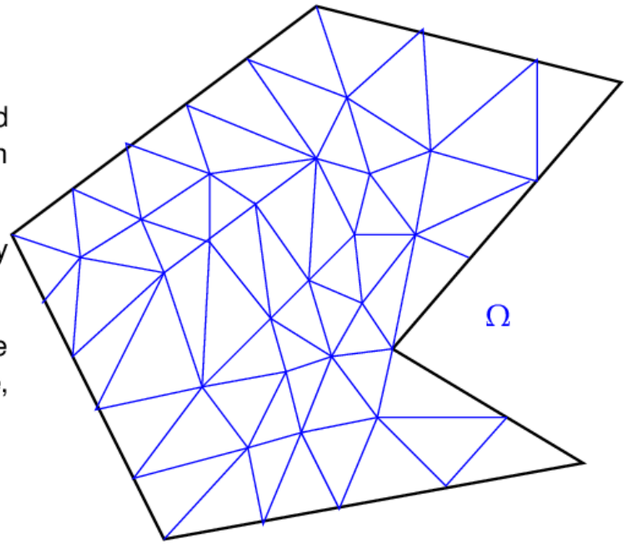
$$-\Delta u = 0 \quad \text{in} \quad \Omega \quad , \quad u = g \quad \text{on} \quad \partial\Omega \, ,$$

with $g \in C^0(\partial\Omega)$.

We use a finite element Galerkin discretization based on $\mathcal{S}_1^0(\mathcal{M})$, where $\mathcal{M}$ is the triangular mesh drawn beside.

We employ the elimination of essential boundary conditions based on the offset function technique.

How many entries of the right-hand side vector of the finite-element linear system of equations will change, when the data $g$ change?



⑥ D:

Sketch the implementation of a class

```
class SolveLaplaceBVP {
public:
  SolveLaplaceBVP(std::shared_ptr<const lf::assemble::DofHandler>
     dofh_p);
  ~SolveLaplaceBVP() = default;

  template <typename FUNCTOR>
  Eigen::VectorXd solveLaplaceBVP(FUNCTOR &&g) const;
private:
  ....
};
```

The constructor takes a shared pointer to a **lf::assemble::DofHandler** object associated belonging to the finite element space $\mathcal{S}_1^0(\mathcal{M})$. The method `solveLaplaceBVP()` takes a functor argument of type `std::function<double(Eigen::Vector2D)>`, which supplies the boundary data $g$. It is supposed to return the basis expansion coefficient vector (with respect to the standard tent function basis) of the finite element solution $\in \mathcal{S}_1^0(\mathcal{M})$ of the Dirichlet boundary value problem

$$-\Delta u = 0 \quad \text{in} \quad \Omega \quad , \quad u = g \quad \text{on} \quad \partial\Omega \, .$$

This list of review questions may not be complete. Additional review questions may be provided in the lecture document.