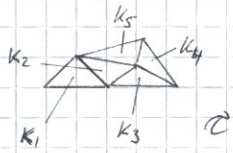
Notes 5.11. pp 2/

Mesh

In 1D we discretised our domain Ω into a finite number of points / intervals.
In 2D our domain Ω represents some area / surface confined by the boundary $\partial\Omega$.
In order to solve our derived variational problems we assume some

Triangulation $\mathcal{T}$

so our domain Ω is somehow divided into triangles, say

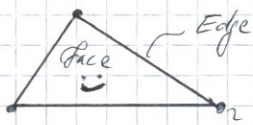


where $\Omega = \bigcup_{K \in \mathcal{T}} K$ (union)

Such triangulation can take time: For the Boeing Airplane it took a month.

Lets repeat the

Notational baggage:



remember triangles do not overlap, if two triangles share part of an edge they share the whole edge!

so $N_i \cap N_j = \begin{cases} \text{Edge} \\ \text{Vertex} \\ \emptyset \end{cases}$ $i \neq j$

In the case of linear interpolation i.e. linear basis fns a.k.a. hat-fn's from 1D

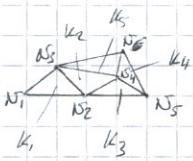
let $h = \min_{\mathcal{T}} \operatorname{diam}(K)$ (minimum longest edge of all triangles)

we define a finite dim. subspace $V^h \in H_0^1(\Omega)$ s.t. as

$$V^h = \{ v \in H_0^1(\Omega) \mid v \text{ cont.}, \; v|_K \text{ linear} \}$$

i.e. basis fn 

So, how can we represent our triangulation $\mathcal{T}$, i.e. store it on a computer?



"Naive" approach: store the vertices in terms of coord.

$$\{ (N_1, N_2, N_3) \; (N_2, N_4, N_3) \; (N_2, N_5, N_4) \; \dots \}$$

Note: counter-clockwise order of vertices

'where each vertex needs $\geq$ double numbers to store coords'

i.e. $((0.2, 0.3), (0.1, \dots), \dots)$ BUT comparing floating pt. numbers is generally speaking a bad idea

take into acc. precision value

Instead we make an array of all vertices

$$Z = (N_1, N_2, \ldots, N_n)$$

and store the indices for all triangles in another list

$$T = ((1,2,3), (2,4,3), (2,5,4), \ldots)$$

Series 5 Ex 1 p.9

The idea is, that then we can access all the information uniformly i.e. we loop over all triangles and simply 'read out' the vertices we need i.o. to compute our solution contributions.

Then $Z(q,i)$ is the q^{th} component of the vertex N_i
(Array Z: $2 \times n$)
and $T(x,m)$ is the x^{th} vertex of the triangle K_m
(Array T: 3×91)
$\Rightarrow$
$$Z(q, T(x,m)) = q^{th} \text{ component of the } x^{th} \text{ vertex of } K_m$$

uncertainty

Quadratic Integration

Instead of a linear (1st order) approach to the Finite El. Method we could also develop a 2nd order finite el. method.

Of course we can do so in 1 and 2 dimensions.

Say we consider the problem in Series 5 Ex. 2

$$\begin{cases} -\Delta u = f(x) & \text{in } \Omega \subset \mathbb{R}^2 \\ u(x) = 0 & \text{on } \partial\Omega \end{cases}$$

where $f \in L^2(\Omega)$

x contains (x_1, x_2)
2> vector

Note: Norms:

$\|v\|_{H_0^1(\Omega)} = \left(\int_\Omega |\nabla v|^2 dx\right)^{1/2}$; $\|v\|_{L^2(\Omega)} = \left(\int_\Omega |v|^2 dx\right)^{1/2}$

hence

$(u,v)_{H_0^1(\Omega)} = \int_\Omega \langle \nabla u, \nabla v \rangle dx$;

$(f,v)_{L^2(\Omega)} = \int_\Omega f(x) \cdot v(x)\, dx$;

Then our Variational Formulation reads:

Find $u \in H_0^1$ s.t. $\forall\, v \in H_0^1$ it holds that

$$(u,v)_{H_0^1(\Omega)} = (f,v)_{L^2(\Omega)}$$

Since we now solve this by means of quadratic finite elements on our triangular mesh, we consider the finite-dim. subspace of $H_0^1(\Omega)$, namely

$$V^h = \{ \omega : \Omega \to \mathbb{R} \mid \omega \text{ is cont.}, \ \omega|_{\partial\Omega} = 0 \ (\text{from BV condition}), \ \omega|_K \text{ is a } 2^{\text{nd}} \text{ order polynomial } \forall K \in \mathcal{C} \}.$$

In general,

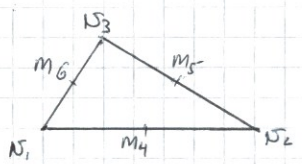
we assume $\omega(x,y) = a_{0,0} + a_{1,0}x + a_{0,1}y + a_{1,1}xy + a_{2,0}x^2 + a_{0,2}y^2$

where $a_{ij} \in \mathbb{R}$.

So, now we have basis functions of two variables and 6 coeff. defined on the whole element K (triangle).

For simplicity we decompose these base fns into two "base fns"

- Contribution of the vertices of K
- Contribution of the mid-points of each edge of K

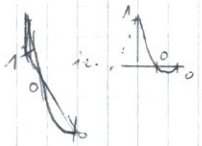


and hence (in analogy to the notation which is used in Series 5)

$x_j \Rightarrow N_j$ (vectors)

$$\delta_i(N_j) := \begin{cases} 1 & i=j \\ 0 & \text{otherwise} \end{cases} \qquad \forall \{i\}_0^{N_V-1} \qquad (N_V \text{ \# of vertices})$$

$$\varphi_i(m_j) := \begin{cases} 1 & i=j \\ 0 & \text{otherwise} \end{cases} \qquad \forall \{i\}_0^{N_E-1} \qquad (N_E \text{ \# of edges})$$



δ_i, φ_i defined globally

Usually, we are then to determine the coefficients (6 per base fn) and of course (since we are lazy and ~~scatter tag back~~) we do that on the reference element, where we establish six solvable eqns

3 from the vertices

3 from the midpoints

Similarly, in 1D we add contributions at all $x_{i+1/2}$ and an acc. base fn. To define these fns you are given (or you define) restrictions

Now, we put these contributions back together to get our shape fns

$$\varphi_i^N := \begin{cases} \delta_i & i \in [0, \dots, N_V-1] \\ \varphi_{i-N_V} & i \in [N_V, \dots, N-1] \end{cases}$$

N the # of base fns $N_V + N_E$

and since we know that

$$\{\varphi_i\}_{i=1}^{N} \text{ form a basis for } V^h$$

2⇒ since our var. problem holds for all $v \in V^h$ it particularly holds for our $v = \varphi_i$ ⇒

we can write our approx. solution

$$u_N(x) = \sum_{i=1}^{N} \mu_i \varphi_i^N(x)$$

$x \to$ vector (x_1, x_2)

where

$$\mu = \{\mu_i\}_{i=0}^{N-1} \quad \in$$

comprises all coefficients "weighing" the shape fns' contribution.

Our var. formulation reads:

$$(\nabla u_N, \nabla \varphi_i) = (f, \varphi_i) \qquad \forall i \in [0, \dots, N-1]$$

and inserting our approx. solution yields:

$$\left(\nabla\left(\sum_j \mu_j \varphi_j\right), \nabla \varphi_i\right) = (f, \varphi_i) \quad \text{, where } \mu_i \text{ are const. w.r.t. } x \text{ (vector)}$$

$$= \left(\sum_j \mu_j \nabla \varphi_j, \nabla \varphi_i\right)$$

$$= \sum_j \mu_j \left(\nabla \varphi_j, \nabla \varphi_i\right)$$

Assembly

So, now for we get this eqn for each $i=1,\dots,N$

$$\sum_{j=1}^{N} \mu_j \underbrace{(\nabla\varphi_j, \nabla\varphi_i)}_{A_{ij}} = \underbrace{(g, \varphi_i)}_{F_i}$$

Hence we get the linear system of eq'ns

$$A\mu = F \quad \text{, where } \mu \text{ is the vector of unknowns } \{\mu_i\}_{i=0}^{N-1}$$

From this we see that we can get lucky i.e. in the 1D case, where we only have contributions from directly neighbouring base fns or unlucky:

In order to further discuss the importance of these constraints I continue by repeating the assembly process in more detail before solving an 'example' exercise where we make use of these concepts in a more 'exam-like' landscape.

Going back to our system of equations, where we assumed a triangulation s.t. triangles do not overlap (except for vertices and edges) and if, those whole edges, we had the matrix element

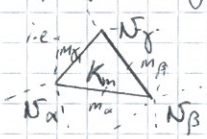
$$A_{ij} = \int_{\Omega} (\nabla\varphi_i, \nabla\varphi_j)\,dx \quad \text{and} \quad \Omega = \overset{\text{Mesh}}{\bigcup_{m=1}^{M}} K_m$$

$$= \sum_{m=1}^{M} \int_{K_m} (\nabla\varphi_i, \nabla\varphi_j)\,dx$$

$(u, v) = u \cdot v$ dot prod.
$u_1v_1 + u_2v_2 \cdots$
remember x is a 'vector'

and it only makes sense to loop over all triangles, when computing the entries.

For a specific triangle



the integral is only non-zero if i,j are to be a vertex on midpoint of K. → in the linear case i.e. 1st order polynomial base fns only vertices

$$\int_{K_m} (\nabla \varphi_i, \nabla \varphi_j) \neq 0 \quad \text{only iff } i,j \in K_m \text{ i and j are vertices (midpoints) of } K_m.$$

which immediately solidifies this new perspective:

- instead of looping over the base fns, we loop over triangles $K_1, \dots, K_M$
- for each triangle we compute the integral for all pairs where $i,j \in K_m$

Of course, to make things easier (remember, we are lazy) we project our triangles and base fns onto a reference element and compute our integrals there.
Diego introduced this 'mapping' very nicely last week, so I am allow myself to skip that step.

Now, for a triangle K (a general triangle in our mesh) let us define the

local basis fns φ_α^K for $\alpha = \begin{cases} 1,2,3 & \text{linear int.} \\ 1,2,3,4,5,6 & \text{quadr. int.} \end{cases}$

so that $$\varphi_\alpha^K(N_i) = \begin{cases} 1 & T(\alpha, K) = i \\ 0 & \text{else} \end{cases}$$

where i is a global variable (all nods).

~~to the vertices~~

You remember, remember, in the linear case (i.e. linear base fns) we get 3×3 contributions to the matrix A, so 9 for each triangle.
→ 9 combinations of inner products
Similarly we get $6 \times 6 \sim 36$ in the quadratic case.
→ 6 combinations of inner products

$$\int_K (\nabla \varphi_\alpha^K, \nabla \varphi_\beta^K)\, dx \quad , \alpha, \beta \in [1, \dots, 6]$$

And we can define the

local stiffness matrix A^K

$$A^K_{\alpha,\beta} = \int_K (\nabla \varphi_\alpha^K, \nabla \varphi_\beta^K)\, dx \quad ; \alpha, \beta = 1, \dots, 6$$

and since the dot prod. is indifferent to the order

A^K is symmetric

→ so no need to compute all entries
(we realize skip 'half' the computations)

12.11.
PP 3/4

CFEA II

26.11.20

Assembly cont.

Of course, now the reference mapping comes in very handy: We solve this locally on $\hat{K}$ s.t. it simplifies our computations.

Using $\hat{K}$ (our ref. triangle) we get

comp. $\bar{J}^1$ by hand

$$A^K_{\alpha,\beta} = \int_{\hat{K}} \underbrace{\left(J_K^{-T} \nabla \varphi_\alpha^{\hat{K}}\right)}_{\nabla \varphi_\alpha^K(\Phi_K(y))} \cdot \left(J_K^{-T} \nabla \varphi_\beta^{\hat{K}}\right) |\det J_K| \, dx$$

So, now you might ask, how do we get back to our system of linear equations? So far we only have local coefficients...

Pseudo-Algorithm:

- Computing A (s.c. stiffness matrix)

  For $m = 1, \dots, M$ // looping over all triples

  Compute A^{K_m}

  For each $\alpha, \beta = 1, 2, 3, (4, 5, 6)$ (quadrat. case)

  // using symmetry!

  $$A^{K_m}_{\alpha,\beta} = \int_{\hat{K}} \dots \, dx$$

  // assembly, globally (2x add. contr.)

  $$A_{T(\alpha,m),T(\beta,m)} \mathrel{+}= A^{K_m}_{\alpha,\beta}$$

  $T(\alpha, m)$ the αth vertex of K_m (2on mode)

(Exk.)

where we remember

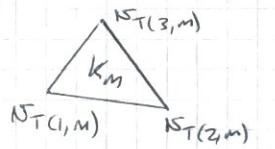
$$A_{ij} = \sum_m \int_{K_m} (\nabla \varphi_i, \nabla \varphi_j) \, dx$$

$$A^{K_m}_{\alpha,\beta} = \int_{K_m} (\nabla \varphi_\alpha^K, \nabla \varphi_\beta^K) \, dx$$

βth vert. of $K_m \to T(\beta, m) \to$ globally j

αth vertex of $K_m \to T(\alpha, m) \to$ globally i

So basically for some triangle K_m with global vertices



now for the RHS i.e. our load vector we have

$$(f, \varphi_i) = \sum_m \int_K (f, \varphi_i) \, dx$$

- Computing F

  Compute local load vectors

  // using change of variables

  $$F_m^\alpha = \int_{\hat{K}} f(\Phi_K(y)) \cdot \varphi_\alpha^{\hat{K}}(y) \, |\det J_K| \, dy$$

  // globally

  $$F_{T(\alpha,m)} \mathrel{+}= F_m^\alpha$$

Let's say same vertex appears twice, hence we get two triplets in an efficient matrix with the same indices,

i.e. Triplet $(i, j, 7.67)$

Triplet $(i, j, e^{\wedge 2})$

$\Rightarrow A_{ij} = 7.67 + e^{\wedge 2}$.