Part 3 - Practical Aspects of Finite Elements in 2D

3.1. Numerical Quadratures

2D Numerical Quadrature

Applying finite elements in $2D$ requires to compute integrals of the form

$$\int_K w(x)\,dx,$$

where

- $K \in \mathcal{T}_h$ is a triangle (simplex),
- $w(x)$ is a function to be integrated;
  typically we have
  - $w(x) = k(x)\nabla\phi_i(x) \cdot \nabla\phi_j(x)$ for entries of the mass matrix or
  - $w(x) = f(x)\,\phi_j(x)$ for entries of the right hand side vector.

2D Numerical Quadrature

Exact evaluation of integrals can be inefficient or impossible.

Practically, we can approximate the integral with numerical quadrature rule:

$$\int_K w(x)\,dx \approx \sum_{i=1}^{n_q} w(q_i)\,\omega_i,$$

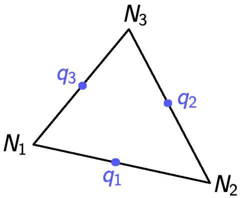
where

- q_i: quadrature points (nodes),
- ω_i: quadrature weights,
- n_q: number of quadrature points.

2D Numerical Quadrature

Example 1: Edge-midpoint rule.

- $q_i = \frac{N_i + N_j}{2}$: edge midpoints as quadrature points,
- $\omega_i = \frac{|K|}{3}$: quadrature weights,
  where $|K|$ is the area of K
- $n_q = 3$: quadrature points.

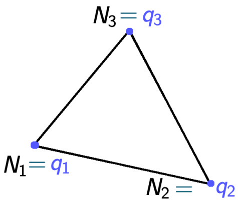


$$\int_K w(x)\,dx \approx \frac{|K|}{3}\left(w(q_1) + w(q_2) + w(q_3)\right)$$

2D Numerical Quadrature

Example 2: 2D trapezoidal rule.

- $q_i = N_i$: corners as quadrature points,
- $\omega_i = \frac{|K|}{3}$: quadrature weights,
  where $|K|$ is the area of K
- $n_q = 3$: quadrature points.



$$\int_K w(x)\,dx \approx \frac{|K|}{3}\left(w(N_1) + w(N_2) + w(N_3)\right)$$

Part 3 - Practical Aspects of Finite Elements in 2D

3.2. Mesh generation

Mesh generation

How do we generate a particular mesh $\mathcal{T}_h$ in Matlab?

Let $\mathcal{T}_h$ consist N nodes and M triangles.

In Matlab, the data structure that describes the mesh is stored in three matrices:

- p: the point or node matrix;
- e: the boundary edge matrix;
- t: the triangulation matrix;

p: the node matrix

The matrix $p \in \mathbb{R}^{2\times N}$ describes the nodes, i.e., the nodes

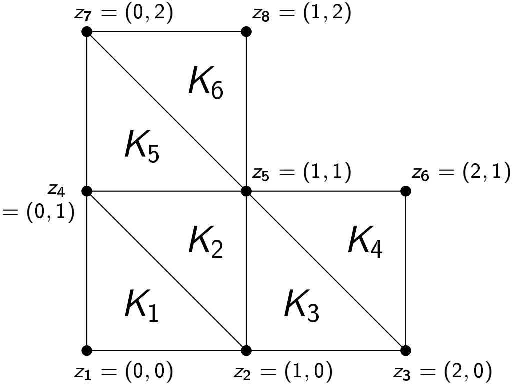
$$z_i = (x_i, y_i) \text{ for } i = 1, \cdots, N$$

form the entries:

$$p = \begin{pmatrix} x_1 & x_2 & \cdots & x_N \\ y_1 & y_2 & \cdots & y_N \end{pmatrix}$$

p: the node matrix

Example:



The node matrix $p \in \mathbb{R}^{2\times N}$ is

$$p = \begin{pmatrix} 0 & 1 & 2 & 0 & 1 & 2 & 0 & 1 \\ 0 & 0 & 0 & 1 & 1 & 1 & 2 & 2 \end{pmatrix}$$

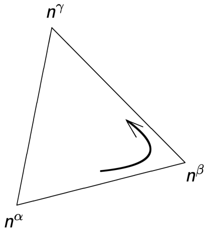
t: the triangulation matrix

Matrix $t \in \mathbb{R}^{3\times M}$ describes the triangles, i.e., which nodes (numerated from 1 to N) form a triangle K and how it is orientated:

$$t = \begin{pmatrix} n_1^\alpha & n_2^\alpha & \cdots & n_M^\alpha \\ n_1^\beta & n_2^\beta & \cdots & n_M^\beta \\ n_1^\gamma & n_2^\gamma & \cdots & n_M^\gamma \\ 1 & 1 & \cdots & 1 \end{pmatrix}.$$

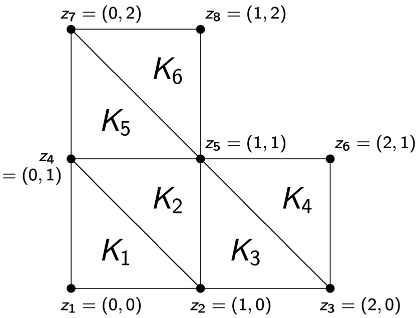
Last row: subdomain number (for simple FEM always 1).

This means that triangle K_i is formed by the nodes with indices n_i^α, n_i^β and n_i^γ (enumeration in counter-clockwise direction).



t: the triangulation matrix

Example:



The triangulation matrix $t \in \mathbb{R}^{3\times M}$ is given by

$$t = \begin{pmatrix} 1 & 2 & 2 & 3 & 4 & 5 \\ 2 & 5 & 3 & 6 & 5 & 8 \\ 4 & 4 & 5 & 5 & 7 & 7 \\ 1 & 1 & 1 & 1 & 1 & 1 \end{pmatrix}.$$

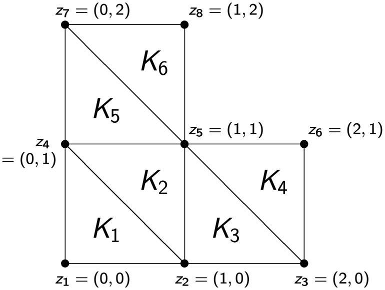
e: the boundary edge matrix

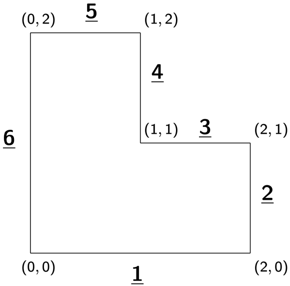
The matrix e describes the edges that form the boundary of the domain
(or the boundaries of subdomains).

- rows 1 and 2:
  indices of starting and ending point of the edges;
- rows 3 and 4:
  starting and ending parameter values
  (percentage location on the boundary segment);
- row 5: boundary segment number;
- rows 6 and 7: left- and right-hand side subdomain numbers ("0" for "outside", i.e., $\mathbb{R}^2 \setminus \Omega$)

e: the boundary edge matrix

Exp:





The edge matrix $e \in \mathbb{R}^{3\times 8}$ is given by

$$
e = \begin{array}{cl}
\left(\begin{array}{cccccccc}
1 & 2 & 3 & 6 & 5 & 8 & 7 & 4 \\
2 & 3 & 6 & 5 & 8 & 7 & 4 & 1 \\
0.0 & 0.5 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.5 \\
0.5 & 1.0 & 1.0 & 1.0 & 1.0 & 1.0 & 0.5 & 1.0 \\
\underline{1} & \underline{1} & \underline{2} & \underline{3} & \underline{4} & \underline{5} & \underline{6} & \underline{6} \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
1 & 1 & 1 & 1 & 1 & 1 & 1 & 1
\end{array}\right)
&
\begin{array}{l}
\longleftarrow \quad \text{starting node of edge} \\
\longleftarrow \quad \text{ending node of edge} \\
\longleftarrow \quad \text{starting parameter value} \\
\longleftarrow \quad \text{ending parameter value} \\
\longleftarrow \quad \text{boundary segment index} \\
\longleftarrow \quad \text{domain to the right} \\
\longleftarrow \quad \text{domain to the left}
\end{array}
\end{array}
$$

e: the boundary edge matrix

Note:

If there is just one computational domain Ω then:

- Ω has the subdomain ID 1;
- $\mathbb{R}^2 \setminus \Omega$ has the subdomain ID 0.

Example code (in Canvas):

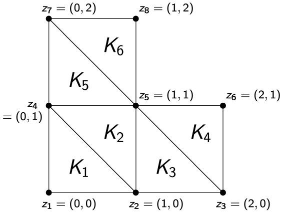
`corner_mesh.m`

Mesh generation

In many examples, we also need the decomposed geometry matrix of the mesh.

The syntax for the domain geometry matrix is as follows and only involves the edges that form the boundary:

$$\text{geom} = \begin{pmatrix} 2 & 2 & 2 & 2 & 2 & 2 & 2 & 2 \\ 0 & 1 & 2 & 2 & 1 & 1 & 0 & 0 \\ 1 & 2 & 2 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 2 & 2 & 1 \\ 0 & 0 & 1 & 1 & 2 & 2 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \end{pmatrix} \begin{array}{ll} \longleftarrow & \text{polygonal boundary} \\ \longleftarrow & x \text{ coordinate 1 of boundary edge} \\ \longleftarrow & x \text{ coordinate 2 of boundary edge} \\ \longleftarrow & y \text{ coordinate 1 of boundary edge} \\ \longleftarrow & y \text{ coordinate 2 of boundary edge} \\ \longleftarrow & \text{domain to the right} \\ \longleftarrow & \text{domain to the left} \end{array}$$



Mesh generation

The matrices can be also generated automatically.

For example, for a square mesh with mesh size $h = 0.1$, we have

```
h = 0.1; % mesh size
[p , e , t] = initmesh( 'squareg' , 'hmax' , h);
```

For a L-shaped mesh with mesh size $h = 0.1$, we have

```
h = 0.1; % mesh size
[p , e , t] = initmesh( 'lshapeg' , 'hmax' , h);
```

The mesh can be also automatically refined using

```
[p,e,t] = refinemesh('lshapeg',p,e,t);
```

Part 3 - Practical Aspects of Finite Elements in 2D

3.3. Matrix assembly

Motivation: L^2-projection

Consider a FE-space V_h and a function $f \in L^2(\Omega)$.

Goal: find L^2-best approximation of f in V_h, i.e.,

find $f_h \in V_h$ with

$$(f_h, v_h)_{L^2(\Omega)} = (f, v_h)_{L^2(\Omega)} \qquad \text{for all } v_h \in V_h.$$

This can be written as an algebraic system

$$\mathsf{M}\mathbf{f}_h = \mathbf{f},$$

where $\mathsf{M} \in \mathbb{R}^{N\times N}$ is the mass matrix with entries

$$\mathsf{M}_{ij} = \int_\Omega \phi_j \, \phi_i \qquad \text{for } i, j = 1, \cdots N.$$

and

$$\mathbf{f}_j = \int_\Omega f \, \phi_j \qquad \text{for } j = 1, \cdots N.$$

Assembly of mass matrix in Matlab

Exemplarily, we consider the assembly of the mass matrix $\mathbf{M} \in \mathbb{R}^{N\times N}$ with entries

$$M_{ij} = \int_\Omega \phi_j \, \phi_i \qquad \text{for } i,j = 1,\cdots N.$$

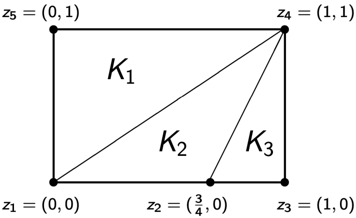
Practically, we split $\mathbf{M}$ into "element contributions" $\mathbf{M}^K$ for each triangle $K \in \mathcal{T}_h$.

It can be easily computed that for $i \neq j$

$$\int_K \phi_{K,i}\, \phi_{K,j} = \frac{1}{12}|K| \qquad \text{and} \qquad \int_K \phi_{K,i}^2 = \frac{1}{6}|K|.$$

Assembly of mass matrix in Matlab

Example:



$$\mathsf{M} = \int_\Omega \begin{pmatrix} \phi_1\phi_1 & \phi_1\phi_2 & \phi_1\phi_3 & \phi_1\phi_4 & \phi_1\phi_5 \\ \phi_2\phi_1 & \phi_2\phi_2 & \phi_2\phi_3 & \phi_2\phi_4 & \phi_2\phi_5 \\ \phi_3\phi_1 & \phi_3\phi_2 & \phi_3\phi_3 & \phi_3\phi_4 & \phi_3\phi_5 \\ \phi_4\phi_1 & \phi_4\phi_2 & \phi_4\phi_3 & \phi_4\phi_4 & \phi_4\phi_5 \\ \phi_5\phi_1 & \phi_5\phi_2 & \phi_5\phi_3 & \phi_5\phi_4 & \phi_5\phi_5 \end{pmatrix} = \sum_{K\in\mathcal{T}_h} \int_K \begin{pmatrix} \phi_1\phi_1 & \phi_1\phi_2 & \phi_1\phi_3 & \phi_1\phi_4 & \phi_1\phi_5 \\ \phi_2\phi_1 & \phi_2\phi_2 & \phi_2\phi_3 & \phi_2\phi_4 & \phi_2\phi_5 \\ \phi_3\phi_1 & \phi_3\phi_2 & \phi_3\phi_3 & \phi_3\phi_4 & \phi_3\phi_5 \\ \phi_4\phi_1 & \phi_4\phi_2 & \phi_4\phi_3 & \phi_4\phi_4 & \phi_4\phi_5 \\ \phi_5\phi_1 & \phi_5\phi_2 & \phi_5\phi_3 & \phi_5\phi_4 & \phi_5\phi_5 \end{pmatrix}$$

$$= \int_{K_1} \begin{pmatrix} \phi_1\phi_1 & 0 & 0 & \phi_1\phi_4 & \phi_1\phi_5 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ \phi_4\phi_1 & 0 & 0 & \phi_4\phi_4 & \phi_4\phi_5 \\ \phi_5\phi_1 & 0 & 0 & \phi_5\phi_4 & \phi_5\phi_5 \end{pmatrix} + \int_{K_2} \begin{pmatrix} \phi_1\phi_1 & \phi_1\phi_2 & 0 & \phi_1\phi_4 & 0 \\ \phi_2\phi_1 & \phi_2\phi_2 & 0 & \phi_2\phi_4 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ \phi_4\phi_1 & \phi_4\phi_2 & 0 & \phi_4\phi_4 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{pmatrix}$$

$$+ \int_{K_3} \begin{pmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & \phi_2\phi_2 & \phi_2\phi_3 & \phi_2\phi_4 & 0 \\ 0 & \phi_3\phi_2 & \phi_3\phi_3 & \phi_3\phi_4 & 0 \\ 0 & \phi_4\phi_2 & \phi_4\phi_3 & \phi_4\phi_4 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{pmatrix} \text{``}=\text{''} \; \mathsf{M}^{K_1} + \mathsf{M}^{K_2} + \mathsf{M}^{K_3}.$$

Assembly of mass matrix in Matlab

Recalling the previous formula, we have

$$\int_K \phi_i \phi_j = \frac{1}{12}(1+\delta_{ij})|K| \quad \text{for } i,j = 1,2,3,$$

where $\delta_{ij} = 1$ for $i = j$ and $\delta_{ij} = 0$ for $i \neq j$.

This gives us the local mass matrices

$$\mathbf{M}^{K_i} = \frac{1}{12}\begin{pmatrix} 2 & 1 & 1 \\ 1 & 2 & 1 \\ 1 & 1 & 2 \end{pmatrix} |K_i|.$$

How do we “add” these 3×3-matrices to the global mass matrix $\mathbf{M}$?

Plain algorithm for assembly of mass matrix

- Construct node matrix p and triangle matrix t matrices.
- Allocate memory for the $N \times N$-matrix M.
- for $K \in \mathcal{T}_h$ compute:

$$\mathsf{M}^K = \frac{1}{12}\begin{pmatrix} 2 & 1 & 1 \\ 1 & 2 & 1 \\ 1 & 1 & 2 \end{pmatrix} |K|$$

- Update the global mass matrix M with M^K:
  `M(t(1:3,K),t(1:3,K)) = M(t(1:3,K),t(1:3,K)) + MK;`
  Here, `t(i,K)` returns the global node index of i'th corner in the triangle K.
- end

Example code:

```
function M = MassAssembler2D(p,t)

np = size(p,2);     % number of nodes
nt = size(t,2);     % number of elements
M = sparse(np,np); % allocate sparse mass matrix

for K = 1:nt     % loop over elements
  loc2glb = t(1:3,K); % local to global index
   x = p(1,loc2glb);
   y = p(2,loc2glb);
   area = polyarea(x,y); % triangle area
   MK = [2 1 1;
         1 2 1;
         1 1 2]/12*area; % element mass matrix
   % add element masses to M:
   M(loc2glb,loc2glb) = M(loc2glb,loc2glb) + MK;
end
```

Example 2: assembly of load vector **b** with entries $\mathbf{b}_i = (f, \phi_i)_{L^2(\Omega)}$:

```
function b = LoadAssembler2D(p,t,f)

  np = size(p,2); % number of global nodes
  nt = size(t,2); % number of triangles
  b = zeros(np,1); % right hand side vector

  for K = 1:nt
    loc2glb = t(1:3,K); % local to global index
    x = p(1,loc2glb);
    y = p(2,loc2glb);
    area = polyarea(x,y);
    % element load vector
    bK = [f(x(1),y(1));
          f(x(2),y(2));
          f(x(3),y(3))]/3*area;
    % add element loads to b
    b(loc2glb) = b(loc2glb) + bK;
end
```

Example: full program for the L^2-projection:
find $f_h \in V_h$ with $(f_h, v_h)_{L^2(\Omega)} = (f, v_h)_{L^2(\Omega)}$:

```
function L2Projector2D()

[p,e,t] = initmesh('squareg','hmax',0.1); % create mesh

f = @(x,y) sin(2*pi*x)*sin(2*pi*y);

M = MassAssembler2D(p,t); % assemble mass matrix
b = LoadAssembler2D(p,t,f); % assemble load vector

N = size(p,2); % total number of nodes

projection_f = zeros( N , 1 );

projection_f = M\b; % solve linear system
pdesurf(p,t,projection_f) % plot projection
```