FEM and Sparse Linear System Solving

Lecture 2, Sept 29, 2017: Triangulations in 2D

http://people.inf.ethz.ch/arbenz/FEM17

Peter Arbenz
Computer Science Department, ETH Zürich
E-mail: arbenz@inf.ethz.ch

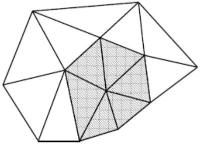
- The finite element method
  - Introduction, model problems.
  - 1D problems. Piecewise polynomials in 1D.
  - 2D problems. Triangulations. Piecewise polynomials in 2D.
  - Variational formulations. Galerkin finite element method.
  - Implementation aspects.
- Direct solvers for sparse systems
  - LU and Cholesky decomposition
  - Sparse matrices
  - Fill-reducing orderings
- Iterative solvers for sparse systems
  - Stationary iterative methods, preconditioning
  - Preconditioned conjugate gradient method (PCG)
  - Incomplete factorization preconditioning
  - Multigrid preconditioning
  - Nonsymmetric problems (GMRES, BiCGstab)
  - Indefinite problems (SYMMLQ, MINRES)

Motivation

- We extend the concept of piecewise polynomial approximation to two dimensions (2D).
- Basic idea: construct spaces of piecewise polynomial functions that are easy to represent in a computer and
- that can be used to approximate more general functions.
- Difficulty: domain must be partitioned into elements, such as triangles, which may be nontrivial if the domain has complex shape.

Triangulations

- For simplicity, we assume that $\Omega \subset \mathbb{R}^2$ is a bounded domain with smooth or polygonal boundary $\partial\Omega$.
- Set of triangles $\{K\}$ defines a triangulation $\mathcal{K}$ of Ω such that
  - $\cup_{K\in\mathcal{K}}\overline{K} = \overline{\Omega}$,
  - the intersection $\overline{K}_1 \cap \overline{K}_2$ of two triangles $K_1, K_2 \in \mathcal{K}$, $K_1 \neq K_2$, is either an edge, a corner, or empty. (Hanging vertices are not allowed.)
- The points where triangle vertices meet are called *nodes*.
- We number the nodes from 1 to n.
- Surrounding any node is a *patch* of triangles that each have that node as a vertex.

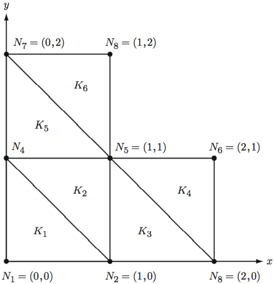


Data structures for triangulations in MATLAB

Represention of triangular mesh with n_p nodes and n_t elements by two matrices:

- Point matrix $\mathbf{p} \in \mathbb{R}^{2\times n_p}$: column j contains coordinates of node N_j .
- connectivity matrix $\mathbf{t} \in \mathbb{R}^{3\times n_t}$: column j contains numbers of the three nodes in triangle K_j

Very coarse triangulation of the L-shaped domain



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

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

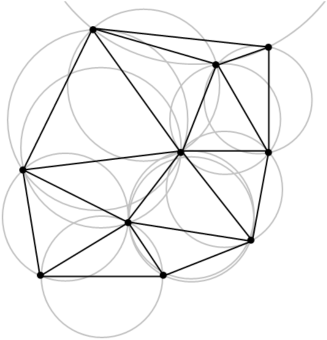
Tetrahedral meshes in 3D

Tetrahedral meshes are (can be) used to partition domains in three dimensions. The way of representing a tetrahedral mesh is similar as with triangular meshes in 2D.

- **p**, point matrix, has 3 rows for the three node coordinates
- **t**, connectivity matrix, has 4 rows containing the four nodes of a tetrahedron

Mesh generation

- In 2D there are efficient algorithms for creating a mesh on quite general domains. Delaunay triangulations ensure that for a set of points the circumcircle associated with each triangle contains no other point in its interior.
  Delaunay triangulations maximize the minimum angle of all the angles of the triangles in the triangulation.

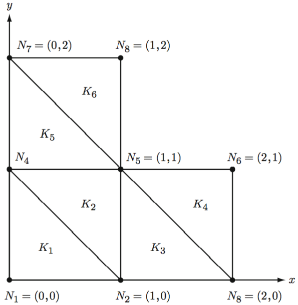


Picture from Wikipedia.

  - PDE-Toolbox in Matlab includes a high quality Delaunay mesh generator for creating high quality triangulations of 2D geometries.

Mesh generation (cont.)

- **g**: Geometry matrix for the L-shaped domain in Matlab



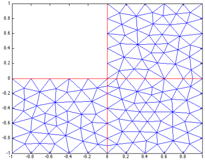
$$\mathbf{g} = \begin{bmatrix} 2 & 2 & 2 & 2 & 2 & 2 \\ 0 & 2 & 2 & 1 & 1 & 0 \\ 2 & 2 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 2 & 2 \\ 0 & 1 & 1 & 2 & 2 & 0 \\ 1 & 1 & 1 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

Row 1 in **g**: '2' for linear boundary segment; row 2,4 (3,5): coordinates of initial (final) endpoint. Row 6 (7): number of geometry piece to left (right) of segment.
For more info see MATLAB docu.

Generate a mesh of the domain g

- Built-in geometries in PDE-Toolbox:
  - `lshapeg`, L-shaped domain
  - `squareg`, square $[-1,1]^2$.
  - `cicrcleg`, the unit radius circle centered at origin.
- More general geometries can be drawn in the PDE-Toolbox GUI. It is initialized by typing **pdetool** at the MATLAB prompt.
- g: geometry matrix; p: point matrix;
  e: edge matrix; t: triangle matrix.

```
g = ‘lshapeg’;
[p,e,t]=initmesh(g,’hmax’,0.1);
pdemesh(p,e,t); % plots
axis square
```



The space of linear polynomials

Meshing a domain allows for a simple construction of piecewise polynomial function spaces.
Let K be a triangle and $\mathbb{P}_1(K)$ space of linear functions on K:

$$\mathbb{P}_1(K) = \{v : v = c_0 + c_1x_1 + c_2x_2, (x_1, x_2) \in K, c_0, c_1, c_2 \in \mathbb{R}\}$$

Any function v in $\mathbb{P}_1(K)$ can be determined by its

$$\{\alpha_0, \alpha_1, \alpha_2\} : \text{nodal values} \quad \alpha_i = v(N_i);\ N_i = (x_1^{(i)}, x_2^{(i)})$$

$$v(x) = \alpha_0\lambda_0 + \alpha_1\lambda_1 + \alpha_2\lambda_2$$

$$\lambda_j(N_i) = \begin{cases} 1, & \text{if } i = j \\ 0, & \text{if } i \neq j \end{cases}$$

On reference triangle $\bar{K}$ with nodes $(0,0), (1,0)$ and $(0,1)$, the nodal basis function for $\mathbb{P}_1(\bar{K})$ are $\lambda_1 = 1 - x_1 - x_2$, $\lambda_2 = x_1$, $\lambda_3 = x_2$.

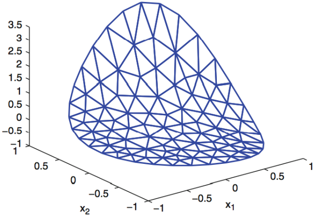
The space of continuous piecewise linear polynomials

The construction of piecewise linear functions on a mesh $\mathcal{K} = \{K\}$: On each triangle K any such function v is simply required to belong to $\mathbb{P}_1(K)$

The space of continuous piecewise linear polynomials V_h

$$V_h = \left\{ v : v \in C^0(\Omega),\ v|_K \in \mathbb{P}_1(K),\ \forall K \in \mathcal{K} \right\}.$$

$C^0(\Omega)$: the space of continuous functions on Ω
$\mathbb{P}_1(K)$: the space of linear polynomials on K



A continuous piecewise linear function v

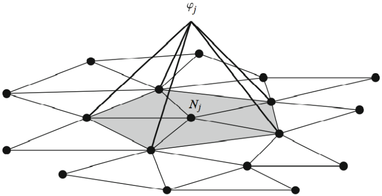
Any function v in V_h can be written as a linear combination of $\{\varphi_i\}_{i=1}^{n_p} \subset V_h$ nodal basis (hat) functions

$$v(x) = \sum_{i=1}^{n_p} \alpha_i \varphi_i(x), \qquad \alpha_i = v(N_i),$$

with

$$\varphi_j(N_i) = \delta_{ij} \equiv \begin{cases} 1, & \text{if } i = j, \\ 0, & \text{if } i \neq j, \end{cases} \qquad i, j = 1, 2, \ldots, n_p.$$

A two-dimensional hat function φ_j on a general triangle mesh



L^2-projection

Definition: The L^2-projection $P_h f$ of $f \in L^2(\Omega)$ onto the space V_h is defined by

$$\int_\Omega (f - P_h f) v_h \, dx = 0, \quad \forall v_h \in V_h. \qquad (*)$$

The definition is equivalent to

$$\int_\Omega (f - P_h f) \varphi_i dx = 0, \quad i = 1, 2, \ldots, n_p.$$

where the φ_i are the basis functions of V_h. Since $P_h f \in V_h$,

$$P_h f = \sum_{j=1}^{n_p} \xi_j \varphi_j$$

ξ_j: the unknown coefficients to be determined.

$$\int_\Omega f\varphi_i dx = \int_\Omega \overbrace{\left(\sum_{j=1}^{n_p} \xi_j \varphi_j\right)}^{P_h f} \varphi_i dx = \sum_{j=1}^{n_p} \xi_j \int_\Omega \varphi_j \varphi_i dx, \quad i = 1, \ldots, n_p.$$

Using the notations:

$$\text{Mass matrix:} \quad m_{ij} = \int_\Omega \varphi_j \varphi_i \, dx, \quad i,j = 1, \ldots, n_p.$$

$$\text{Load vector:} \quad b_i = \int_\Omega f\varphi_i dx, \quad i = 1, \ldots, n_p.$$

The linear system for the unknown coefficients ξ_j is

$$\mathbf{M}\xi = \mathbf{b} \quad \Longleftrightarrow \quad \sum_{j=1}^{n_p} m_{ij}\xi_j = b_i, \quad i = 1, \ldots, n_p.$$

The $n_p \times n_p$ matrix $\mathbf{M}$ is SPD.

Properties of the mass matrix

Theorem: $\mathbf{M}$ is SPD.

Proof:

$$\begin{aligned}
\xi^T \mathbf{M} \xi &= \sum_{i,j=1}^{n_p} m_{ij} \xi_i \xi_j \\
&= \sum_{i,j=1}^{n_p} \left(\int_\Omega \varphi_j \varphi_i \, dx \right) \xi_i \xi_j \\
&= \int_\Omega \left(\sum_{i=1}^{n_p} \xi_i \varphi_i \right) \left(\sum_{j=1}^{n_p} \xi_j \varphi_j \right) \, dx \\
&= \left\| \sum_{i=1}^{n_p} \xi_i \varphi_i \right\|^2 > 0 \qquad \text{if } \xi \neq \mathbf{0},
\end{aligned}$$

since the φ_j are linearly independent by construction. □

Quadrature rules

The integral is approximated by a sum of weights times the values of the integrand at a set of carefully selected quadrature points.

$$\int_K f dx \approx \sum_j w_j f(q_j).$$

$\{q_j\}$: set of quadrature points in triangle $\bar{K}$.
$\{w_j\}$: quadrature weights
Simple quadrature formulas for integrating a continuous function f over a general triangle K with nodes (vertices) N_1, N_2, and N_3 are:

1. Center of gravity rule
2. Trapezoidal rule (aka. corner quadrature formula)
3. A better quadrature formula is 2D mid-point rule

1. The simplest quadrature formula is center of gravity rule:

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

   $|K|$: the area of K. (Variant of 2D mid-point rule)

2. Trapezoidal rule

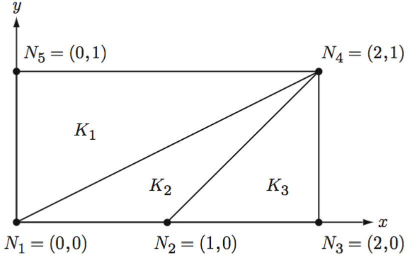
$$\int_K f dx \approx \sum_{i=1}^{3} f(N_i)\frac{|K|}{3}$$

3. 2D mid-point rule

$$\int_K f dx \approx \sum_{1 \le i < j \le 3}^{3} f\left(\frac{N_i + N_j}{2}\right)\frac{|K|}{3}$$

   $(N_i + N_j)/2$: the mid-point of the edge between node number i and j.

Let's consider the small mesh of the rectangle $\Omega = [0,2] \times [0,1]$



We want to compute the mass matrix M

$$M = \int_{\Omega} \begin{bmatrix} \varphi_1\varphi_1 & \varphi_2\varphi_1 & \varphi_3\varphi_1 & \varphi_4\varphi_1 & \varphi_5\varphi_1 \\ \varphi_1\varphi_2 & \varphi_2\varphi_2 & \varphi_3\varphi_2 & \varphi_4\varphi_2 & \varphi_5\varphi_2 \\ \varphi_1\varphi_3 & \varphi_2\varphi_3 & \varphi_3\varphi_3 & \varphi_4\varphi_3 & \varphi_5\varphi_3 \\ \varphi_1\varphi_4 & \varphi_2\varphi_4 & \varphi_3\varphi_4 & \varphi_4\varphi_4 & \varphi_5\varphi_4 \\ \varphi_1\varphi_5 & \varphi_2\varphi_5 & \varphi_3\varphi_5 & \varphi_4\varphi_5 & \varphi_5\varphi_5 \end{bmatrix} dx$$

Break the integral over Ω into a sum of integrals over the triangles $K_i,\ i=1,2,3$.

$$M=\sum_{i=1}^{3}\int_{K_i}\begin{bmatrix}\varphi_1\varphi_1 & \varphi_2\varphi_1 & \varphi_3\varphi_1 & \varphi_4\varphi_1 & \varphi_5\varphi_1\\ \varphi_1\varphi_2 & \varphi_2\varphi_2 & \varphi_3\varphi_2 & \varphi_4\varphi_2 & \varphi_5\varphi_2\\ \varphi_1\varphi_3 & \varphi_2\varphi_3 & \varphi_3\varphi_3 & \varphi_4\varphi_3 & \varphi_5\varphi_3\\ \varphi_1\varphi_4 & \varphi_2\varphi_4 & \varphi_3\varphi_4 & \varphi_4\varphi_4 & \varphi_5\varphi_4\\ \varphi_1\varphi_5 & \varphi_2\varphi_5 & \varphi_3\varphi_5 & \varphi_4\varphi_5 & \varphi_5\varphi_5\end{bmatrix}dx=\sum_{i=1}^{3}M^{K_i}$$

There are three non-zero hat functions on each triangle.

$$M=\int_{K_1}\begin{bmatrix}\varphi_1\varphi_1 & 0 & 0 & \varphi_4\varphi_1 & \varphi_5\varphi_1\\ 0 & 0 & 0 & 0 & 0\\ 0 & 0 & 0 & 0 & 0\\ \varphi_1\varphi_4 & 0 & 0 & \varphi_4\varphi_4 & \varphi_5\varphi_4\\ \varphi_1\varphi_5 & 0 & 0 & \varphi_4\varphi_5 & \varphi_5\varphi_5\end{bmatrix}dx \quad +\int_{K_2}\begin{bmatrix}\varphi_1\varphi_1 & \varphi_2\varphi_1 & 0 & \varphi_4\varphi_1 & 0\\ \varphi_1\varphi_2 & \varphi_2\varphi_2 & 0 & \varphi_4\varphi_2 & 0\\ 0 & 0 & 0 & 0 & 0\\ \varphi_1\varphi_4 & \varphi_2\varphi_4 & 0 & \varphi_4\varphi_4 & 0\\ 0 & 0 & 0 & 0 & 0\end{bmatrix}dx$$

$$+\int_{K_3}\begin{bmatrix}0 & 0 & 0 & 0 & 0\\ 0 & \varphi_2\varphi_2 & \varphi_3\varphi_2 & \varphi_4\varphi_2 & 0\\ 0 & \varphi_2\varphi_3 & \varphi_3\varphi_3 & \varphi_4\varphi_3 & 0\\ 0 & \varphi_2\varphi_4 & \varphi_3\varphi_4 & \varphi_4\varphi_4 & 0\\ 0 & 0 & 0 & 0 & 0\end{bmatrix}dx$$

The computation of mass matrix M is reduce to a series of operations on the triangles.

Computation of the element masses

The computation of the element masses could be done using quadrature. However, there is a much easier way. By induction

$$\int_K \varphi_1^m \varphi_2^n \varphi_3^p dx = \frac{2m!n!p!}{(m+n+p+2)!}|K|$$

on triangle K with its three nodes N_1, N_2, and N_3, and corresponding hat functions φ_1, φ_2, and φ_3.

$$M_{ij}^K = \int_K \varphi_i \varphi_j dx = \frac{1}{12}(1+\delta_{ij})|K| \quad i,j = 1,2,3$$

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

Local-to-global mapping is used when adding the entries of the local element mass matrix M^K to appropriate positions in the global mass matrix M.

Algorithm: assembly of the mass matrix

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

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

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

The load vector b is assembled by summing element load vector b^K over the mesh

$$b_i^K = \int_K f \varphi_i \, dx, \quad i = 1, 2, 3.$$

Using the trapezoidal rule (node quadrature)

$$b_i^K \approx \frac{1}{3} f(N_i)|K|, \quad i = 1, 2, 3.$$

Algorithm: assembly of the mass matrix

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

np = size(p,2);
nt = size(t,2);
b = zeros(np,1);

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

Compute L2-projection of $f = 5x_1x_2$ on the unit square $\Omega = [0,1] \times [0,1]$

```
function L2Projector2D()

g = 'squareg';                        % unit square
[p,e,t] = initmesh(g,'hmax',0.5);    % create mesh
M = MassAssembler2D(p,t);             % assemble mass matrix
b = LoadAssembler2D(p,t,@Foo2);       % assemble load vector
Pf = M\b;                             % solve linear system
pdesurf(p,t,Pf)                       % plot projection


function f = Foo2(x, y)
f = 5*x.*y;
```

Exercise 2

http://people.inf.ethz.ch/arbenz/FEM17/pdfs/exercise2.pdf