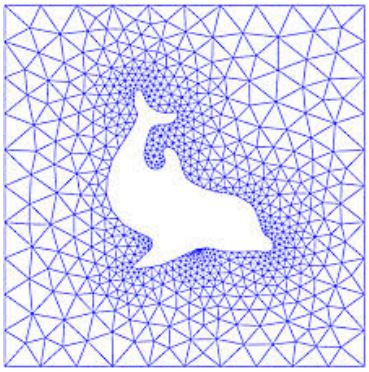
Lecture 4: Finite element method (FEM)

FEM ideology:

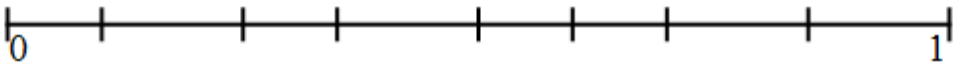
- Discretize the domain: triangles (tetrahedra, quadrilaterals, ...)



- Discretize the functions: piecewise linear on each triangle (in general, piecewise polynomial)
- Discretize the equations: Galerkin, collocation
- Solve: same methods as for FDM

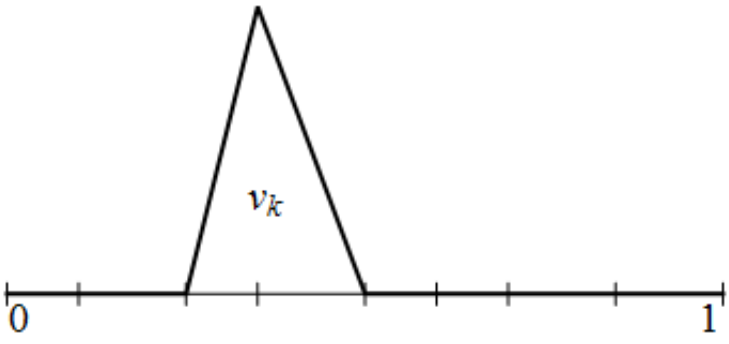
FEM in 1D:

- Start with a domain $\Omega = [0, 1]$
- Mesh (triangulation) $\mathcal{T}$ consisting of
  - segments $[x_i, x_{i+1}]$ (where $x_0, \ldots, x_N$ are nodes)



- Idea: the material consists of small triangles (segments, tetrahedra) that act like basic building blocks of a material
- Functions that are piecewise linear (more precisely, piecewise affine) with respect to $\mathcal{T}$:

$$u_h(x) = \sum_{i=1}^{N} c_i v_i(x)$$



- Such space of function is denoted as $\mathcal{P}^1(\mathcal{T})$
- Often we'll use
$$\mathcal{P}_0^1(\mathcal{T}) := \{u \in \mathcal{P}_0^1(\mathcal{T}) : u|_\Omega = 0\}$$
- Discretize the equations...

Discretization of Equations (Galerkin's method):

First we need the variational (weak) form

Variational form

- Start with the equation (now I like to have minus in front of Δ):
$$\begin{aligned} -\Delta u &= f \\ u|_\Gamma &= 0 \end{aligned}$$
- Multiply the equation by a function v (called the test function) such that $v|_\Gamma = 0$ and integrate:
$$-\int_\Omega (\Delta u)v = -\int_\Gamma \nabla u v \cdot n + \int_\Omega \nabla u \cdot \nabla v = \int_\Omega fv \qquad \forall v$$
- because $v = 0$ on the boundary, we have
$$\int_\Omega \nabla u \cdot \nabla v = \int_\Omega fv \qquad \forall v$$
- This is called the **variational form**

Alternative way for Variational Form

- Recall from Lecture 2 that the energy of interaction of a two-dimensional network of springs was something like (in 1D):
$$E_h^{1\mathrm{d}}(u) = \sum_x k(u(x+h) - u(x))^2/2 - f(x)u(x),$$
where $f(x)$ is the external force acting on the mass x.
- if we let $k = 1/h$, scale f accordingly, and let $h \to 0$ then we have that
$$E(u) = \int_\Omega |\nabla u|^2/2 - \int_\Omega fu$$
(this formula works in any dimension)
- Now, we can ask a question:
$$\text{find } u \text{ such that } E(u) \text{ is minimal}$$

- To find the equation for u, one can take a functional derivative (equivalent to the principle of virtual displacements of mechanics):
$$\langle \partial E(u), v \rangle = \lim_{\epsilon \to 0} \frac{E(u+\epsilon v)}{\epsilon} = \int_\Omega \nabla u \cdot \nabla v - \int_\Omega f v = 0, \qquad \forall v$$
so we derived the variational form without the strong form
- In general, the variational form is obtained directly from the energy minimum principle.

What if we have Neumann or mixed BCs

(BC = boundary condition)

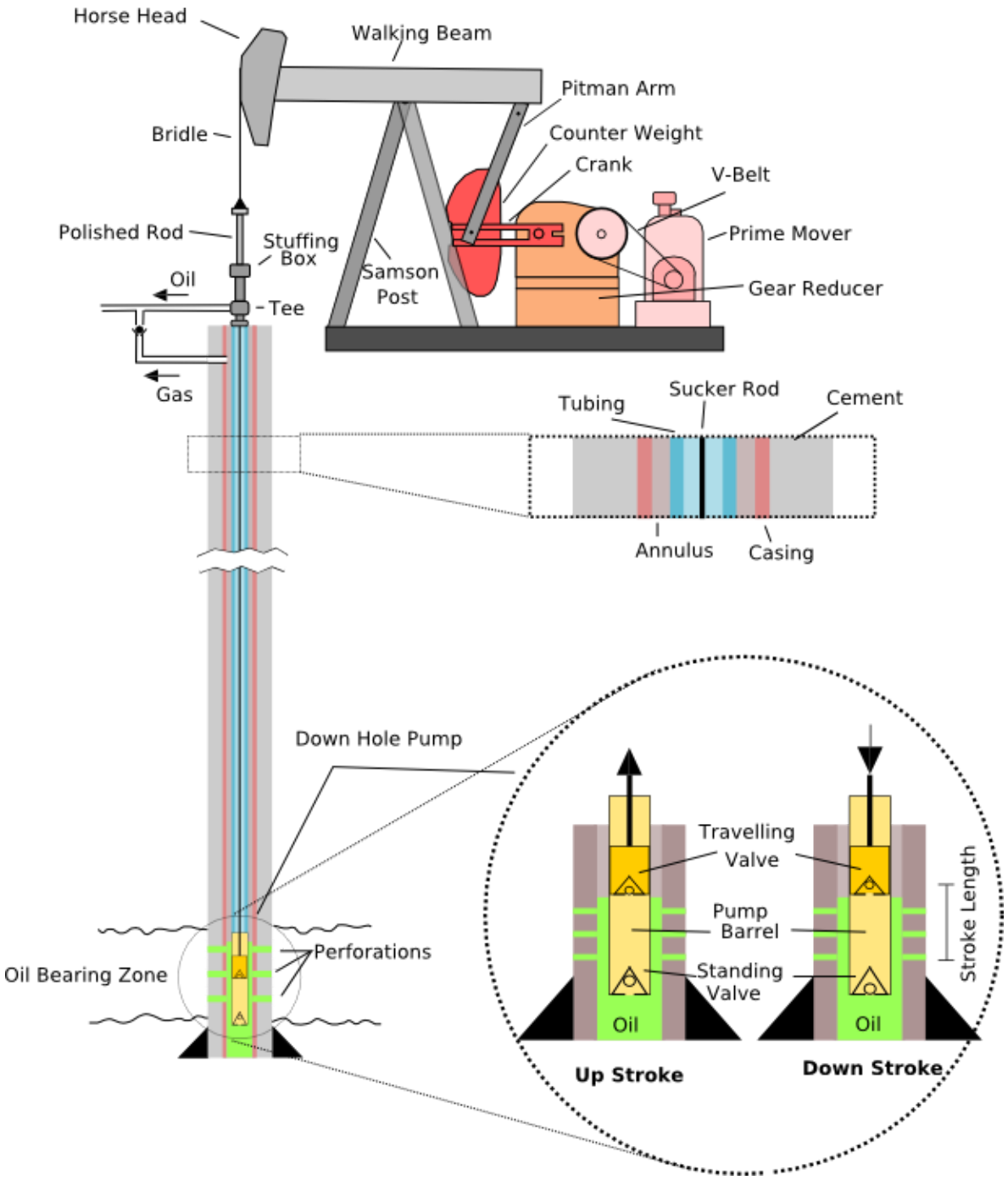
- Suppose we have a problem
$$\begin{aligned} -\Delta u &= f \qquad \text{on } \Omega \\ u &= 0 \qquad \text{on } \Gamma_1 \\ u_n &= 0 \qquad \text{on } \Gamma_2 \end{aligned}$$
where u_n is the normal derivative and $\Gamma_1 \cup \Gamma_2 = \partial\Omega$.
- The trick is then to introduce the space $X = \{\text{function } u : u|_{\Gamma_1} = 0\}$. Then we have, for $v \in X$
$$-\int_\Omega \Delta u v = -\int_{\Gamma_1} u_n v - \int_{\Gamma_2} u_n v + \int_\Omega \nabla u \cdot \nabla v$$
- Magically, the first integral =0 because $v = 0$ on Γ_1 and the second integral =0 because $u_n = 0$ on Γ_2.
- **Same variational formulation**, only spaces are different

Galerkin Method, cont'd

- Denote $A(u,v) = \int_\Omega \nabla u \cdot \nabla v$ and $F(v) := \int_\Omega f v$.
- Discrete equations:
$$A(u_h, v_h) = F(v_h) \qquad \forall v_h \in \mathcal{P}_0^1(\mathcal{T})$$
(That's it!)
- But to implement, we do
$$A(u_h, v_\ell) = F(v_\ell), \qquad \ell = 1, \ldots, N-1$$
(i.e., it is enough to test only with basis functions)
- Then we substitute $u_h = \sum_{k=1}^{N-1} c_k v_k$:
$$\sum_{k=1}^{N-1} c_k A(v_k, v_\ell) = F(v_\ell) \qquad \ell = 1, \ldots, N-1.$$
- Thus, $A(v_k, v_\ell)$ are the elements of the matrix (often called the **stiffness matrix**), and $F(v_\ell)$ are the components of the right-hand side
- It remains to solve the linear system...

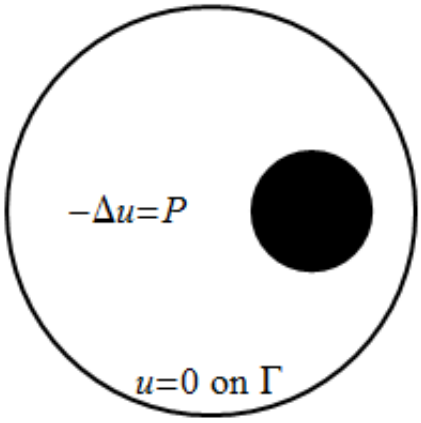
An example of a practical problem

- We want to compute the resistance of pumping the oil (very practical for Russia! :))



The corresponding model problem

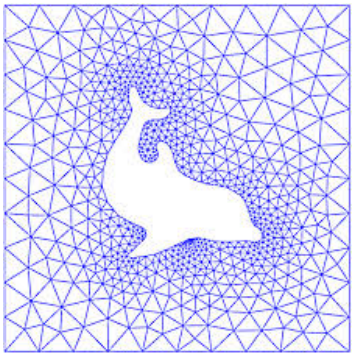
- Cross-section of a tube:



- Here $P = 1$ is the (dimensionless) pressure drop on the tube
- $\int_\Omega u$ will then be the total fluid flux through the tube

FEM in 2D

- The mesh consists of triangles (tetrahedra, quadrilaterals, ...)

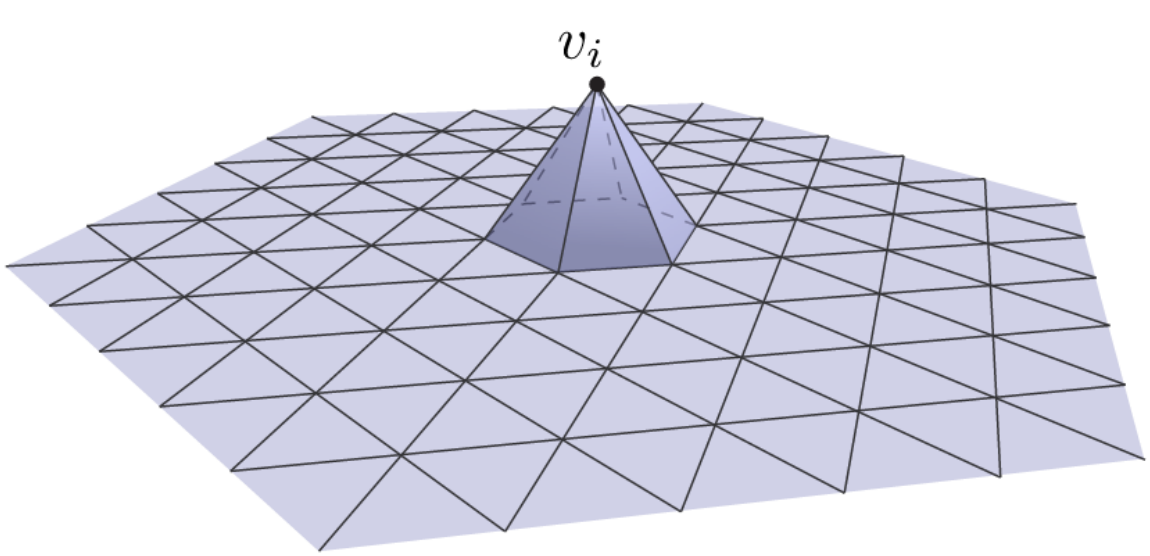


  - The mesh is defined by, typically, 3 arrays:

    `nodes:`
    x_1, y_1
    x_2, y_2
    ...
    x_N, y_N
    (i.e., the first node has coords (x_1, y_1), etc.)

    `triangles:`
    n_1, n_2, n_3
    m_1, m_2, m_3
    ...
    (i.e., the first triangle has nodes n_1, n_2, n_3 as vertices, etc.)

    `faces:`
    n_1, n_2 ...
    (i.e., the first face **on the boundary** has nodes n_1, n_2, etc.)
- Basis functions look like this:



Assembling the stiffness matrix

Classical way:

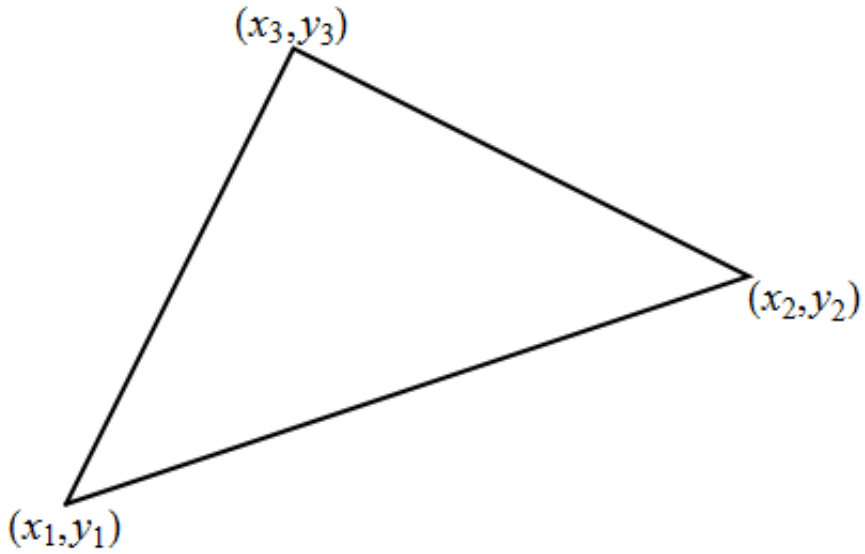
- Notice that $\nabla v_k|$ are piecewise constant:
$$A_{k,\ell} = \int_\Omega \nabla v_k \nabla v_\ell = \sum_{T \in \mathcal{T}} |T| (\nabla v_k|_T)(\nabla v_\ell|_T)$$
where $|T|$ is the volume of T
- The algorithm would the look something like this:

```
for k = 1 to n
for l = 1 to n
  for all T
    if(k and l are nodes of T)
    // otherwise the integral is zero
      A(k,l) += |T|
         *(\nabla v_k|_T)
         *(\nabla v_l|_T)
```

- The loops in the algorithm are often reversed:

```
for all T
  for k, vertices of T
      for l, vertices of T
          A(k,l) += <<as before>>
```

- Advantage: can now loop only over 3 vertices of each triangle, not waisting time
- Now the we only need to consider the geometry of a single triangle



- We have
$$2|T| = \det \begin{pmatrix} x_2 - x_1 & y_2 - y_1 \\ x_3 - x_1 & y_3 - y_1 \end{pmatrix}$$
Let T's nodes being n_1, n_2, n_3. Denote $\eta_i := v_{n_i}$. Then it can be shown
$$\nabla \eta_j = \frac{1}{2|T|} \begin{pmatrix} y_{j+1} - y_{j+2} \\ x_{j+2} - x_{j+1} \end{pmatrix}$$
Here we mean $(x_4, y_4) = (x_1, y_1)$, $(x_5, y_5) = (x_2, y_2)$
- One can do calculations and derive that the matrix $M_{j,k} = |T|(\nabla \eta_j) \cdot (\nabla \eta_k)$ can be evaluated as
$$M = \frac{|T|}{2} G G^T \qquad \text{where} \qquad G = \begin{pmatrix} 1 & 1 & 1 \\ x_1 & x_2 & x_3 \\ y_1 & y_2 & y_3 \end{pmatrix}^{-1} \begin{pmatrix} 0 & 0 \\ 1 & 0 \\ 0 & 1 \end{pmatrix}$$
- the pseudo-code would then look like this:

```
for all T
  calculate M
  for k=1..3
      for l=1..3
          A(triangles(k),triangles(l)) += M(k,l)
```

- More details in the Remarks around 50 lines of Matlab: short finite element implementation (http://www2.mathematik.hu-berlin.de/~cc/cc_homepage/download/1999-AJ_CC_FS-50_Lines_of_Matlab.pdf)

Assembling the right-hand side (forces)

- To evaluate forces, and avoid the exact integration in a triangle
$$\int_T f \eta_i$$
one can approximate

$$\int_T f\eta_i \approx f(x_S, y_S) \int_\Omega \eta_i,$$

where (x_S, y_S) is the barycenter (i.e., center of mass) of the triangle. The code (in 2D) would the look like

```
for all T
calculate f(xS, yS)
for k=1..3
  f(triangles(k)) += 1/3 * area(T) * f(xS, yS)
```

How about boundary conditions?

- What do you think?

How about boundary conditions?

- The code should actually look like this:

```
A = zero matrix for free nodes
for k = free nodes
    for l = free nodes
        for all T
            <<SAME>>
```

This fills only the needed rows & columns of the matrix

- The loops in the algorithm are often reversed:

```
A = zero matrix for all nodes
for all T
  <<SAME>>
remove rows and columns from A corr. to non-free nodes
```

- The **free nodes** are those that are on Γ_1 in

$$\begin{aligned} -\Delta u &= 0 && \text{on } \Omega \\ u &= 0 && \text{on } \Gamma_1 \\ u_n &= 0 && \text{on } \Gamma_2 \end{aligned}$$

- Alternatively, one can **replace** the corresponding row and column by

$$\begin{pmatrix} \cdot & 0 & \cdot & \cdot \\ 0 & 1 & 0 & 0 \\ \cdot & 0 & \cdot & \cdot \\ \cdot & 0 & \cdot & \cdot \end{pmatrix}$$

- **Exercise**: think why

Algorithm

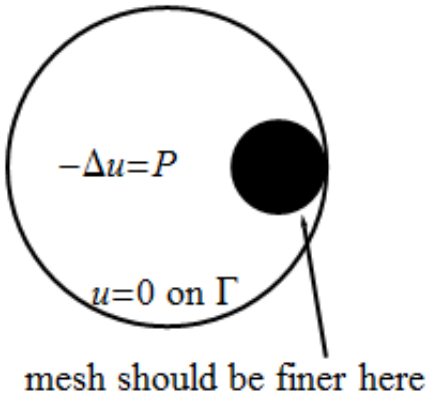
- Let's think together what the algorithm would look like (and draft on a whiteboard)

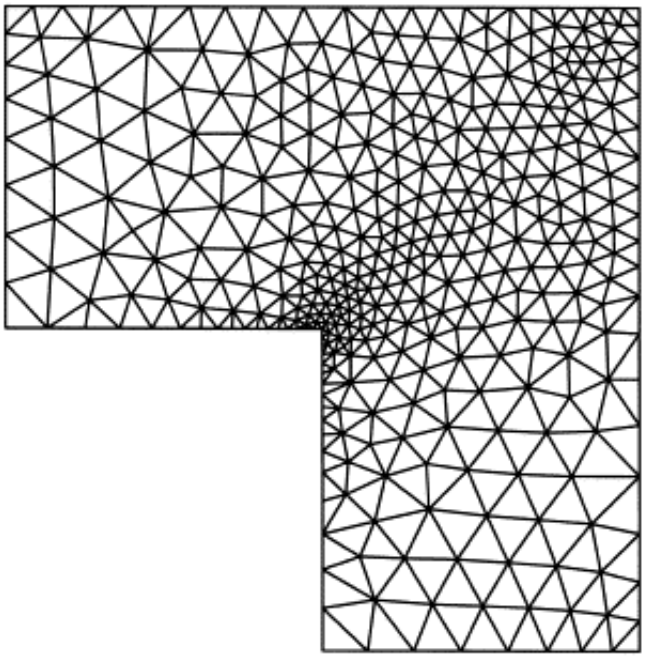
More advanced stuff

Error analysis

- If u is the exact solution and u_h is the approximate solution then one can prove that
$$\|\nabla u_h - \nabla u\|_{L^2} \leq h\|\nabla^2 u\|_{L^2},$$
where h is the **maximal** size of the triangle and $\nabla^2 u$ is the Hessian matrix (http://en.wikipedia.org/wiki/Hessian_matrix).
- Comes from a truly beautiful argument:
  - One can show that u_h is the minimizer of
$$\int \nabla(v_h - u) \cdot \nabla(v_h - u) = \|\nabla v_h - \nabla u\|_{L^2}^2$$
over all $v_h \in \mathcal{P}_1(\mathcal{T})$ (let's prove it).
  - It remains to invoke a geometry-related argument that given u there exists v_h such that $\|\nabla v_h - \nabla u\| \leq h\|\nabla^2 u\|$. This is achieved with, e.g., the Clément interpotation (warning: google for that on your own risk!)

Mesh refinement / Adaptivity





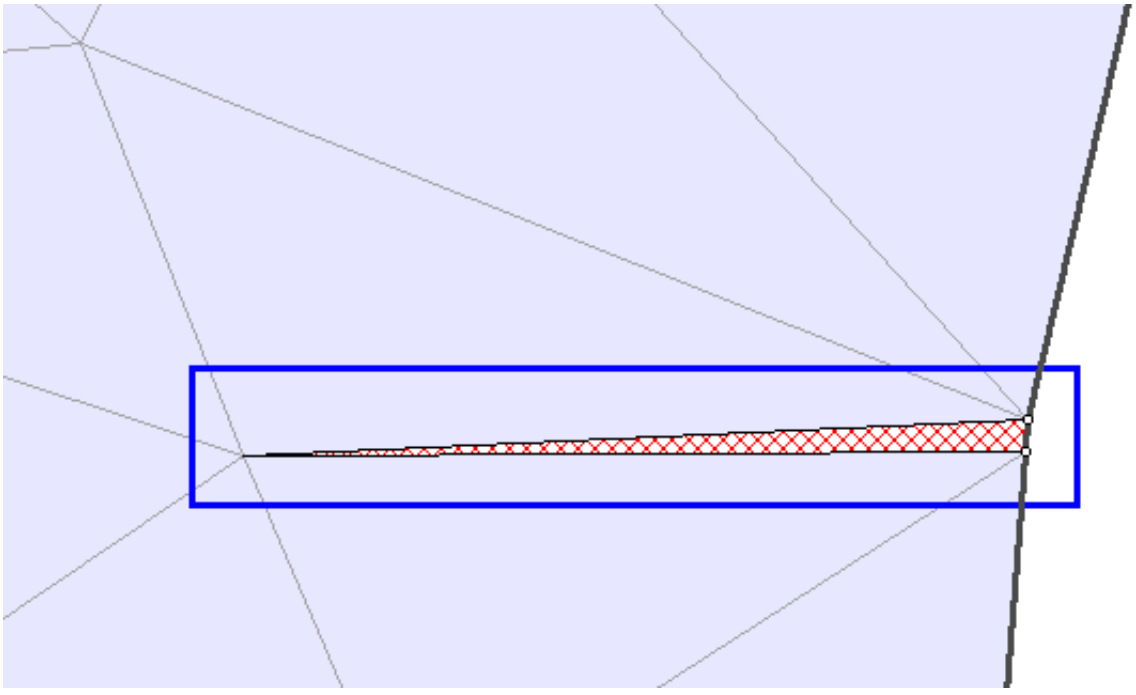
- Often done through the a posteriori error estimate, e.g.,

$$\|\nabla u_h - \nabla u\|^2 \leq \sum_{\text{edge}} h_{\text{edge}} \left[\frac{\nabla u_h}{h_{\text{edge}}} \right]^2_{\text{edge}} \Bigg|$$

  where $[\bullet]_{\text{edge}} \Big|$ is the jump across the edge (recall that $\nabla u_h \Big|$ is piecewise constant)

Mesh quality

- Mesh should not have very sharp angles (otherwise the approximation will be bad)!



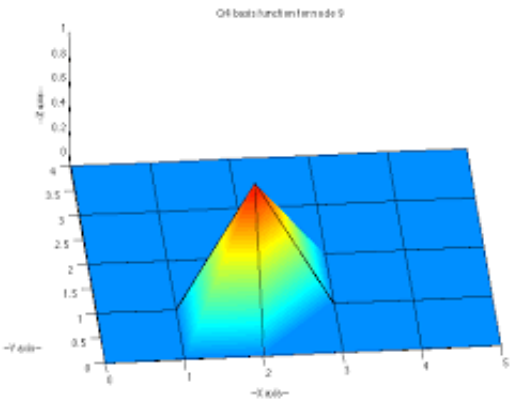
Collocation

- If we have non-constant coefficients, we use the quadrature rules

$$\int_T k(x,y)\nabla u \cdot \nabla v \approx \sum_i w_i k(x_i, y_i)\nabla u(x_i, y_i) \cdot \nabla v(x_i, y_i)$$

- must be careful to take the quadrature rule of sufficiently high order

Quad(rilateral) elements:



High-order elements

- $\mathcal{P}^p(\mathcal{T})$: polynomials of power p
- hp-refinement

A good source for these is
C. Schwab, "p- and hp- Finite Element Methods: Theory and Applications to Solid and Fluid Mechanics", Clarendon Press, 1998.

More stuff:

- Discontinous Galerkin, $\mathcal{P}^0$ elements
- mixed FEM, generalized FEM, extended FEM, what-not FEM...

Adv/Disadv of FEM

- Good for simple physics, solid mechanics
- Good energy (and other) conservation properties

But...

- Not easy for fluid flow (must work hard for stability)
- sometimes non-monotonicity

Questions?