FEM

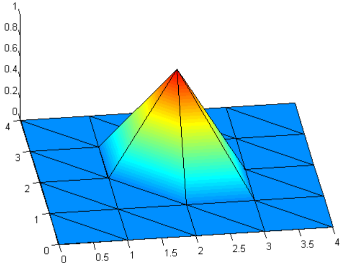
Sebastian Müller, Fritz Schelten

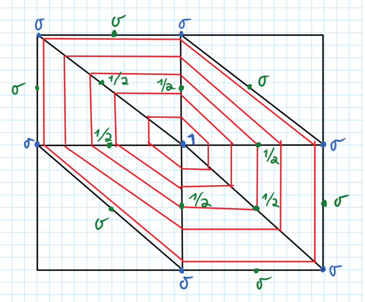
Juni 19, 2017

Überblick

Basisfunctions, Shapefunctions

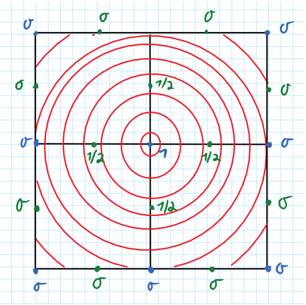
Triangle elements





Basisfunctions, Shapefunctions

Square elements



Stiffness matrix

Calculation for rect. triangles and squares

```
for all ∇φ_i
  for all ∇φ_j = ∇φ_1, ..., ∇φ_i
    a_ij = 0
    if basis nodes of φ_i, φ_j are neighbours
            for all shape functions s_i of ∇φ_i
              for all shape functions s_j of ∇φ_j
              if (domain(s_i) == domain(s_j))
                a_ij = a_ij + ∫_{d(s_i)} ∇φ_i · ∇φ_j (+φ_i · φ_j) dx;
              end
            end
        end
    A(i,j)=A(j,i)=a_ij;
  end
end
```

all gradients of basis functions are calculated before

Stiffness matrix

Calculation for arb. triangles

```
for all ∇φ_i
  for all ∇φ_j
    a_ij = 0
        for all shape functions s_i of ∇φ_i
          for all shape functions s_j of ∇φ_j
            if (domain(s_i) == domain(s_j))
              a_ij = a_ij + ∫_{d(s_i)} ∇φ_i · ∇φ_j(+φ_i · φ_j)dx;
            end
          end
        A(i,j)=a_ij;
  end
end
```

all gradients of basis functions are calculated before

Evaluation of solution

for rectangular triangles

Class solution assembles a shape function for each domain by summing up weighted corresponding shape functions of all basis functions.

Evaluation is done by detecting the domain related to (x, y)

```
interval = ceil( $x/nodeDistance_x; y/nodeDistance_y$ )
domainIndex = $2 \cdot [((interval_y - 1) \cdot meshIntervals_y) + interval_x]$
u = solution.shapeScalarFunctions(domainIndex).evaluate(x,y);
if (u == 0) then u = solution.shapeScalerFunctions(domainIndex-1).evaluate(x,y);
```

Evaluation of solution

for squares

Class solution assembles a shape function for each domain by summing up weighted corresponding shape functions of all basis functions.

Evaluation is done by detecting the domain related to (x, y)

```
interval = ceil(x/nodeDistance_x; y/nodeDistance_y)
domainIndex = ((interval_y - 1) · meshIntervals_y) + interval_x
u = solution.shapeScalarFunctions(domainIndex).evaluate(x,y);
```

A posteriori estimator

Calculation

$r_K(u_h) := (f + \nabla u_h)|_K$ and $r_E(u_h) := [\eta_E \cdot \nabla u_h]_E$

$\eta := \left[\sum_K \eta_K^2\right]^{1/2}$, where $\eta_K^2 := h_K^2 \|r_K^2\|_{0,K}^2 + \frac{1}{2} \sum_{E \subset K} h_E \|r_E\|_{0,E}^2$

where K are domains and E are edges.

This uses the shape scalar function array of the solution.