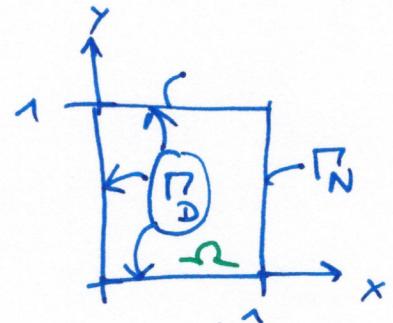


- Programmierübung 2 -

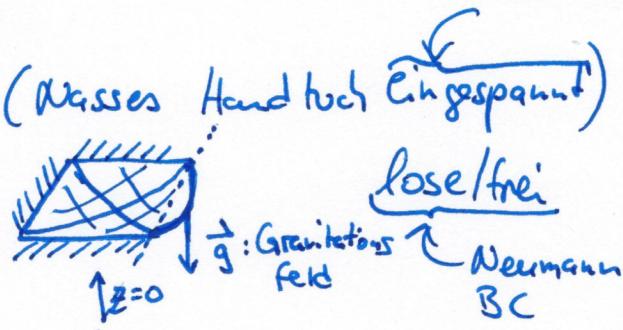
$$\begin{cases} -\Delta u = f & (=u_D) \text{ (vgl. } f=0 \Rightarrow \text{Laplace Problem)} \\ u = 0 \text{ auf } \Gamma_D \\ \partial_n u = 0 \text{ auf } \Gamma_N & (=h) \end{cases}$$

$$\Omega = [0,1] \times [0,1]$$



Physikalische Beispiele:

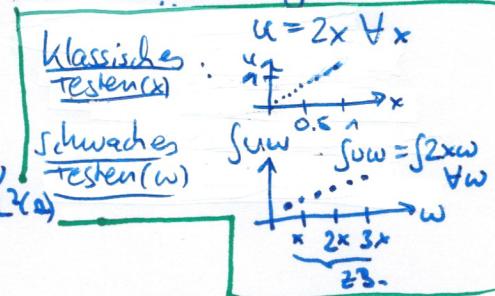
- 2D-Wärmeleitung mit f als Quellterm
- Verschiebung im Strukturmech. Sinn im Gravitationsfeld (wasser Handtuch eingespannt)



- Klassische Lösung $u \in C^2(\Omega) \cap C^0(\bar{\Omega})$ erfüllt: $-\Delta u = f$
- Aber: Δu muss stetig sein (Restriktion), daher schwache Formulierung (kein Punktweise Betrachtung, sondern Funktionenweise)
- Multiplikation mit Testfunktion w (weighting function) & Integration:

Find $u \in H^2(\Omega)$ sodass $\left\{ \begin{array}{l} -\Delta u = f \Leftrightarrow \\ - \int_{\Omega} (\Delta u) w = \int_{\Omega} f w \quad \text{EL}(u) \end{array} \right.$

(mit $u = u_D$ auf Γ_D und $\frac{\partial u}{\partial n} = h$)



→ Nun muss Δu nicht mehr stetig, sondern nur noch quadratisch integrierbar.
 $\Rightarrow u \in H^2(\Omega)$

- Probleme:
 - I) u, w aus versch. Räumen. Für Implementierung wären gleiche Räume nice.
 - II) Wie bringen wir die Randbed. mit ein? (NM, DL)
 - III) $u \in H^2(\Omega)$ schon relativ restriktiv im Sinne der Regularität.

(2)

• Erste Greensche Identität:

$$\int_{\Omega} (\underline{w} \nabla^2 u + \nabla w \cdot \nabla u) d\Omega = \int_{\partial\Omega} w \frac{\partial u}{\partial \vec{n}} dS = \int_{\partial\Omega} (\underline{w} \nabla u) \cdot \vec{n} dS = \int_{\partial\Omega} \nabla \cdot (\underline{w} \nabla u) d\Gamma$$

→ vgl.
partielle
Integration

kommt von der:
"Produktregel"
 $\nabla \cdot (\underline{w} \nabla u) \stackrel{\text{"Intuition (PR)"} \atop \text{div}(\underline{w} \nabla u)}{=} \nabla(\underline{w}) \cdot \nabla u + \underline{w} \cdot \nabla(\nabla u)$
 (aber andere Schreibweise niv.)

• Analogie

mit part. Integration:

$$\int_{\Omega} w \Delta u = \int_{\partial\Omega} (\underline{w} \nabla u) \cdot \vec{n} dS \int_{\Omega} \nabla w \cdot \nabla u$$

vs. $\int_0^1 f' g' = [fg]_0^1 - \int_0^1 f' g$ (1D)
 ↪ minus eine Dimension

• Schwache Formulierung:

$$-\int_{\Omega} w \Delta u = \int_{\Omega} fw$$

$$\Leftrightarrow -\int_{\partial\Omega} (\underline{w} \nabla u) \cdot \vec{n} + \int_{\Omega} \nabla w \cdot \nabla u = \int_{\Omega} fw$$

$$\Leftrightarrow \int_{\Omega} \nabla w \cdot \nabla u d\Omega = \int_{\Omega} fw d\Omega + \int_{\Gamma} w \frac{\partial u}{\partial \vec{n}} d\Gamma \quad \forall w \in V = \{w \in H^1(\Omega) \mid w|_{\Gamma_D} = 0\}$$

R: (Weil nur ∇w im Integral)

Find $u \in H^1(\Omega)$ with: $\int_{\Omega} \nabla w \cdot \nabla u d\Omega = \int_{\Omega} fw d\Omega + \int_{\Gamma} w \frac{\partial u}{\partial \vec{n}} d\Gamma$

with $u|_{\Gamma_D} = u_D$

Setze $w = 0$ weil $\frac{\partial u}{\partial \vec{n}}$ nicht bekannt auf Γ_D →

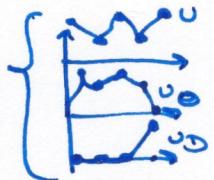
• Galerkin Formulierung:

(Selber Ansatzraum für u und Testraum)

→ Aufteilung von $u := u_0 + u_D$

↳ weil $S := \{u \in H^1 : u|_{\Gamma_D} = 0\}$ unknown

nicht mal Vektorspace



$$\Rightarrow \text{Finde } u \in V = \{u \in S : u|_{\Gamma_D} = 0\} \text{ mit: } \int_{\Omega} \nabla w \cdot \nabla(u_0 + u_D) = \int_{\Omega} fw + \int_{\Gamma} w \frac{\partial u}{\partial \vec{n}} d\Gamma$$

Linerarität

$$\Leftrightarrow \int_{\Omega} \nabla w \cdot \nabla u_0 = \int_{\Omega} fw + \int_{\Gamma} w \frac{\partial u}{\partial \vec{n}} d\Gamma - \int_{\Omega} \nabla w \cdot \nabla u_D \quad \forall w \in V = \{w \in H^1 : w|_{\Gamma_D} = 0\}$$

SAME SPACES

important for e.g. (ax-Hilfssatz Lemma)

Schwache (Variations-) Formulierung in Galerkin Form:

$$\text{Find } u_0 \in V := \{u_0 \in H^1(\Omega) : u_0|_{\Gamma_D} = 0\} \text{ sodass: } \int_{\Omega} \nabla w \cdot \nabla u_0 = \int_{\Omega} f w + \int_{\Gamma_N} w \frac{\partial u}{\partial n} d\Gamma_N - \int_{\Gamma_D} w \cdot \nabla u_0 \quad \forall w \in V$$

- Projection onto Lower-Dimensional Function Spaces (Approximation to allow solving with Computer Numerics)

$$\rightarrow V^h \subset V \quad V^h = \left\{ w \in H^1(\Omega) : w|_{\Gamma_D} \in P_m(\Omega^e), w=0 \text{ on } \Gamma_D \right\}$$

↑ Polynomials of degree m e.g.

$$\Rightarrow \text{Find } u^h \in V^h \text{ s.t. } \forall w^h \in V^h : \underbrace{\int_{\Omega} \nabla w^h \cdot \nabla u^h}_{a(w^h, u^h)} = (w^h, f) + \left(w^h, \frac{\partial u}{\partial n} \right)_{\Gamma_N} - a(w^h, u^h)$$

→ Error not so large (→ Cea's Lemma)

$$\text{Diskrete Formulierung (ohne RB erstmal): } \left\{ \begin{array}{l} u = u_0 + u_g \text{ eingesetzt} \\ \text{find } u \in V^h \\ a(w^h, u^h) = (w^h, f) \quad \forall w^h \in V^h \end{array} \right.$$

$$\bullet u^h(\underline{x}) := \sum_{B \in \mathcal{N}/\mathcal{N}_D} N_B(\underline{x}) \cdot u_B \quad \text{mit } N_B(\underline{x}) \text{ Ansatzfunktion}$$

zu finden!

$$\mathcal{N} = \{1, \dots, n_n\}, \mathcal{N}_D \stackrel{\text{nodes}}{=} \text{nodes on } \Gamma_D$$

- Problem: • Schw. Diskrete Formulierung muss $\forall w \in V^h$, aber V^h unendlich groß \Rightarrow viel Aufwand und unnötig, denn wir können eine Basis von V^h testen. Durch Linearität des Problems kann mit Linearkombination alle $w \in V^h$ dargestellt werden, die die schw. F erfüllen, sofern \rightarrow die es Basis tut.

Testen von Basis ausreichend! $w(\underline{x}) := \sum_{A \in \mathcal{N}/\mathcal{N}_D} N_A(\underline{x}) \cdot w_A$

$$\rightarrow \{w_A\}^n = \{1, 0, \dots, 0\} \quad \left\{ \begin{array}{l} \vdots \\ \{w_A\}^{n_n} = \{0, \dots, 0, 1\} \end{array} \right\} \text{ Sets}$$

Einsetzen der Ansätze in SF:

$$f(\omega^h, f) = \int_h \omega^h f$$

(4)

$$a\left(\sum_{3 \in \mathcal{N}/\mathcal{N}_0} N_3(x) u_3, \sum_{A \in \mathcal{N}/\mathcal{N}_D} N_A(x) u_A\right) = f\left(\sum_{A \in \mathcal{N}/\mathcal{N}_D} N_A(x) u_A, f\right) \quad \forall \{u_A\}_{A=1}^{n_n}$$

Sets of
coeffs

Linearität
+ Testen der Basis
 \Leftrightarrow

$$\sum_{3 \in \mathcal{N}/\mathcal{N}_D} a(N_3(x), N_A(x)) \cdot u_3 = f(N_A(x), f) \quad \forall A = 1 \dots n_n$$

$$\begin{aligned} \Leftrightarrow & \left\{ \begin{array}{l} a(N_1(x), N_1(x)) \cdot u_1 + \dots + a(N_{n_n}(x), N_1(x)) \cdot u_{n_n} = f(N_1(x), f) \quad [A=1] \\ \vdots \\ a(N_1(x), N_{n_n}(x)) \cdot u_1 + \dots + a(N_{n_n}(x), N_{n_n}(x)) \cdot u_{n_n} = f(N_{n_n}(x), f) \quad [A=n_n] \end{array} \right. \end{aligned}$$

Nodewise

$$\begin{aligned} \text{LGS} \Leftrightarrow & \left[\int_{\Omega} \nabla N_1(x) \cdot \nabla N_1(x) \dots \int_{\Omega} \nabla N_{n_n}(x) \cdot \nabla N_1(x) \right] \cdot \begin{bmatrix} u_1 \\ \vdots \\ u_{n_n-1} \\ u_{n_n} \end{bmatrix} = \begin{bmatrix} \int_{\Omega} N_1(x) - f \\ \vdots \\ \int_{\Omega} N_{n_n}(x) - f \end{bmatrix} \\ & \underbrace{\left[\int_{\Omega} \nabla N_1(x) \cdot \nabla N_{n_n}(x) \dots \int_{\Omega} \nabla N_{n_n}(x) \cdot \nabla N_{n_n}(x) \right]}_{[n_{\text{nodes}} \times n_{\text{nodes}}]} \quad \underbrace{\begin{bmatrix} u_1 \\ \vdots \\ u_{n_n-1} \\ u_{n_n} \end{bmatrix}}_{[n_{\text{nodes}} \times 1]} \\ & \Rightarrow \text{Das ist zu finden.} \end{aligned}$$

Shiftless Matrix K

N nur in
allen Nachbar
 Δ 's definiert

Elementweise Betrachtung der Integrale
(Vorteil: Spart Rechenzeit, weil N_i-Support klein)

$$K^e = \left[\int_{\Delta e} \nabla N_1^e(x) \cdot \nabla N_1^e(x) \dots \int_{\Delta e} \nabla N_1^e \cdot \nabla N_3^e \right]$$

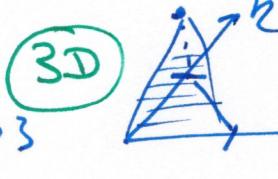
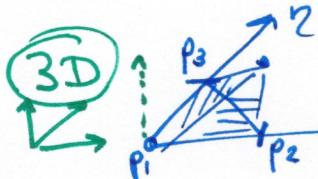
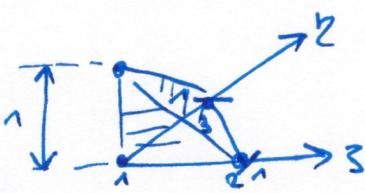
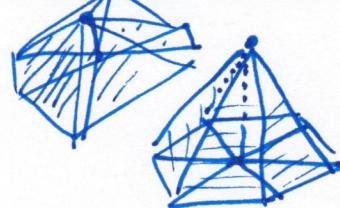
Sind nur 9 Werte, da
nur jeweils 3 Funktionen
im Δ -Element
existieren!

→ Loop over Elements

• Stückweise Lineare Funktionen auf Δ -Gittern

Shape Funktionen normalerweise def. auf: (warum, sehen wir später)

→ Referenzelement: Ω_{ref}



P1-Elemente

$$\hat{N}_1(\zeta, \eta) = 1 - \zeta - \eta$$

$$\hat{N}_2(\zeta, \eta) = \zeta$$

$$\hat{N}_3(\zeta, \eta) = \eta$$

Mit Hab
ML:

(N_i müssen
definiert
werden)

$$\nabla \hat{N}_1(\zeta, \eta) = \begin{pmatrix} -1 \\ -1 \end{pmatrix}$$

$$\nabla \hat{N}_2 = \begin{pmatrix} 1 \\ 0 \end{pmatrix}$$

$$\nabla \hat{N}_3 = \begin{pmatrix} 0 \\ 1 \end{pmatrix}$$

(am besten
hervorheben)

→ Transformation von Dreieckselement Ω_e zu Ω_{ref} :

$$\underline{x} = (x, y)$$

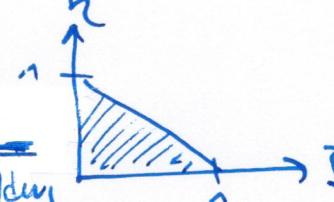
$$\cdot \int_{\Omega_e} \nabla_x N_i(x, y) \cdot \nabla_x N_j(x, y) d\Omega_e \stackrel{\substack{\text{vgl.} \\ \text{vorherige} \\ \text{HA}}}{=} \int_{\Omega_{\text{ref}}} \nabla_{\underline{x}} N_i(\underline{x}(\zeta, \eta)) \cdot \nabla_{\underline{x}} N_j(\underline{x}(\zeta, \eta)) |\mathcal{J}| d\Omega_{\text{ref}}$$



Vektorwertige
Abbildung

$$\underline{\Phi}_k(\zeta, \eta) = \begin{pmatrix} x(\zeta, \eta) \\ y(\zeta, \eta) \end{pmatrix}$$

Richtung des Abbildung



$$\mathcal{J} = \det \underline{\Phi}_k(\zeta, \eta)$$

$$\rightarrow \mathcal{J} = \det \underline{\Phi}_k(\zeta, \eta) = \begin{bmatrix} \frac{\partial x}{\partial \zeta} & \frac{\partial x}{\partial \eta} \\ \frac{\partial y}{\partial \zeta} & \frac{\partial y}{\partial \eta} \end{bmatrix}$$

→ Problem: $\nabla_x N_i(\zeta, \eta)$ Gradient ist in physikalischen (x, y)-Koords.

→ Aber wir betrachten:

$$\nabla_{\underline{x}} N_i(\underline{x}(\zeta, \eta)) = \nabla_{\underline{x}} N_i(x(\zeta, \eta), y(\zeta, \eta))$$

$$\begin{bmatrix} \partial_x N_i(x(\zeta, \eta)) \\ \partial_y N_i(x(\zeta, \eta)) \end{bmatrix} \stackrel{\text{Kettenregel}}{=} \begin{bmatrix} \frac{\partial N_i(x(\zeta, \eta))}{\partial x} \frac{\partial x}{\partial \zeta} + \frac{\partial N_i(x(\zeta, \eta))}{\partial y} \frac{\partial y}{\partial \zeta} \\ \frac{\partial N_i(x(\zeta, \eta))}{\partial x} \frac{\partial x}{\partial \eta} + \frac{\partial N_i(x(\zeta, \eta))}{\partial y} \frac{\partial y}{\partial \eta} \end{bmatrix}$$

$$= \begin{bmatrix} \frac{\partial x}{\partial \zeta} & \frac{\partial x}{\partial \eta} \\ \frac{\partial y}{\partial \zeta} & \frac{\partial y}{\partial \eta} \end{bmatrix} \cdot \begin{bmatrix} \frac{\partial N_i(x(\zeta, \eta))}{\partial x} \\ \frac{\partial N_i(x(\zeta, \eta))}{\partial y} \end{bmatrix} = \mathcal{J}^T \cdot \nabla_{\underline{x}} N_i(x(\zeta, \eta))$$

$$\Leftrightarrow \nabla_x N_i(x(\zeta, \eta), y(\zeta, \eta)) = \mathcal{J}^{-T} \nabla_{\underline{x}} N_i(\zeta, \eta) \quad (\text{HA!})$$

$\nabla_{\underline{x}} \hat{N}_i(\zeta, \eta)$
Referenz
Shape Funktion

• Element integral lautet also ganz einfach:
 $x = \begin{pmatrix} x \\ y \end{pmatrix}$

⑥

$$\int_{\Omega_e} \nabla_x N_i(x, y) \cdot \nabla_x N_j(x, y) d\Omega_e = \int_{\Omega_{ref}} \underbrace{\nabla_x N_i(x(3,\eta), y(3,\eta)) \cdot \nabla_x N_j(x(3,\eta), y(3,\eta)) \cdot |J|}_{= J^{-T} \nabla_3 \hat{N}_i(3,\eta)} d\eta$$

$$= \int_{\Omega_{ref}} \left[J^{-T} \nabla_3 \hat{N}_i(3,\eta) \right] \cdot \left[J^{-T} \nabla_3 \hat{N}_j(3,\eta) \right] \cdot |J| d\eta$$

mit Jacobimatrix:

$$\Phi_k(\underline{\eta}) = \begin{bmatrix} 1 & 1 \\ (\alpha_2 - \alpha_1) & (\alpha_3 - \alpha_1) \end{bmatrix} \cdot \underline{\eta} + \underline{\alpha}_1 \quad (\text{HAS})$$

$$\Leftrightarrow \underline{\Phi}_k(3,\eta) = \begin{bmatrix} \alpha_2 x - \alpha_1 x & \alpha_3 x - \alpha_1 x \\ \alpha_2 y - \alpha_1 y & \alpha_3 y - \alpha_1 y \end{bmatrix} \cdot \begin{pmatrix} 3 \\ \eta \end{pmatrix} + \begin{pmatrix} \alpha_1 x \\ \alpha_1 y \end{pmatrix} = \begin{pmatrix} x(3,\eta) \\ y(3,\eta) \end{pmatrix}$$

$$\Rightarrow J = \begin{bmatrix} \frac{\partial x(3,\eta)}{\partial \eta} & \frac{\partial x(3,\eta)}{\partial \eta} \\ \frac{\partial y(3,\eta)}{\partial \eta} & \frac{\partial y(3,\eta)}{\partial \eta} \end{bmatrix} = \begin{bmatrix} (\alpha_2 x - \alpha_1 x) & (\alpha_3 x - \alpha_1 x) \\ (\alpha_2 y - \alpha_1 y) & (\alpha_3 y - \alpha_1 y) \end{bmatrix}$$

hann easy!
berechnet
werden,
hängt nur
vom Mesh ab!

MATLAB: $J^{-T} = \text{meshops.calcInverseJacOfTriangle}(i)$

$|J| = \text{meshOps.calcJacobianDeterminantOfTriangle}(i)$

mit bekannten Ableitungen der Shape Fkt. auf Ref- Δ

$$\nabla_3 \hat{N}_i(3,\eta) = \begin{bmatrix} \nabla_3 \hat{N}_1(3,\eta) \\ \vdots \\ \nabla_3 \hat{N}_3(3,\eta) \end{bmatrix}^T = \begin{bmatrix} -1 & -1 \\ 1 & 0 \\ 0 & 1 \end{bmatrix}$$

Drei, nicht x_i

immer gleich ∇_0
(spart Rechenzeit)
(daher überhaupt Trafo...)

Einziges Problem: Integral $\int_{\Omega_e} (\dots) \nabla_3 \hat{N}_i(3,\eta) \cdot \nabla_3 \hat{N}_j(3,\eta) (\dots) d\eta$

muss numerisch gelöst werden.

\Rightarrow Quadratur!

Quadratur: Numerisches Integrieren

$$\int_{\Omega} \nabla_3 \hat{N}_i(3, \eta) \cdot \nabla_3 \hat{N}_j(3, \eta) \approx \sum_{l=1}^{n_{\text{quad}}} \nabla_3 \hat{N}_i(3_l, \eta_l) \cdot \nabla_3 \hat{N}_j(3_l, \eta_l) \cdot w_l$$

Weighting
Faktor

Integration ist die gewichtete Summation

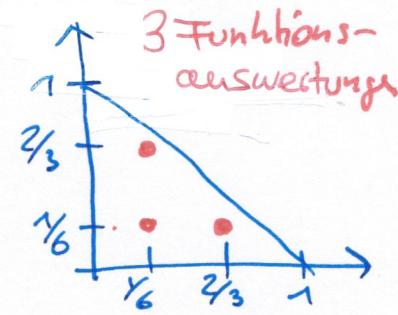
Von simplen Funktionsauswertungen. Diese bleiben sogar immer gleich weil auf Referenzbereich \Rightarrow Vorberechnung \Rightarrow genial!

Abhängigkeit der Genauigkeit von Geometrie und n_{quad} : z.B. Referenzdreieck

$$\int_0^1 \int_0^{1-\eta} g(3, \eta) d3 d\eta$$

$n_{\text{quad}} = 3$

$$\approx \frac{1}{6} \cdot g\left(\frac{1}{6}, \frac{1}{6}\right) + \frac{1}{6} g\left(\frac{1}{6}, \frac{4}{6}\right) + \frac{1}{6} g\left(\frac{4}{6}, \frac{1}{6}\right)$$



\rightarrow Gewichte $\{w_l\}_{l=1}^{n_{\text{quad}}}$ und Stützstellen $\{(3_l, \eta_l)\}_{l=1}^{n_{\text{quad}}}$ aus Literatur oder NS-Berechnungen.

\rightarrow Bei uns ist alles schon gemacht

\Rightarrow MATLAB: $[w, p, n] = \text{meshops.IntegrationRuleOfTriangle}()$

$\uparrow \{(3_l, \eta_l)\}_{l=1}^{n_{\text{quad}}}$

Rechte Seite auch elementweise

$$\int_{\Omega_e} f(x, y) \cdot N_i(x, y) \stackrel{\text{Tetra.}}{=} \int_{\Omega_{\text{ref}}} f(x(3, \eta), y(3, \eta)) \cdot \underbrace{N_i(x(3, \eta), y(3, \eta)) \cdot |J|}_{\Omega_{\text{ref}}} d\eta$$

$$= \sum_{l=1}^{n_{\text{quad}}} f(x(3_l, \eta_l)) \cdot \hat{N}_i(3_l, \eta_l) \cdot |J| \cdot w_l$$

MATLAB: mit $x_{\cdot} = \text{meshops.calcMappedIntPointOfTri}(..., p[l, :])$

Dirichlet Randwerte behaupt

des globalen LGS -

- Nachträglich die Zeile der Dirichlet-Nodes nur auf Diagonale = 1 setzen und Rechte Seite dieser Zeile im LGS gleich dem Dirichlet-Randwert setzen.

Pseudocode für "SetEqSystem.m"

- $A \in \mathbb{R}^{n_{\text{nods}} \times n_{\text{nods}}}$
- $f \in \mathbb{R}^{n_{\text{nods}}}$

// order checken

- if (order == 1) {
 - $\phi_i = \{\phi(x_i, \eta_1), \phi(x_i, \eta_2), \dots, \phi(x_i, \eta_n)\}$
 - $\nabla \phi_i = \{1 - x_i - \eta_1, 1 - x_i - \eta_2, \dots, 1 - x_i - \eta_n\}$
}

praktische Funktion Handle
- if order == 2:
 - $\phi_i = \{\star^1; \star^2; \star^3; \star^4; \star^5\}$
 - $\nabla \phi_i = \text{analog}$
}

See Self exercise or Google

// Integrationsregel

- $[w, p, h] = \text{MeshOps. Get IntegrationsRegel}$

// Element-Loop

- for k = 1 .. nElements
 - $J = \text{MeshOps} \dots ; J^{-T} = \text{MeshOps} \dots$
 - $A^e = \text{zeros}(3, 3) \quad // \text{für 1-Order. Besser zeros}(3 * \text{order}, 3 * \text{order})$
 - $f^e = \text{zeros}(3)$

for i = 1 .. $3 * \text{order}$:

- for j = 1 .. $3 * \text{order}$:
 - $A^e[i, j] += w \cdot \left[\left(J^{-T} \cdot \nabla \hat{N}_j(\underline{p}_e) \right)^T \left(J^{-T} \cdot \nabla N_j(\underline{p}_e) \right) \cdot J \right]$
 - if $j = 1$: $f^e[i] = f(\text{calcMappedIntPoint}(f_e)) \cdot N_i(\underline{p}_e) | J |$

// Einordnen in Globale Matrix: conn = MeshOps.getNodeNumOfTri(k);