Semesterarbeit:

Angle conditions for face and edge elements

Mathias Leander Hack

Supervisor: Prof. Dr. R. Hiptmair (SAM, D-MATH)

10. Januar 2005

Inhaltsverzeichnis

Kapitel 1

Aufgabenstellung

1.1 Problem

Kantenelemente sind $\mathbf{H(curl;\Omega)}$ konforme finite Elemente niedrigster Ordnung (Raum W^1(M) , M eine Triangulierung von $\Omega \subseteq R^3$), Flächenelemente anderderseits setzen Finite Element Unterräume von $\mathbf{H(div;\Omega\ )}$ voraus (Raum W^2(M)). Dazu gibt es kanonische lokale Interpolationsoperatoren I^1 : (C^0 (Ω) $)^3 \mapsto W^1$(M) und I^2 : (C^0 (Ω) $)^3 \mapsto W^2$(M) , welche zwar nicht auf $\mathbf{H(curl;\Omega)}$ respektive auf $\mathbf{H(div;\Omega\ )}$ definiert sind, dafür aber auf den Räumen $\mathbf{H^1(curl;\Omega\ )} := \{\ \mathbf{u} \subseteq\ \mathbf{H^1(\Omega\ )} : \mathbf{curl\ u} \subseteq\ \mathbf{H^1(\Omega\ )}\ \}$, respektive auf $\mathbf{H^1(\Omega\ )}$.

1.2 Ziel

$$\begin{aligned} \| \mathbf{u - I^1 u} \|_{L^2(T)} &\leq C_1 \cdot \left(\| \mathbf{u} \|^2_{H^1(T)} + \| \mathbf{curl\ u} \|^2_{H^1(T)} \right)^{\frac{1}{2}} \\ \| \mathbf{u - I^2 u} \|_{L^2(T)} &\leq C_2 \cdot \| \mathbf{u} \|_{H^1(T)} \end{aligned}$$

Für beliebige Tetraeder T sollen die besten Konstanten C_1 und C_2 für die lokalen Interpolationsungleichungen berechnet werden. Für die Berechnung soll eine polynomiale Spektralmethode verwendet werden.

1.3 Focus

Implementation einer polynomialen Spektral-Methode und Durchführung numerischer Experimente um die Winkelabhängigkeit der Konstanten C_1 , C_2 zu bestimmen.

Kapitel 2

Lösung

2.1 Polynomiale Spektralmethode I

Sei T das Tetraeder (P_1 , P_2 , P_3 und P_4).

Damit werden die baryzentrischen Funktionen berechnet:

$$\begin{aligned}
\lambda_1 &= M_{(1,1)} \cdot x + M_{(1,2)} \cdot y + M_{(1,3)} \cdot z + M_{(1,4)} \\
\lambda_2 &= M_{(2,1)} \cdot x + M_{(2,2)} \cdot y + M_{(2,3)} \cdot z + M_{(2,4)} \\
\lambda_3 &= M_{(3,1)} \cdot x + M_{(3,2)} \cdot y + M_{(3,3)} \cdot z + M_{(3,4)} \\
\lambda_4 &= M_{(4,1)} \cdot x + M_{(4,2)} \cdot y + M_{(4,3)} \cdot z + M_{(4,4)}
\end{aligned} \tag{2.1}$$

Wir betrachten das Problem im Raum $P_m(T)$, dem Raum der Polynome über T vom Grad $\leq$ m.

Die Basis $\mathbb{B}$ für $P_m(T)$ wird wiefolgt gewählt:

$$\mathbb{B} = \left\{ \begin{pmatrix} \lambda_1^{\alpha_1} \cdot \lambda_2^{\alpha_2} \cdot \lambda_3^{\alpha_3} \cdot \lambda_4^{\alpha_4} \cdot 1_\alpha \\ \lambda_1^{\beta_1} \cdot \lambda_2^{\beta_2} \cdot \lambda_3^{\beta_3} \cdot \lambda_4^{\beta_4} \cdot 1_\beta \\ \lambda_1^{\gamma_1} \cdot \lambda_2^{\gamma_2} \cdot \lambda_3^{\gamma_3} \cdot \lambda_4^{\gamma_4} \cdot 1_\gamma \end{pmatrix} \middle| \begin{array}{l} \alpha = (\alpha_1, \alpha_2, \alpha_3, \alpha_4), \beta = (\beta_1, \beta_2, \beta_3, \beta_4), \gamma = (\gamma_1, \gamma_2, \gamma_3, \gamma_4) \\ \mid \alpha \mid, \mid \beta \mid, \mid \gamma \mid \in \{0, m\}, \quad \alpha_i, \beta_i, \gamma_i \in \mathbb{N}, \quad \alpha, \beta, \gamma \in \mathbb{N}^4 \\ \delta = (\alpha, \beta, \gamma), \quad \mid \delta \mid = m, \quad \delta \in \mathbb{N}^{12} \end{array} \right\}$$

mit $\quad 1_\alpha = \begin{cases} 1 & , \ |\alpha| = m \\ 0 & , \ sonst \end{cases}$

N = | $\mathbb{B}$ | = (m+1) · (m+2) · (m+3) / 2 Dimension der Basis $\mathbb{B}$

Betrachten wir als Beispiel den Fall m = 2:

Es folgt N = 30, d.h. wir haben die 30 Basisfunktionen:

$$\begin{pmatrix} \lambda_1^2 \\ 0 \\ 0 \end{pmatrix}, \begin{pmatrix} \lambda_1\lambda_2 \\ 0 \\ 0 \end{pmatrix}, \begin{pmatrix} \lambda_1\lambda_3 \\ 0 \\ 0 \end{pmatrix}, \begin{pmatrix} \lambda_1\lambda_4 \\ 0 \\ 0 \end{pmatrix}, \begin{pmatrix} \lambda_2^2 \\ 0 \\ 0 \end{pmatrix}, \begin{pmatrix} \lambda_2\lambda_3 \\ 0 \\ 0 \end{pmatrix}, \begin{pmatrix} \lambda_2\lambda_4 \\ 0 \\ 0 \end{pmatrix}, \begin{pmatrix} \lambda_3^2 \\ 0 \\ 0 \end{pmatrix},$$
$$\begin{pmatrix} \lambda_3\lambda_4 \\ 0 \\ 0 \end{pmatrix}, \begin{pmatrix} \lambda_4^2 \\ 0 \\ 0 \end{pmatrix}, \begin{pmatrix} 0 \\ \lambda_1^2 \\ 0 \end{pmatrix}, \begin{pmatrix} 0 \\ \lambda_1\lambda_2 \\ 0 \end{pmatrix}, \begin{pmatrix} 0 \\ \lambda_1\lambda_3 \\ 0 \end{pmatrix}, \begin{pmatrix} 0 \\ \lambda_1\lambda_4 \\ 0 \end{pmatrix}, \begin{pmatrix} 0 \\ \lambda_2^2 \\ 0 \end{pmatrix}, \begin{pmatrix} 0 \\ \lambda_2\lambda_3 \\ 0 \end{pmatrix},$$
$$\begin{pmatrix} 0 \\ \lambda_2\lambda_4 \\ 0 \end{pmatrix}, \begin{pmatrix} 0 \\ \lambda_3^2 \\ 0 \end{pmatrix}, \begin{pmatrix} 0 \\ \lambda_3\lambda_4 \\ 0 \end{pmatrix}, \begin{pmatrix} 0 \\ \lambda_4^2 \\ 0 \end{pmatrix}, \begin{pmatrix} 0 \\ 0 \\ \lambda_1^2 \end{pmatrix}, \begin{pmatrix} 0 \\ 0 \\ \lambda_1\lambda_2 \end{pmatrix}, \begin{pmatrix} 0 \\ 0 \\ \lambda_1\lambda_3 \end{pmatrix}, \begin{pmatrix} 0 \\ 0 \\ \lambda_1\lambda_4 \end{pmatrix},$$
$$\begin{pmatrix} 0 \\ 0 \\ \lambda_2^2 \end{pmatrix}, \begin{pmatrix} 0 \\ 0 \\ \lambda_2\lambda_3 \end{pmatrix}, \begin{pmatrix} 0 \\ 0 \\ \lambda_2\lambda_4 \end{pmatrix}, \begin{pmatrix} 0 \\ 0 \\ \lambda_3^2 \end{pmatrix}, \begin{pmatrix} 0 \\ 0 \\ \lambda_3\lambda_4 \end{pmatrix}, \begin{pmatrix} 0 \\ 0 \\ \lambda_4^2 \end{pmatrix}$$

Das ursprüngliche Problem

$$\| \mathbf{u} - \mathbf{I}^1\mathbf{u} \|_{L^2(T)} \quad \leq \quad C_1 \cdot \left(\| \mathbf{u} \|^2_{H^1(T)} + \| \mathbf{curl\ u} \|^2_{H^1(T)} \right)^{\frac{1}{2}} \tag{2.2}$$

$$\| \mathbf{u} - \mathbf{I}^2\mathbf{u} \|_{L^2(T)} \quad \leq \quad C_2 \cdot \| \mathbf{u} \|_{H^1(T)} \tag{2.3}$$

können wir nun algebraisch umschreiben für $\vec{u} \in P_m(T)$ und erhalten :

$$(\vec{u} - \overrightarrow{I^1 u})^T \cdot M_{L^2(T)} \cdot (\vec{u} - \overrightarrow{I^1 u}) \quad \leq \quad {C_1}^2 \cdot \left(\vec{u}^T \cdot M_{H^1(T)} \cdot \vec{u} \ + \ \vec{u}^T \cdot Curl^T \cdot M_{H^1(T)} \cdot Curl \cdot \vec{u} \right) \tag{2.4}$$

$$(\vec{u} - \overrightarrow{I^2 u})^T \cdot M_{L^2(T)} \cdot (\vec{u} - \overrightarrow{I^2 u}) \quad \leq \quad {C_2}^2 \cdot \vec{u}^T \cdot M_{H^1(T)} \cdot \vec{u} \tag{2.5}$$

$$\begin{aligned} mit \qquad & \vec{u}, \overrightarrow{I^1 u}, \overrightarrow{I^2 u} \in \mathbb{R}^N \\ & M_{H^1(T)}, M_{L^2(T)}, Curl \ \in \mathbb{R}^{N \times N} \end{aligned}$$

Datei base.m

```
function Basis = base(m)
% Output Variablen:
% Base : Basis B für den Raum derPolynome vom Grad <= m
%        ( Nx12-Matrix )
% ---------------------------------------------
% Input Variablen:
% m   : Polynomgrad
%---------------------------------------------
```

```
N = (m+1)*(m+2)*(m+3)/2;
Basis = zeros(N,12);
index = 1;
for i=m:-1:0
 for j=m:-1:0
  for k=m:-1:0
   for l=m:-1:0
     if ( (i+j+k+l) == m )
        Basis(index         ,1:4)  = [i,j,k,l];% alpha
        Basis(index + N/3   ,5:8)  = [i,j,k,l];% beta
        Basis(index + 2*N/3 ,9:12) = [i,j,k,l];% gamma
        index = index + 1;
     end
   end
  end
 end
end
```

2.2 Interpolationsoperatoren

Betrachten wir nun die Interpolationsoperatoren. Sie sind folgendermassen definiert:

$$\mathbf{I}^1\mathbf{u} = \sum_{e\ edge} \left(\int_e \mathbf{u} \cdot \mathbf{ds} \right) \cdot \mathbf{b}_e \tag{2.6}$$

$$\mathbf{I}^2\mathbf{u} = \sum_{f\ face} \left(\int_f \mathbf{u} \cdot \mathbf{n} \cdot dS \right) \cdot \mathbf{b}_f \tag{2.7}$$

Die Normfunktionen $\mathbf{b}_e$ und $\mathbf{b}_f$ sind wiefolgt definiert:

Kantennormfunktionen

$$\mathbf{b}_{ij} = \lambda_i \cdot \nabla\lambda_j - \lambda_j \cdot \nabla\lambda_i \tag{2.8}$$

Flächennormfunktionen

$$\mathbf{b}_{ijk} = 2 \cdot \Big(\lambda_i \cdot (\nabla\lambda_j \times \nabla\lambda_k) + \lambda_j \cdot (\nabla\lambda_k \times \nabla\lambda_i) + \lambda_k \cdot (\nabla\lambda_i \times \nabla\lambda_j) \Big) \tag{2.9}$$

Berechnen wir zuerst die Integrale.

2.2.1 Integralberechnung

Sei

$$\mathbf{u} = \sum_{\substack{\delta\in\mathbb{R}^{12}\\ |\delta|=m\\ (\alpha,\beta,\gamma)=\delta}} u_\delta \begin{pmatrix} \lambda_1^{\alpha_1}\cdot\lambda_2^{\alpha_2}\cdot\lambda_3^{\alpha_3}\cdot\lambda_4^{\alpha_4}\cdot 1_\alpha \\ \lambda_1^{\beta_1}\cdot\lambda_2^{\beta_2}\cdot\lambda_3^{\beta_3}\cdot\lambda_4^{\beta_4}\cdot 1_\beta \\ \lambda_1^{\gamma_1}\cdot\lambda_2^{\gamma_2}\cdot\lambda_3^{\gamma_3}\cdot\lambda_4^{\gamma_4}\cdot 1_\gamma \end{pmatrix}$$

Dann folgt :

$$\begin{aligned}
\int_{Kante_{ij}} \mathbf{u}\cdot\mathbf{ds} &= \int_{Kante_{ij}} \sum_{\substack{\delta\in\mathbb{R}^{12}\\ |\delta|=m\\ (\alpha,\beta,\gamma)=\delta}} u_\delta \begin{pmatrix} \lambda_1^{\alpha_1}\cdot\lambda_2^{\alpha_2}\cdot\lambda_3^{\alpha_3}\cdot\lambda_4^{\alpha_4}\cdot 1_\alpha \\ \lambda_1^{\beta_1}\cdot\lambda_2^{\beta_2}\cdot\lambda_3^{\beta_3}\cdot\lambda_4^{\beta_4}\cdot 1_\beta \\ \lambda_1^{\gamma_1}\cdot\lambda_2^{\gamma_2}\cdot\lambda_3^{\gamma_3}\cdot\lambda_4^{\gamma_4}\cdot 1_\gamma \end{pmatrix} \cdot \underbrace{\mathbf{ds}}_{=\ \mathbf{r}\cdot ds\ mit|\mathbf{r}|=1} \\
&= \sum_{\substack{\delta\in\mathbb{R}^{12}\\ |\delta|=m\\ (\alpha,\beta,\gamma)=\delta}} u_\delta \int_{Kante_{ij}} \begin{pmatrix} \lambda_1^{\alpha_1}\cdot\lambda_2^{\alpha_2}\cdot\lambda_3^{\alpha_3}\cdot\lambda_4^{\alpha_4}\cdot 1_\alpha \\ \lambda_1^{\beta_1}\cdot\lambda_2^{\beta_2}\cdot\lambda_3^{\beta_3}\cdot\lambda_4^{\beta_4}\cdot 1_\beta \\ \lambda_1^{\gamma_1}\cdot\lambda_2^{\gamma_2}\cdot\lambda_3^{\gamma_3}\cdot\lambda_4^{\gamma_4}\cdot 1_\gamma \end{pmatrix} \cdot \mathbf{r}\cdot ds \\
&= \sum_{\substack{\delta\in\mathbb{R}^{12}\\ |\delta|=m\\ (\alpha,\beta,\gamma)=\delta}} u_\delta \sum_{\zeta\in\{\alpha,\beta,\gamma\}} r_\zeta\cdot 1_\zeta \cdot \int_{Kante_{ij}} (\lambda_1^{\zeta_1}\cdot\lambda_2^{\zeta_2}\cdot\lambda_3^{\zeta_3}\cdot\lambda_4^{\zeta_4})ds \qquad (2.10)
\end{aligned}$$

$$\begin{aligned}
\int_{Flaeche_{ijk}} \mathbf{u}\cdot\mathbf{n}\cdot dS &= \int_{Flaeche_{ijk}} \sum_{\substack{\delta\in\mathbb{R}^{12}\\ |\delta|=m\\ (\alpha,\beta,\gamma)=\delta}} u_\delta \begin{pmatrix} \lambda_1^{\alpha_1}\cdot\lambda_2^{\alpha_2}\cdot\lambda_3^{\alpha_3}\cdot\lambda_4^{\alpha_4}\cdot 1_\alpha \\ \lambda_1^{\beta_1}\cdot\lambda_2^{\beta_2}\cdot\lambda_3^{\beta_3}\cdot\lambda_4^{\beta_4}\cdot 1_\beta \\ \lambda_1^{\gamma_1}\cdot\lambda_2^{\gamma_2}\cdot\lambda_3^{\gamma_3}\cdot\lambda_4^{\gamma_4}\cdot 1_\gamma \end{pmatrix} \cdot \mathbf{n}\cdot dS \\
&= \sum_{\substack{\delta\in\mathbb{R}^{12}\\ |\delta|=m\\ (\alpha,\beta,\gamma)=\delta}} u_\delta \int_{Flaeche_{ijk}} \begin{pmatrix} \lambda_1^{\alpha_1}\cdot\lambda_2^{\alpha_2}\cdot\lambda_3^{\alpha_3}\cdot\lambda_4^{\alpha_4}\cdot 1_\alpha \\ \lambda_1^{\beta_1}\cdot\lambda_2^{\beta_2}\cdot\lambda_3^{\beta_3}\cdot\lambda_4^{\beta_4}\cdot 1_\beta \\ \lambda_1^{\gamma_1}\cdot\lambda_2^{\gamma_2}\cdot\lambda_3^{\gamma_3}\cdot\lambda_4^{\gamma_4}\cdot 1_\gamma \end{pmatrix} \cdot \mathbf{n}\cdot dS \\
&= \sum_{\substack{\delta\in\mathbb{R}^{12}\\ |\delta|=m\\ (\alpha,\beta,\gamma)=\delta}} u_\delta \sum_{\zeta\in\{\alpha,\beta,\gamma\}} n_\zeta\ 1_\zeta \int_{Flaeche_{ijk}} (\lambda_1^{\zeta_1}\cdot\lambda_2^{\zeta_2}\cdot\lambda_3^{\zeta_3}\cdot\lambda_4^{\zeta_4})dS \qquad (2.11)
\end{aligned}$$

Aus der Formel

$$\int_K \lambda_1^{\theta_1}\cdot\ \ldots\ \cdot\lambda_{d+1}^{\theta_{d+1}}dx \ = \ |K|\cdot d!\cdot\frac{\theta_1!\cdot\ \ldots\ \cdot\theta_{d+1}!}{(\theta_1+\ldots+\theta_{d+1}+d)!} \quad , \qquad \forall\theta\ \in\ \mathbb{N}_0^{d+1} \qquad (2.12)$$

folgen die 2 Integrale:

$$\begin{aligned}
\int_{Kante_{ij}} \lambda_i^{\zeta_i}\cdot\lambda_j^{\zeta_j}\cdot\lambda_k^{\zeta_k}\cdot\lambda_l^{\zeta_l}\cdot ds &= \begin{cases} |Kante_{ij}|\cdot\frac{\zeta_i!\cdot\zeta_j!}{(\zeta_i+\zeta_j+1)!} & ,\ \zeta_k,\zeta_l=0 \\ 0 & ,\ sonst \end{cases} \\
\int_{Flaeche_{ijk}} \lambda_i^{\zeta_i}\cdot\lambda_j^{\zeta_j}\cdot\lambda_k^{\zeta_k}\cdot\lambda_l^{\zeta_l}\cdot dS &= \begin{cases} 2\cdot|Flaeche_{ijk}|\cdot\frac{\zeta_i!\cdot\zeta_j!\cdot\zeta_k!}{(\zeta_i+\zeta_j+\zeta_k+2)!} & ,\ \zeta_l=0 \\ 0 & ,\ sonst \end{cases}
\end{aligned} \qquad (2.13)$$

2.2.2 Berechnung der Normfunktionen

Die Normfunktionen (2.8) und (2.9) sollen in der Basis $\mathbb{B}$ dargestellt werden. Dazu benötigen wir folgende Gleichung:

$$\begin{aligned} \lambda_i &= \lambda_i \cdot (\lambda_1 + \lambda_2 + \lambda_3 + \lambda_4)^{m-1} \\ &= \lambda_i \cdot \sum_{\substack{\zeta \in \mathbb{R}^4 \\ |\zeta| = m-1}} \frac{(m-1)!}{\zeta_1! \cdot \zeta_2! \cdot \zeta_3! \cdot \zeta_4!} \lambda_1^{\zeta_1} \cdot \lambda_2^{\zeta_2} \cdot \lambda_3^{\zeta_3} \cdot \lambda_4^{\zeta_4} \end{aligned} \tag{2.14}$$

Erweitert man (2.14) in 3 Dimensionen, so erhält man:

$$\begin{pmatrix} \lambda_i \\ \lambda_j \\ \lambda_k \end{pmatrix} = \begin{pmatrix} \lambda_i & 0 & 0 \\ 0 & \lambda_j & 0 \\ 0 & 0 & \lambda_k \end{pmatrix} \cdot \sum_{\substack{\delta \in \mathbb{R}^{12} \\ |\delta| = m-1 \\ (\alpha,\beta,\gamma) = \delta}} \frac{(m-1)!}{\alpha! \cdot \beta! \cdot \gamma!} \begin{pmatrix} \lambda_1^{\alpha_1} \cdot \lambda_2^{\alpha_2} \cdot \lambda_3^{\alpha_3} \cdot \lambda_4^{\alpha_4} \cdot 1_\alpha \\ \lambda_1^{\beta_1} \cdot \lambda_2^{\beta_2} \cdot \lambda_3^{\beta_3} \cdot \lambda_4^{\beta_4} \cdot 1_\beta \\ \lambda_1^{\gamma_1} \cdot \lambda_2^{\gamma_2} \cdot \lambda_3^{\gamma_3} \cdot \lambda_4^{\gamma_4} \cdot 1_\gamma \end{pmatrix} \tag{2.15}$$

$$mit \quad \alpha! = \alpha_1!\alpha_2!\alpha_3!\alpha_4!, \quad \beta! = \beta_1!\beta_2!\beta_3!\beta_4!, \quad \gamma! = \gamma_1!\gamma_2!\gamma_3!\gamma_4!$$

Für die Kantennormfunktion (2.8) erhält man:

$$\begin{aligned} \mathbf{b}_{ij} &= \lambda_i \cdot \begin{pmatrix} M_{(j,1)} \\ M_{(j,2)} \\ M_{(j,3)} \end{pmatrix} - \lambda_j \cdot \begin{pmatrix} M_{(i,1)} \\ M_{(i,2)} \\ M_{(i,3)} \end{pmatrix} \\ &= \begin{pmatrix} M_{(j,1)} & 0 & 0 \\ 0 & M_{(j,2)} & 0 \\ 0 & 0 & M_{(j,3)} \end{pmatrix} \cdot \lambda_i \cdot \sum_{\substack{\delta \in \mathbb{R}^{12} \\ |\delta| = m-1 \\ (\alpha,\beta,\gamma) = \delta}} \frac{(m-1)!}{\alpha!\beta!\gamma!} \begin{pmatrix} \lambda_1^{\alpha_1} \cdot \lambda_2^{\alpha_2} \cdot \lambda_3^{\alpha_3} \cdot \lambda_4^{\alpha_4} \cdot 1_\alpha \\ \lambda_1^{\beta_1} \cdot \lambda_2^{\beta_2} \cdot \lambda_3^{\beta_3} \cdot \lambda_4^{\beta_4} \cdot 1_\beta \\ \lambda_1^{\gamma_1} \cdot \lambda_2^{\gamma_2} \cdot \lambda_3^{\gamma_3} \cdot \lambda_4^{\gamma_4} \cdot 1_\gamma \end{pmatrix} \\ &- \begin{pmatrix} M_{(i,1)} & 0 & 0 \\ 0 & M_{(i,2)} & 0 \\ 0 & 0 & M_{(i,3)} \end{pmatrix} \cdot \lambda_j \cdot \sum_{\substack{\delta \in \mathbb{R}^{12} \\ |\delta| = m-1 \\ (\alpha,\beta,\gamma) = \delta}} \frac{(m-1)!}{\alpha!\beta!\gamma!} \begin{pmatrix} \lambda_1^{\alpha_1} \cdot \lambda_2^{\alpha_2} \cdot \lambda_3^{\alpha_3} \cdot \lambda_4^{\alpha_4} \cdot 1_\alpha \\ \lambda_1^{\beta_1} \cdot \lambda_2^{\beta_2} \cdot \lambda_3^{\beta_3} \cdot \lambda_4^{\beta_4} \cdot 1_\beta \\ \lambda_1^{\gamma_1} \cdot \lambda_2^{\gamma_2} \cdot \lambda_3^{\gamma_3} \cdot \lambda_4^{\gamma_4} \cdot 1_\gamma \end{pmatrix} \\ &= \sum_{\substack{\delta \in \mathbb{R}^{12} \\ |\delta| = m-1 \\ (\alpha,\beta,\gamma) = \delta}} \frac{(m-1)!}{\alpha!\beta!\gamma!} \cdot \begin{pmatrix} (\lambda_1^{\alpha_1} \cdot \lambda_2^{\alpha_2} \cdot \lambda_3^{\alpha_3} \cdot \lambda_4^{\alpha_4}) \cdot 1_\alpha \cdot (M_{(j,1)} \cdot \lambda_i - M_{(i,1)} \cdot \lambda_j) \\ (\lambda_1^{\beta_1} \cdot \lambda_2^{\beta_2} \cdot \lambda_3^{\beta_3} \cdot \lambda_4^{\beta_4}) \cdot 1_\beta \cdot (M_{(j,2)} \cdot \lambda_i - M_{(i,2)} \cdot \lambda_j) \\ (\lambda_1^{\gamma_1} \cdot \lambda_2^{\gamma_2} \cdot \lambda_3^{\gamma_3} \cdot \lambda_4^{\gamma_4}) \cdot 1_\gamma \cdot (M_{(j,3)} \cdot \lambda_i - M_{(i,3)} \cdot \lambda_j) \end{pmatrix} \end{aligned} \tag{2.16}$$

Für die Flächennormfunktion (2.9) ergibt sich :

$$
\begin{aligned}
\mathbf{b}_{ijk} &= 2 \cdot \Big(\lambda_i \cdot \underbrace{(\nabla\lambda_j \times \nabla\lambda_k)}_{=\ \mathbf{v}^{jk}} + \quad \lambda_j \cdot \underbrace{(\nabla\lambda_k \times \nabla\lambda_i)}_{=\ \mathbf{v}^{ki}} + \quad \lambda_k \cdot \underbrace{(\nabla\lambda_i \times \nabla\lambda_j)}_{=\ \mathbf{v}^{ij}} \Big) \\
&= 2 \cdot \begin{pmatrix} \lambda_i \cdot v_1^{jk} + \lambda_j \cdot v_1^{ki} + \lambda_k \cdot v_1^{ij} & 0 & 0 \\ 0 & \lambda_i \cdot v_2^{jk} + \lambda_j \cdot v_2^{ki} + \lambda_k \cdot v_2^{ij} & 0 \\ 0 & 0 & \lambda_i \cdot v_3^{jk} + \lambda_j \cdot v_3^{ki} + \lambda_k \cdot v_3^{ij} \end{pmatrix} \\
&\quad \cdot \sum_{\substack{\delta\in\mathbb{R}^{12} \\ |\delta|=m-1 \\ (\alpha,\beta,\gamma)=\delta}} \frac{(m-1)!}{\alpha!\beta!\gamma!} \cdot \begin{pmatrix} (\lambda_1^{\alpha_1} \cdot \lambda_2^{\alpha_2} \cdot \lambda_3^{\alpha_3} \cdot \lambda_4^{\alpha_4} \cdot 1_\alpha) \\ (\lambda_1^{\beta_1} \cdot \lambda_2^{\beta_2} \cdot \lambda_3^{\beta_3} \cdot \lambda_4^{\beta_4} \cdot 1_\beta) \\ (\lambda_1^{\gamma_1} \cdot \lambda_2^{\gamma_2} \cdot \lambda_3^{\gamma_3} \cdot \lambda_4^{\gamma_4} \cdot 1_\gamma) \end{pmatrix} \\
&= \sum_{\substack{\delta\in\mathbb{R}^{12} \\ |\delta|=m-1 \\ (\alpha,\beta,\gamma)=\delta}} \frac{2\cdot(m-1)!}{\alpha!\beta!\gamma!} \cdot \begin{pmatrix} (\lambda_1^{\alpha_1} \cdot \lambda_2^{\alpha_2} \cdot \lambda_3^{\alpha_3} \cdot \lambda_4^{\alpha_4} \cdot 1_\alpha) \cdot (\lambda_i \cdot v_1^{jk} + \lambda_j \cdot v_1^{ki} + \lambda_k \cdot v_1^{ij}) \\ (\lambda_1^{\beta_1} \cdot \lambda_2^{\beta_2} \cdot \lambda_3^{\beta_3} \cdot \lambda_4^{\beta_4} \cdot 1_\beta) \cdot (\lambda_i \cdot v_2^{jk} + \lambda_j \cdot v_2^{ki} + \lambda_k \cdot v_2^{ij}) \\ (\lambda_1^{\gamma_1} \cdot \lambda_2^{\gamma_2} \cdot \lambda_3^{\gamma_3} \cdot \lambda_4^{\gamma_4} \cdot 1_\gamma) \cdot (\lambda_i \cdot v_3^{jk} + \lambda_j \cdot v_3^{ki} + \lambda_k \cdot v_3^{ij}) \end{pmatrix} \qquad (2.17)
\end{aligned}
$$

Wir haben also die Darstellung der Normfunktionen $\mathbf{b}_{ij}$ und $\mathbf{b}_{ijk}$ in der Basis $\mathbb{B}$.
(Notation : $\overrightarrow{b_{ij}}$ und $\overrightarrow{b_{ijk}}$)

Mit (2.10) , (2.11) und (2.13) kann man die Interpolationsoperatoren (2.6) (2.7) neu schreiben.

$$
\overrightarrow{I^1 u} = \sum_{(ij)\ Kante} \overrightarrow{b_{ij}} \cdot \sum_{\substack{\delta\in\mathbb{R}^{12} \\ |\delta|=m \\ (\alpha,\beta,\gamma)=\delta}} u_\delta \sum_{\zeta\in\{\alpha,\beta,\gamma\}} r_{ij}(\zeta) 1_\zeta \cdot \begin{cases} |Kante_{ij}| \cdot \frac{\zeta_i!\cdot\zeta_j!}{(\zeta_i+\zeta_j+1)!} & , \zeta_k, \zeta_l = 0 \\ 0 & , sonst \end{cases} \qquad (2.18)
$$

$$
\overrightarrow{I^2 u} = \sum_{(ijk)\ Flaeche} \overrightarrow{b_{ijk}} \cdot \sum_{\substack{\delta\in\mathbb{R}^{12} \\ |\delta|=m \\ (\alpha,\beta,\gamma)=\delta}} u_\delta \sum_{\zeta\in\{\alpha,\beta,\gamma\}} n_{ijk}(\zeta) 1_\zeta \cdot \begin{cases} 2 \cdot |Flaeche_{ijk}| \cdot \frac{\zeta_i!\cdot\zeta_j!\cdot\zeta_k!}{(\zeta_i+\zeta_j+\zeta_k+1)!} & , \zeta_l = 0 \\ 0 & , sonst \end{cases} \qquad (2.19)
$$

Damit haben wir die Matrizen $\mathbf{I^1}$, $\mathbf{I^2} \in \mathbb{R}^{N\times N}$ bestimmt, die folgendes erfüllen:

$$
\overrightarrow{I^1 u} = \mathbf{I^1} \cdot \vec{u} \qquad (2.20)
$$

$$
\overrightarrow{I^2 u} = \mathbf{I^2} \cdot \vec{u} \qquad (2.21)
$$

Datei get_I1.m

```
function I1 = get_I1(Base,M,P,f)
% Output Variablen:
% I1 : Interpolationsmatrix für Kantenelemente
% ---------------------------------------------
% Input Variablen:
% Base: Basis B ( Nx12-Matrix )
% M   : Baryzentrische Funktionen
%        Lambda(i) = < M(i,:), (x,y,z,1) >
% P   : [P1',P2',P3',P4'] Eckpunkte ( 3x4-Matrix )
% f   : Vektor für Berechnung von Fakultäten
%       Aufruf für j! : fa(j,f)
% ---------------------------------------------
% Subroutines:
% - fa(j,f)                      : berechnet j!
% - get_in_B( Base,[delta],f,nn) : berechnet Darstellung von [delta] in B
%---------------------------------------------
N  = length(Base);
m  = Base(N,12);
I1 = zeros(N,N);
% Kantendefinitionen:
% kk(i,1) : Anfangspunkt der Kante i
% kk(i,2) : Endpunkt der Kante i
kk(1,1) = 1 ; kk(1,2) = 2 ;
kk(2,1) = 1 ; kk(2,2) = 3 ;
kk(3,1) = 1 ; kk(3,2) = 4 ;
kk(4,1) = 2 ; kk(4,2) = 3 ;
kk(5,1) = 2 ; kk(5,2) = 4 ;
kk(6,1) = 3 ; kk(6,2) = 4 ;
% ---------------------------------------------
% Schlaufe über die 6 Kantenelemente:
for i=1:6
    I_kante = zeros(N,N);
    v_int = zeros(1,N);
    v_normfunction = zeros(N,1);
    Kante = norm( (P(:,kk(i,2)) - P(:,kk(i,1))) );
```

```
    r = ( P(:,kk(i,2)) - P(:,kk(i,1)) )/ Kante ;
% - Berechnung des Integrals
    for j=1:N             % Schlaufe über die N Basiselemente:
        if ( Base(j,kk(i,1)) + Base(j,kk(i,2)) == m ) % alpha
            v_int(j) = r(1)*Kante*fa(Base(j,kk(i,1)),f)* ...
                fa(Base(j,kk(i,2)),f)/ fa(m+1,f) ;
        elseif ( Base(j,kk(i,1)+4) + Base(j,kk(i,2)+4) == m ) % beta
            v_int(j) = r(2)*Kante*fa(Base(j,kk(i,1)+4),f)* ...
                fa(Base(j,kk(i,2)+4),f)/ fa(m+1,f) ;
        elseif ( Base(j,kk(i,1)+8) + Base(j,kk(i,2)+8) == m ) % gamma
            v_int(j) = r(3)*Kante*fa(Base(j,kk(i,1)+8),f)* ...
                fa(Base(j,kk(i,2)+8),f)/ fa(m+1,f) ;
        end
    end
% - Berechnung der Darstellung der Normfunktion
%   --- Bezeichnung: m_j1_i ~ M(j,1)*lambda_i
        m_j1_i = zeros(1,12); m_j1_i(kk(i,1)  ) = 1 ;
        m_j2_i = zeros(1,12); m_j2_i(kk(i,1)+4) = 1 ;
        m_j3_i = zeros(1,12); m_j3_i(kk(i,1)+8) = 1 ;
        m_i1_j = zeros(1,12); m_i1_j(kk(i,2)  ) = 1 ;
        m_i2_j = zeros(1,12); m_i2_j(kk(i,2)+4) = 1 ;
        m_i3_j = zeros(1,12); m_i3_j(kk(i,2)+8) = 1 ;
        v_normfunction = M(kk(i,2),1)*get_in_B( Base,m_j1_i,f,0)  ...
                       - M(kk(i,1),1)*get_in_B( Base,m_i1_j,f,0)  ...
                       + M(kk(i,2),2)*get_in_B( Base,m_j2_i,f,0)  ...
                       - M(kk(i,1),2)*get_in_B( Base,m_i2_j,f,0)  ...
                       + M(kk(i,2),3)*get_in_B( Base,m_j3_i,f,0)  ...
                       - M(kk(i,1),3)*get_in_B( Base,m_i3_j,f,0) ;
% - Berechnung der Kantenmatrix
    for k=1:N
        I_kante(k,:) = v_int * v_normfunction(k) ;
    end
    I1 = I1 + I_kante ;
end
```

Datei get_I2.m

```
function I2 = get_I2(Base,M,P,f)
% Output Variablen:
% I2 : Interpolationsmatrix für Flächenelemente
% ---------------------------------------------
% Input Variablen:
% Base: Basis B ( Nx12-Matrix )
% M   : Baryzentrische Funktionen
%        Lambda(i) = < bary_coeff(i,:), (x,y,z,1) >
% P   : [P1',P2',P3',P4'] Eckpunkte ( 3x4-Matrix )
% f   : Vektor für Berechnung von Fakultäten
%       Aufruf für j! : fa(j,f)
% ---------------------------------------------
% Subroutines:
% - fa(j,f)                      : berechnet j!
% - get_in_B( Base,[delta],f,nn) : berechnet Darstellung von delta in B
% - area(p1,p2,p3)               : berechnet Flaeche von Dreieck (p1,p2,p3)
%---------------------------------------------
N  = length(Base);
m  = Base(N,12);
I2 = zeros(N,N);
% ---------------------------------------------
% Flächendefinitionen:
% kk(i,1) : Punkt i
% kk(i,2) : Punkt j
% kk(i,3) : Punkt k
kk(1,1) = 1 ; kk(1,2) = 3 ; kk(1,3) = 2 ;
kk(2,1) = 1 ; kk(2,2) = 2 ; kk(2,3) = 4 ;
kk(3,1) = 1 ; kk(3,2) = 4 ; kk(3,3) = 3 ;
kk(4,1) = 2 ; kk(4,2) = 3 ; kk(4,3) = 4 ;
% ---------------------------------------------
% Schlaufe über die 4 Flächenelemente:
for i=1:4
    I_flaeche = zeros(N,N);
    v_int = zeros(1,N);
    v_normfunction = zeros(N,1);
```

```
    Flaeche = area( (P(:,kk(i,1))) , (P(:,kk(i,2))) , (P(:,kk(i,3)) ));
    n = cross((P(:,kk(i,2))-P(:,kk(i,1))),(P(:,kk(i,3))-P(:,kk(i,1))) )/ ...
        norm(cross((P(:,kk(i,3))-P(:,kk(i,1))),(P(:,kk(i,2))-P(:,kk(i,1))))) ;
% - Berechnung des Integrals
    for j=1:N             % Schlaufe über die N Basiselemente:
        if ( Base(j,kk(i,1)) + Base(j,kk(i,2)) + Base(j,kk(i,3)) == m ) % alpha
            v_int(j) = n(1) * 2 * Flaeche * fa(Base(j,kk(i,1)),f) * ...
                fa(Base(j,kk(i,2)),f) * fa(Base(j,kk(i,3)),f) / fa(m+2,f) ;
        elseif ( Base(j,kk(i,1)+4) + Base(j,kk(i,2)+4)+ Base(j,kk(i,3)+4) == m ) % beta
            v_int(j) = n(2)* 2 * Flaeche * fa(Base(j,kk(i,1)+4),f)* ...
                fa(Base(j,kk(i,2)+4),f) * fa(Base(j,kk(i,3)+4),f) / fa(m+2,f) ;
        elseif ( Base(j,kk(i,1)+8) + Base(j,kk(i,2)+8) + Base(j,kk(i,3)+8) == m ) % gamma
            v_int(j) = n(3) * 2 * Flaeche * fa(Base(j,kk(i,1)+8),f)* ...
                fa(Base(j,kk(i,2)+8),f) * fa(Base(j,kk(i,3)+8),f) / fa(m+2,f) ;
        end
    end
% - Berechnung der Darstellung der Normfunktion
%   --- Vektoren v_ij,v_jk, v_ki
        v_ij = cross(M(kk(i,1),1:3) , M(kk(i,2),1:3) );
        v_ki = cross(M(kk(i,3),1:3) , M(kk(i,1),1:3) );
        v_jk = cross(M(kk(i,2),1:3) , M(kk(i,3),1:3) );
%   --- Bezeichnung: l_i_jk_2 ~ lambda_i * v_jk(2)
        l_i_jk_1 = zeros(1,12); l_i_jk_1(kk(i,1)  ) = 1 ;
        l_i_jk_2 = zeros(1,12); l_i_jk_2(kk(i,1)+4) = 1 ;
        l_i_jk_3 = zeros(1,12); l_i_jk_3(kk(i,1)+8) = 1 ;
        l_j_ki_1 = zeros(1,12); l_j_ki_1(kk(i,2)  ) = 1 ;
        l_j_ki_2 = zeros(1,12); l_j_ki_2(kk(i,2)+4) = 1 ;
        l_j_ki_3 = zeros(1,12); l_j_ki_3(kk(i,2)+8) = 1 ;
        l_k_ij_1 = zeros(1,12); l_k_ij_1(kk(i,3)  ) = 1 ;
        l_k_ij_2 = zeros(1,12); l_k_ij_2(kk(i,3)+4) = 1 ;
        l_k_ij_3 = zeros(1,12); l_k_ij_3(kk(i,3)+8) = 1 ;
%   --- b_ijk in Basis B
        v_normfunction = (   v_jk(1) * get_in_B( Base, l_i_jk_1 ,f,0)  ...
                           + v_ki(1) * get_in_B( Base, l_j_ki_1 ,f,0)  ...
                           + v_ij(1) * get_in_B( Base, l_k_ij_1 ,f,0)  ...
                           + v_jk(2) * get_in_B( Base, l_i_jk_2 ,f,0)  ...
```

```
                          + v_ki(2) * get_in_B( Base, l_j_ki_2 ,f,0)  ...
                          + v_ij(2) * get_in_B( Base, l_k_ij_2 ,f,0)  ...
                          + v_jk(3) * get_in_B( Base, l_i_jk_3 ,f,0)  ...
                          + v_ki(3) * get_in_B( Base, l_j_ki_3 ,f,0)  ...
                          + v_ij(3) * get_in_B( Base, l_k_ij_3 ,f,0)  )*2 ;
% - Berechnung der Kantenmatrix
    for k=1:N
        I_flaeche(k,:) = v_int * v_normfunction(k) ;
    end
    I2 = I2 + I_flaeche ;
end
```

Datei get_in_B.m

```
function v = get_in_B( Base, pot ,fac,nn)
% Output Variablen:
% v   : Darstellung von pot in der Basis B
% ---------------------------------------------
% Input Variablen:
% Base: Basis B ( Nx12-Matrix )
% pot : [pot(1), ... , pot(12)] aus N^12
% fac : Vektor für Berechnung von Fakultäten
%       Aufruf für j! : fa(j,fac)
% nn  : Index wenn pot = [0, ... ,0]
%     : 1<=nn<=4->alpha, 5<=nn<=8->beta, 9<=nn<=12-> gamma,
% ---------------------------------------------
% Subroutines:
% - fa(j,fac) : Berechnet j!
%---------------------------------------------
N = length(Base);
m = Base(N,12);
mpot = dot( pot , [1,1,1,1,1,1,1,1,1,1,1,1] );
v = zeros(N,1);
for jj=1:N
   if ( Base(jj,1)  >= pot(1)  & Base(jj,2)  >= pot(2)  & Base(jj,3)  >= pot(3)  & ...
        Base(jj,4)  >= pot(4)  & Base(jj,5)  >= pot(5)  & Base(jj,6)  >= pot(6)  & ...
```

```
  Base(jj,7)  >= pot(7)  & Base(jj,8)  >= pot(8)  & Base(jj,9)  >= pot(9)  & ...
  Base(jj,10) >= pot(10) & Base(jj,11) >= pot(11) & Base(jj,12) >= pot(12) )
% Normalfall, wenn pot ~= 0
if ( mpot ~= 0 )
      v(jj) = fa(m-mpot,fac)/ ...
      ( fa(Base(jj,1)-pot(1),fac)   * fa(Base(jj,2)-pot(2),fac)   * ...
        fa(Base(jj,3)-pot(3),fac)   * fa(Base(jj,4)-pot(4),fac)   * ...
        fa(Base(jj,5)-pot(5),fac)   * fa(Base(jj,6)-pot(6),fac)   * ...
        fa(Base(jj,7)-pot(7),fac)   * fa(Base(jj,8)-pot(8),fac)   * ...
        fa(Base(jj,9)-pot(9),fac)   * fa(Base(jj,10)-pot(10),fac) * ...
        fa(Base(jj,11)-pot(11),fac) * fa(Base(jj,12)-pot(12),fac)      );
% Falls pot == 0
elseif (   nn >= 1 & nn <= 4 & jj >= 1 & jj <= N/3 )
      v(jj) = fa(m-mpot,fac)/ ...
      ( fa(Base(jj,1)-pot(1),fac)   * fa(Base(jj,2)-pot(2),fac)   * ...
        fa(Base(jj,3)-pot(3),fac)   * fa(Base(jj,4)-pot(4),fac)   * ...
        fa(Base(jj,5)-pot(5),fac)   * fa(Base(jj,6)-pot(6),fac)   * ...
        fa(Base(jj,7)-pot(7),fac)   * fa(Base(jj,8)-pot(8),fac)   * ...
        fa(Base(jj,9)-pot(9),fac)   * fa(Base(jj,10)-pot(10),fac) * ...
        fa(Base(jj,11)-pot(11),fac) * fa(Base(jj,12)-pot(12),fac)      );

elseif (   nn >= 5 & nn <= 8 & jj >= N/3+1 & jj <= 2*N/3 )
      v(jj) = fa(m-mpot,fac)/ ...
      ( fa(Base(jj,1)-pot(1),fac)   * fa(Base(jj,2)-pot(2),fac)   * ...
        fa(Base(jj,3)-pot(3),fac)   * fa(Base(jj,4)-pot(4),fac)   * ...
        fa(Base(jj,5)-pot(5),fac)   * fa(Base(jj,6)-pot(6),fac)   * ...
        fa(Base(jj,7)-pot(7),fac)   * fa(Base(jj,8)-pot(8),fac)   * ...
        fa(Base(jj,9)-pot(9),fac)   * fa(Base(jj,10)-pot(10),fac) * ...
        fa(Base(jj,11)-pot(11),fac) * fa(Base(jj,12)-pot(12),fac)      );
elseif (   nn >= 9 & nn <= 12 & jj >= 2*N/3+1 & jj <= N )
      v(jj) = fa(m-mpot,fac)/ ...
      ( fa(Base(jj,1)-pot(1),fac)   * fa(Base(jj,2)-pot(2),fac)   * ...
        fa(Base(jj,3)-pot(3),fac)   * fa(Base(jj,4)-pot(4),fac)   * ...
        fa(Base(jj,5)-pot(5),fac)   * fa(Base(jj,6)-pot(6),fac)   * ...
        fa(Base(jj,7)-pot(7),fac)   * fa(Base(jj,8)-pot(8),fac)   * ...
        fa(Base(jj,9)-pot(9),fac)   * fa(Base(jj,10)-pot(10),fac) * ...
```

```
            fa(Base(jj,11)-pot(11),fac) * fa(Base(jj,12)-pot(12),fac)      );
      end
   end
end
```

2.3 Berechnung der Norm-Matrizen

Seien $\delta = (\alpha, \beta, \gamma),\ \theta = (\epsilon, \zeta, \eta) \in \mathbb{R}^{12}$.

$M_{L^2(T)} = (m_{\delta\theta})$

$$\begin{aligned} m_{\delta\theta} &= \int_T < \begin{pmatrix} \lambda_1^{\alpha_1} \cdot \lambda_2^{\alpha_2} \cdot \lambda_3^{\alpha_3} \cdot \lambda_4^{\alpha_4} \cdot 1_\alpha \\ \lambda_1^{\beta_1} \cdot \lambda_2^{\beta_2} \cdot \lambda_3^{\beta_3} \cdot \lambda_4^{\beta_4} \cdot 1_\beta \\ \lambda_1^{\gamma_1} \cdot \lambda_2^{\gamma_2} \cdot \lambda_3^{\gamma_3} \cdot \lambda_4^{\gamma_4} \cdot 1_\gamma \end{pmatrix}, \begin{pmatrix} \lambda_1^{\epsilon_1} \cdot \lambda_2^{\epsilon_2} \cdot \lambda_3^{\epsilon_3} \cdot \lambda_4^{\epsilon_4} \cdot 1_\epsilon \\ \lambda_1^{\zeta_1} \cdot \lambda_2^{\zeta_2} \cdot \lambda_3^{\zeta_3} \cdot \lambda_4^{\zeta_4} \cdot 1_\zeta \\ \lambda_1^{\eta_1} \cdot \lambda_2^{\eta_2} \cdot \lambda_3^{\eta_3} \cdot \lambda_4^{\eta_4} \cdot 1_\eta \end{pmatrix} > dV \\ &= \sum_{(\nu,\mu)\ \in\{(\alpha,\epsilon),(\beta,\zeta),(\gamma,\eta)\}} 1_\nu \cdot 1_\mu \cdot \int_T \lambda_1^{(\nu_1+\mu_1)} \cdot \lambda_2^{(\nu_2+\mu_2)} \cdot \lambda_3^{(\nu_3+\mu_3)} \cdot \lambda_4^{(\nu_4+\mu_4)} \cdot dV \end{aligned} \tag{2.22}$$

Mit der aus (2.12) abgeleiteten Formel

$$\int_T \lambda_1^{\rho_1} \cdot \lambda_2^{\rho_2} \cdot \lambda_3^{\rho_3} \cdot \lambda_4^{\rho_4} \cdot dV = 3! \cdot |T| \cdot \frac{\rho_1! \cdot \rho_2! \cdot \rho_3! \cdot \rho_4!}{(\rho_1 + \rho_2 + \rho_3 + \rho_4 + 3)!} \tag{2.23}$$

folgt für $M_{L^2(T)}$:

$$m_{\delta\theta} = 3! \cdot |T| \cdot \sum_{(\nu,\mu)\ \in\{(\alpha,\epsilon),(\beta,\zeta),(\gamma,\eta)\}} 1_\nu \cdot 1_\mu \cdot \frac{(\nu_1+\mu_1)! \cdot (\nu_2+\mu_2)! \cdot (\nu_3+\mu_3)! \cdot (\nu_4+\mu_4)!}{(\nu_1+\nu_2+\nu_3+\nu_4+\mu_1+\mu_2+\mu_3+\mu_4+3)!} \tag{2.24}$$

Datei get_L2.m

```
function L2 = get_L2(Base,T,f)
% Output Variablen:
% L2 : L^2-Norm-Matrix
% ---------------------------------------------
% Input Variablen:
% Base: Basis B ( Nx12-Matrix )
% T   : Tetraeder-Volumen
% f   : Vektor für Berechnung von Fakultäten
%       Aufruf für j! : fa(j,f)
% ---------------------------------------------
% Subroutines:
% - fa(j,f) : berechnet j!
%---------------------------------------------
N = length(Base);
m = Base(N,12);
L2 = zeros(N,N);
for ii=1:N
  for jj=1:N
    if ( ( dot(Base(ii,1:4),[1,1,1,1]) == m ) & (dot(Base(jj,1:4),[1,1,1,1]) == m ) )
     L2(ii,jj) = (  fa( Base(ii,1)+Base(jj,1) ,f)*fa(Base(ii,2)+Base(jj,2),f)* ...
                    fa(Base(ii,3)+Base(jj,3),f)*fa(Base(ii,4)+Base(jj,4),f)) * ...
                    6 * T / fa(2*m+3,f) ;
    elseif ( ( dot(Base(ii,5:8),[1,1,1,1]) == m ) & (dot(Base(jj,5:8),[1,1,1,1]) == m ) )
     L2(ii,jj) = (  fa( Base(ii,5)+Base(jj,5) ,f)*fa(Base(ii,6)+Base(jj,6),f)* ...
                    fa(Base(ii,7)+Base(jj,7),f)*fa(Base(ii,8)+Base(jj,8),f)) * ...
                    6 * T / fa(2*m+3,f) ;
    elseif ( ( dot(Base(ii,9:12),[1,1,1,1]) == m ) & (dot(Base(jj,9:12),[1,1,1,1]) == m ) )
     L2(ii,jj) = (  fa( Base(ii,9)+Base(jj,9) ,f)*fa(Base(ii,10)+Base(jj,10),f)* ...
                    fa(Base(ii,11)+Base(jj,11),f)*fa(Base(ii,12)+Base(jj,12),f)) * ...
                    6 * T / fa(2*m+3,f) ;
    end
  end
end
```

Betrachten wir nun die partiellen Ableitungen:

$$\begin{aligned}
\frac{\partial}{\partial x_i}\begin{pmatrix} \lambda_1^{\alpha_1}\cdot\lambda_2^{\alpha_2}\cdot\lambda_3^{\alpha_3}\cdot\lambda_4^{\alpha_4}\cdot 1_\alpha \\ \lambda_1^{\beta_1}\cdot\lambda_2^{\beta_2}\cdot\lambda_3^{\beta_3}\cdot\lambda_4^{\beta_4}\cdot 1_\beta \\ \lambda_1^{\gamma_1}\cdot\lambda_2^{\gamma_2}\cdot\lambda_3^{\gamma_3}\cdot\lambda_4^{\gamma_4}\cdot 1_\gamma \end{pmatrix} &= \begin{pmatrix} \alpha_1\cdot M_{(1,i)}\cdot\lambda_1^{\alpha_1-1}\cdot\lambda_2^{\alpha_2}\cdot\lambda_3^{\alpha_3}\cdot\lambda_4^{\alpha_4}\cdot 1_\alpha \\ \beta_1\cdot M_{(1,i)}\cdot\lambda_1^{\beta_1-1}\cdot\lambda_2^{\beta_2}\cdot\lambda_3^{\beta_3}\cdot\lambda_4^{\beta_4}\cdot 1_\beta \\ \gamma_1\cdot M_{(1,i)}\cdot\lambda_1^{\gamma_1-1}\cdot\lambda_2^{\gamma_2}\cdot\lambda_3^{\gamma_3}\cdot\lambda_4^{\gamma_4}\cdot 1_\gamma \end{pmatrix} \\
&+ \begin{pmatrix} \alpha_2\cdot M_{(2,i)}\cdot\lambda_1^{\alpha_1}\cdot\lambda_2^{\alpha_2-1}\cdot\lambda_3^{\alpha_3}\cdot\lambda_4^{\alpha_4}\cdot 1_\alpha \\ \beta_2\cdot M_{(2,i)}\cdot\lambda_1^{\beta_1}\cdot\lambda_2^{\beta_2-1}\cdot\lambda_3^{\beta_3}\cdot\lambda_4^{\beta_4}\cdot 1_\beta \\ \gamma_2\cdot M_{(2,i)}\cdot\lambda_1^{\gamma_1}\cdot\lambda_2^{\gamma_2-1}\cdot\lambda_3^{\gamma_3}\cdot\lambda_4^{\gamma_4}\cdot 1_\gamma \end{pmatrix} \\
&+ \begin{pmatrix} \alpha_3\cdot M_{(3,i)}\cdot\lambda_1^{\alpha_1}\cdot\lambda_2^{\alpha_2}\cdot\lambda_3^{\alpha_3-1}\cdot\lambda_4^{\alpha_4}\cdot 1_\alpha \\ \beta_3\cdot M_{(3,i)}\cdot\lambda_1^{\beta_1}\cdot\lambda_2^{\beta_2}\cdot\lambda_3^{\beta_3-1}\cdot\lambda_4^{\beta_4}\cdot 1_\beta \\ \gamma_3\cdot M_{(3,i)}\cdot\lambda_1^{\gamma_1}\cdot\lambda_2^{\gamma_2}\cdot\lambda_3^{\gamma_3-1}\cdot\lambda_4^{\gamma_4}\cdot 1_\gamma \end{pmatrix} \\
&+ \begin{pmatrix} \alpha_4\cdot M_{(4,i)}\cdot\lambda_1^{\alpha_1}\cdot\lambda_2^{\alpha_2}\cdot\lambda_3^{\alpha_3}\cdot\lambda_4^{\alpha_4-1}\cdot 1_\alpha \\ \beta_4\cdot M_{(4,i)}\cdot\lambda_1^{\beta_1}\cdot\lambda_2^{\beta_2}\cdot\lambda_3^{\beta_3}\cdot\lambda_4^{\beta_4-1}\cdot 1_\beta \\ \gamma_4\cdot M_{(4,i)}\cdot\lambda_1^{\gamma_1}\cdot\lambda_2^{\gamma_2}\cdot\lambda_3^{\gamma_3}\cdot\lambda_4^{\gamma_4-1}\cdot 1_\gamma \end{pmatrix}
\end{aligned} \tag{2.25}$$

Die Darstellung von ($\nu_j\cdot M_{(j,i)}\cdot\lambda_1^{\nu_1-1}\cdot\lambda_2^{\nu_2}\cdot\lambda_3^{\nu_3}\cdot\lambda_4^{\nu_4}\cdot 1_\nu$) in der Basis $\mathbb{B}$ sieht wiefolgt aus:

$$(\nu_1\cdot M_{(1,i)}\cdot\lambda_1^{\nu_1-1}\lambda_2^{\nu_2}\lambda_3^{\nu_3}\lambda_4^{\nu_4}\cdot 1_\nu)\sum_{\substack{\mu\in\mathbb{N}_0^4\\ |\mu|\,=\,1}}\lambda_1^{\mu_1}\lambda_2^{\mu_2}\lambda_3^{\mu_3}\lambda_4^{\mu_4} \tag{2.26}$$

Datei d_i_Bj.m

```
function v = d_i_Bj(Base,M,i,j,f)
% Output Variablen:
% v   : i-te Ableitung der j-ten Basis
% ---------------------------------------------
% Input Variablen:
% Base: Basis B ( Nx12-Matrix )
% M   : Baryzentrische Funktionen
% i   : i=1 -> x , i=2 -> y , i=3 -> z
% j   : j-te Basisfunktion
% f   : Vektor für Berechnung von Fakultäten
%       Aufruf für j! : fa(j,f)
% ---------------------------------------------
% Subroutines:
% - get_in_B( Base,[delta],f,nn) : berechnet Darstellung von delta in B
%---------------------------------------------
```

```
v = zeros(length(Base),1);
koeff = [Base(j,1)*M(1,i) , Base(j,2)*M(2,i) , Base(j,3)*M(3,i) , Base(j,4)*M(4,i), ...
         Base(j,5)*M(1,i) , Base(j,6)*M(2,i) , Base(j,7)*M(3,i) , Base(j,8)*M(4,i), ...
         Base(j,9)*M(1,i) , Base(j,10)*M(2,i), Base(j,11)*M(3,i), Base(j,12)*M(4,i)];
pot = [Base(j,:);Base(j,:);Base(j,:);Base(j,:);Base(j,:);Base(j,:); ...
       Base(j,:);Base(j,:);Base(j,:);Base(j,:);Base(j,:);Base(j,:)] ;
for kk=1:12
    if ( (koeff(kk) ~= 0) & (pot(kk,kk) > 0) )
        pot(kk,kk) =  pot(kk,kk) - 1;
        v = v + koeff(kk) * get_in_B(Base,pot(kk,:),f,kk);
    end
end
```

Damit können wir die Ableitungsmatrizen $\mathbf{D_x}$, $\mathbf{D_y}$ und $\mathbf{D_z}$ bestimmen und damit $M_{H^1(T)}$ berechnen:

$$M_{H^1(T)} \;=\; M_{L^2(T)} \;+\; \mathbf{D_x}^T \cdot M_{L^2(T)} \cdot \mathbf{D_x} \;+\; \mathbf{D_y}^T \cdot M_{L^2(T)} \cdot \mathbf{D_y} \;+\; \mathbf{D_z}^T \cdot M_{L^2(T)} \cdot \mathbf{D_z} \tag{2.27}$$

Nun müssen wir noch **Curl** bestimmen:

$$\mathbf{curl\ u} \;=\; \sum_{\substack{\delta\in\mathbb{R}^{12}\\ |\delta|=m\\ (\alpha,\beta,\gamma)=\delta}} u_\delta \cdot \begin{pmatrix} \frac{\partial}{\partial y}(\lambda_1^{\gamma_1}\cdot\lambda_2^{\gamma_2}\cdot\lambda_3^{\gamma_3}\cdot\lambda_4^{\gamma_4}\cdot 1_\gamma) & - & \frac{\partial}{\partial z}(\lambda_1^{\beta_1}\cdot\lambda_2^{\beta_2}\cdot\lambda_3^{\beta_3}\cdot\lambda_4^{\beta_4}\cdot 1_\beta) \\ \frac{\partial}{\partial z}(\lambda_1^{\alpha_1}\cdot\lambda_2^{\alpha_2}\cdot\lambda_3^{\alpha_3}\cdot\lambda_4^{\alpha_4}\cdot 1_\alpha) & - & \frac{\partial}{\partial x}(\lambda_1^{\gamma_1}\cdot\lambda_2^{\gamma_2}\cdot\lambda_3^{\gamma_3}\cdot\lambda_4^{\gamma_4}\cdot 1_\gamma) \\ \frac{\partial}{\partial x}(\lambda_1^{\beta_1}\cdot\lambda_2^{\beta_2}\cdot\lambda_3^{\beta_3}\cdot\lambda_4^{\beta_4}\cdot 1_\beta) & - & \frac{\partial}{\partial y}(\lambda_1^{\alpha_1}\cdot\lambda_2^{\alpha_2}\cdot\lambda_3^{\alpha_3}\cdot\lambda_4^{\alpha_4}\cdot 1_\alpha) \end{pmatrix} \tag{2.28}$$

Und $\frac{\partial}{\partial y}(\lambda_1^{\gamma_1}\cdot\lambda_2^{\gamma_2}\cdot\lambda_3^{\gamma_3}\cdot\lambda_4^{\gamma_4}\cdot 1_\gamma)$ können wir mit (2.26) darstellen.

Datei get_Curl.m

```
function Curl = get_Curl(Dx,Dy,Dz)
% Output Variablen:
% Curl : Rotationsmatrix für u
% ----------------------------------------------
% Input Variablen:
% Dx : Ableitung nach x
% Dy : Ableitung nach y
% Dz : Ableitung nach z
% ----------------------------------------------
```

```
N = length(Dx);
N3 = N / 3 ;
Curl(:,1:N3       ) = - Dy(:,2*N3+1:N ) + Dz(:,N3+1:2*N3);
Curl(:,N3+1 : 2*N3) = - Dz(:,1:N3     ) + Dx(:,2*N3+1:N );
Curl(:,2*N3+1:N   ) = - Dx(:,N3+1:2*N3) + Dy(:,1:N3     );
```

2.4 Polynomiale Spektralmethode II

Unser Problem $\big($ (2.4) und (2.5) $\big)$ kann nun weiter umgeformt werden.

$$
\begin{aligned}
(\vec{u} - I^1 \cdot \vec{u})^T \cdot M_{L^2(T)} \cdot (\vec{u} - I^1 \cdot \vec{u}) &\leq {C_1}^2 \cdot \Big(\vec{u}^T \cdot M_{H^1(T)} \cdot \vec{u} \; + \; \vec{u}^T \cdot Curl^T \cdot M_{H^1(T)} \cdot Curl \cdot \vec{u}\Big) \\
(\vec{u} - I^2 \cdot \vec{u})^T \cdot M_{L^2(T)} \cdot (\vec{u} - I^2 \cdot \vec{u}) &\leq {C_2}^2 \cdot \vec{u}^T \cdot M_{H^1(T)} \cdot \vec{u}
\end{aligned}
$$

$$\Longleftrightarrow$$

$$
\begin{aligned}
\vec{u}^T \cdot \overbrace{(Id - I^1)^T \cdot M_{L^2(T)} \cdot (Id - I^1)}^{=:A_1} \cdot \vec{u} &\leq {C_1}^2 \cdot \vec{u}^T \cdot \overbrace{(M_{H^1(T)} + Curl^T \cdot M_{H^1(T)} \cdot Curl)}^{=:B_1} \cdot \vec{u} \\
\vec{u}^T \cdot \underbrace{(Id - I^2)^T \cdot M_{L^2(T)} \cdot (Id - I^2)}_{=:A_2} \cdot \vec{u} &\leq {C_2}^2 \cdot \vec{u}^T \cdot M_{H^1(T)} \cdot \vec{u}
\end{aligned}
$$

Da $M_{H^1(T)}$ und B_1 positiv definit sind, existiert eine Cholesky-Zerlegung:

$$
\begin{aligned}
B_1 &= R_1^T \cdot R_1 \\
M_{H^1(T)} &= R_2^T \cdot R_2
\end{aligned}
$$

Damit können wir das Problem umschreiben zu

$$
\begin{aligned}
\vec{u}^T \cdot (R_1^T)^{-1} \cdot A_1 \cdot (R_1)^{-1} \cdot \vec{u} &\leq {C_1}^2 \cdot \vec{u}^T \cdot \vec{u} \\
\vec{u}^T \cdot (R_2^T)^{-1} \cdot A_2 \cdot (R_2)^{-1} \cdot \vec{u} &\leq {C_2}^2 \cdot \vec{u}^T \cdot \vec{u}
\end{aligned}
$$

$$
\Longrightarrow \quad
\begin{aligned}
{C_1}^2 &\equiv \textit{maximaler Eigenwert von } (R_1^T)^{-1} \cdot A_1 \cdot (R_1)^{-1} \\
{C_2}^2 &\equiv \textit{maximaler Eigenwert von } (R_2^T)^{-1} \cdot A_2 \cdot (R_2)^{-1}
\end{aligned}
$$

Kapitel 3

Testprotokoll

Das Programm wird mit dem Blachbox-Verfahren getestet.

3.1 Test-Szenario 1

Die Punkte P_1=(0,0,0) , P_2=(1,0,0), P_3=(0,1,0) und P_4=(0,0,1) definieren das Tetraeder. Getestet wird für Polynomgrad m = 1, . . . , 14 .

Datei Szenario01.m

```
% Szenario 1
% Polynomgrad : m = 1, ... , 14
% Tetraeder T : P1 = (0,0,0), P2 = (1,0,0), P3 = (0,1,0), P4 = (0,0,1)
P1 = [0,0,0];P2 = [1,0,0];
P3 = [0,1,0];P4 = [0,0,1];
file = 'Szenario1.prot';fid = fopen(file,'w');
for m=1:14
    [C1_1(m),C1_2(m),C2(m)] = get_C1C2(P1,P2,P3,P4,m);
    fprintf(fid,'%f,%f,%f\n',C1_1(m),C1_2(m),C2(m));
end
fclose(fid);
```

In der folgenden Graphik sieht man das Konvergenzverhalten beim Einheitstetraeder:

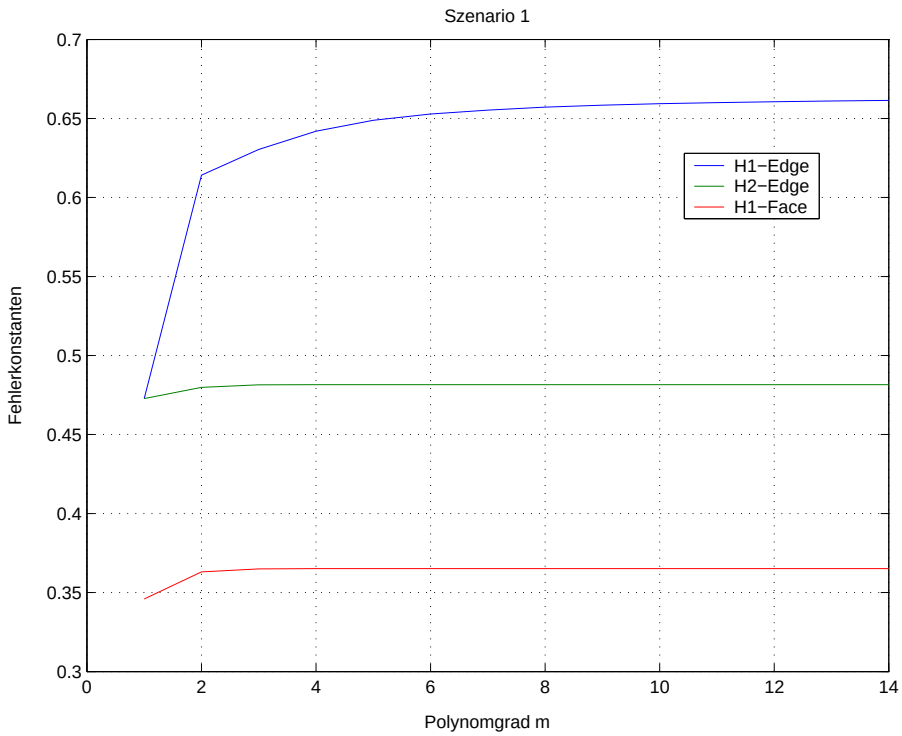

Abbildung 3.1: Datei szen1.eps

3.2 Test-Szenario 2

Die Punkte P_1=(0,0,0) , P_2=(h,0,0), P_3=(0,h,0) und P_4=(0,0,h) definieren das Tetraeder. Getestet wird für Polynomgrad m = 5, und h = 1, . . . , 20 .

Datei Szenario02.m

```
% Szenario 2
% Polynomgrad : m = 5
% Tetraeder T : P1 = (0,0,0), P2 = (h,0,0), P3 = (0,h,0), P4 = (0,0,h)
%               mit h = 1, ... , 20
P1 = [0,0,0]; m  = 5;
file = 'Szenario2.prot';fid = fopen(file,'w');
for h=1:20
    P2 = [h,0,0];P3 = [0,h,0];P4 = [0,0,h];
    [C1_1(h),C1_2(h),C2(h)] = get_C1C2(P1,P2,P3,P4,m);
    fprintf(fid,'%f,%f,%f\n',C1_1(h),C1_2(h),C2(h));
end
fclose(fid);
```

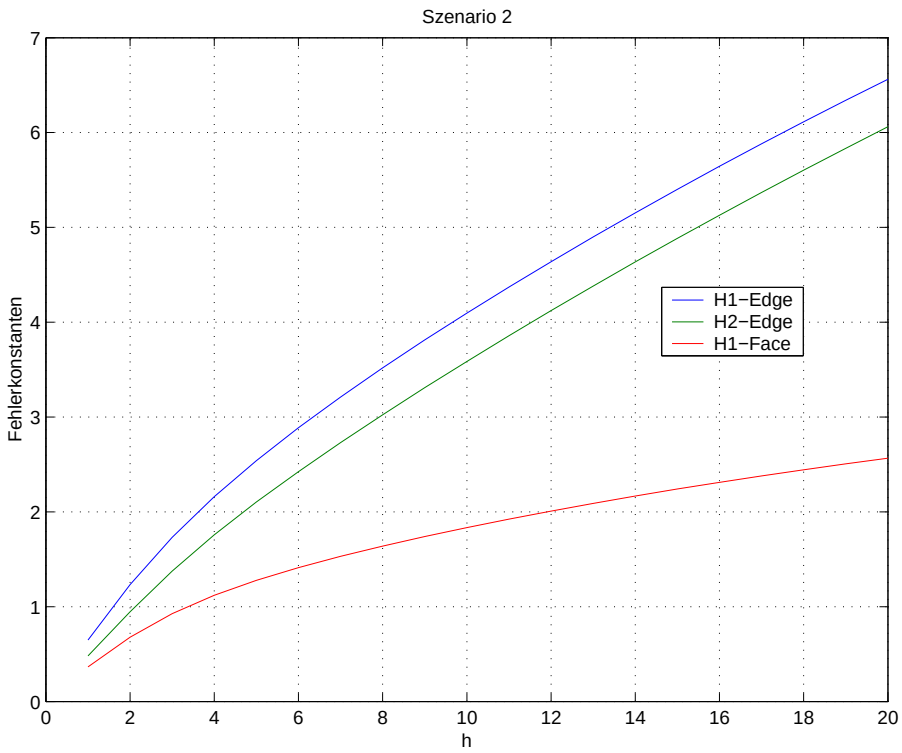

Abbildung 3.2: Datei szen2.eps

3.3 Test-Szenario 3

Die Punkte P_1=(0,0,0) , P_2=(1,0,0), P_3=(0.5,sqrt(3)/2,0) und P_4=(0.5,sqrt(3)/6,h) definieren das Tetraeder.

Getestet wird für Polynomgrad m = 1, ... 5

und für h = 0.01, 0.05, 0.1, 0.5, 1, 5, 10, 50, 100, 500, 1000 .

Datei Szenario03.m

```
% Szenario 3
% Polynomgrad : m = 1, ... ,5
% Tetraeder T : P1 = (0,0,0), P2 = (1,0,0), P3 = (0.5,sqrt(3)/2,0),
%               P4 = (0.5,sqrt(3)/6,h)
%               mit h = [0.01,0.05,0.1,0.5,1,5,10,50,100,500,1000]
file1 = 'Szenario3_C1_1.prot';fid1 = fopen(file1,'w');
file2 = 'Szenario3_C1_2.prot';fid2 = fopen(file2,'w');
file3 = 'Szenario3_C2.prot';  fid3 = fopen(file3,'w');
P1 = [0,0,0];P2 = [1,0,0];P3 = [0.5,(sqrt(3)/2),0];
h = [0.01,0.05,0.1,0.5,1,5,10,50,100,500,1000];
```

```
for ii=1:length(h)
 for m=1:5
   P4 = [0.5,(sqrt(3)/6),h(ii)];
   [C1_1(ii,m),C1_2(ii,m),C2(ii,m)]= get_C1C2(P1,P2,P3,P4,m);
   fprintf(fid1,'%f,%f,%f,%f,%f\n',C1_1(ii,1),C1_1(ii,2),C1_1(ii,3),C1_1(ii,4),C1_1(ii,5));
   fprintf(fid2,'%f,%f,%f,%f,%f\n',C1_2(ii,1),C1_2(ii,2),C1_2(ii,3),C1_2(ii,4),C1_2(ii,5));
   fprintf(fid3,'%f,%f,%f,%f,%f\n',C2(ii,1),C2(ii,2),C2(ii,3),C2(ii,4),C2(ii,5));
 end
end
fclose(fid1);fclose(fid2);fclose(fid3);
```

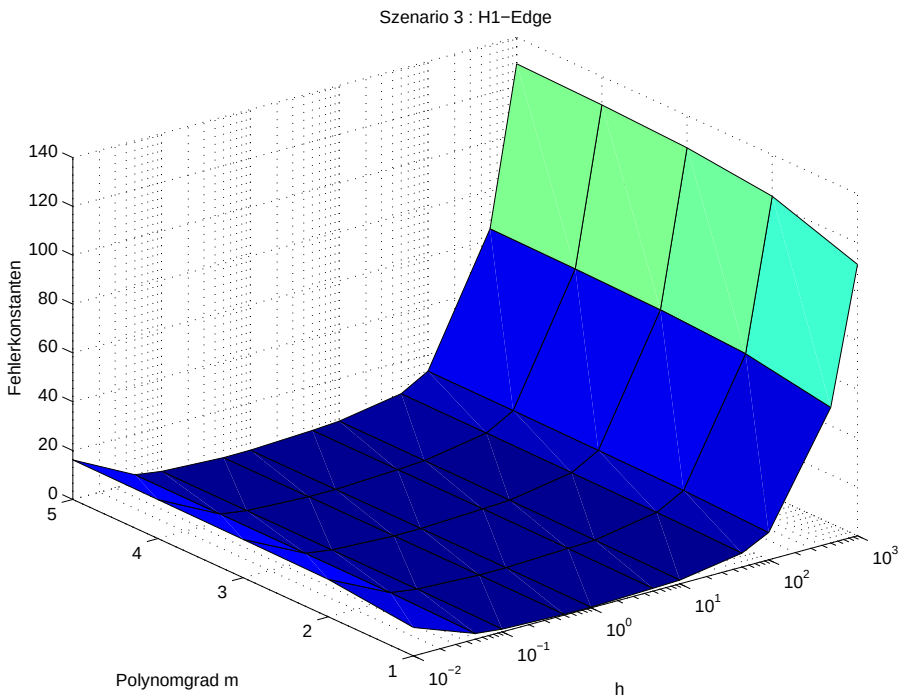

Abbildung 3.3: Datei szen31.eps

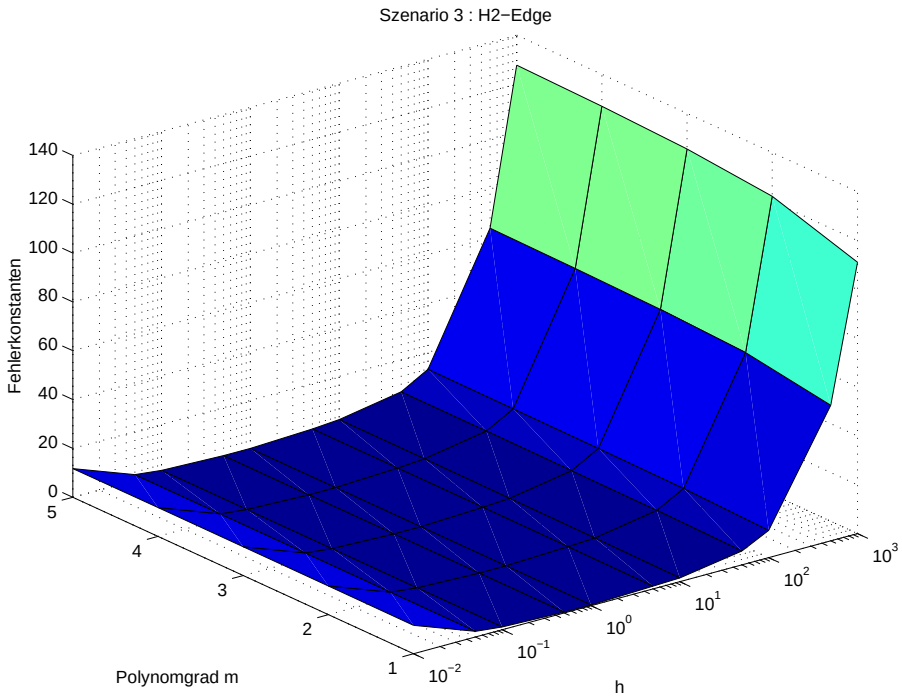

Abbildung 3.4: Datei szen32.eps

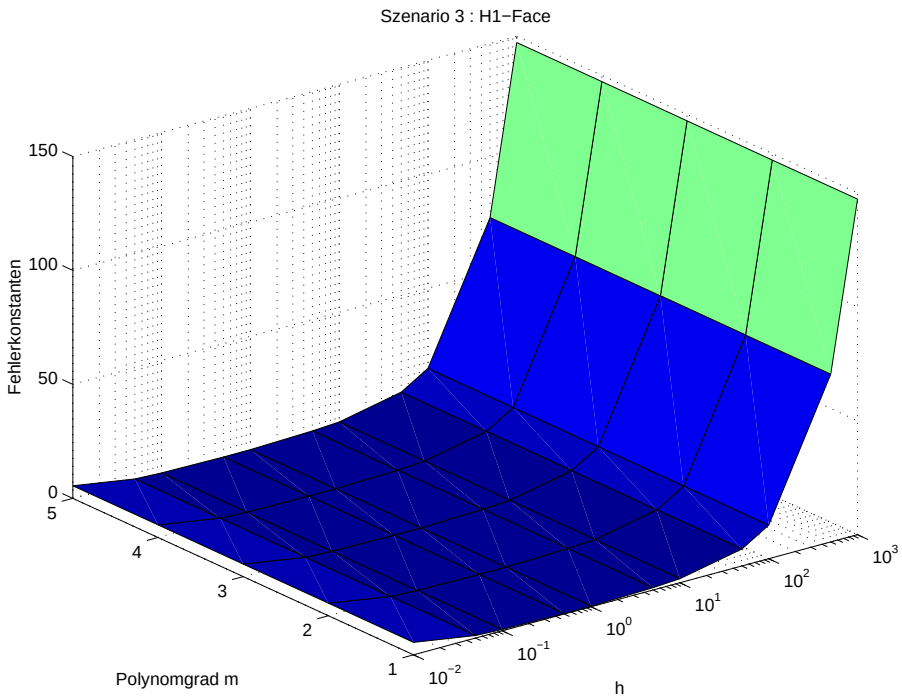

Abbildung 3.5: Datei szen33.eps

3.4 Test-Szenario 4

In diesem Szenario soll die Abhängigkeit zwischen Fehlerkonstanten und Stärke der Stauchung des Tetraeders untersucht werden. Dazu wurde folgendes Programm geschrieben:

Datei Szenario04.m

```
 Szenario 4
% Polynomgrad : m = 1, ... ,5
% Tetraeder T : P1 = (0,0,0), P2 = (1,0,0), P3 = (0,1,0),
%               P4 = (1/3,1/3,h)
%               mit h = [0.01,0.025,0.05,0.075,0.1,0.25,0.5,0.75,1,5,10]
P1 = [0,0,0];P2 = [1,0,0];P3 = [0,1,0];
h = [0.01,0.025,0.05,0.075,0.1,0.25,0.5,0.75,1,5,10];
file1 = 'Szenario4_C1_1.prot';fid1 = fopen(file1,'w');
file2 = 'Szenario4_C1_2.prot';fid2 = fopen(file2,'w');
file3 = 'Szenario4_C2.prot';  fid3 = fopen(file3,'w');
for ii=1:length(h)
    for m=1:5
        P4 = [1/3,1/3,h(ii)];
        [C1_1(ii,m),C1_2(ii,m),C2(ii,m)]= get_C1C2(P1,P2,P3,P4,m);
    end
    fprintf(fid1,'%f,%f,%f,%f,%f\n',C1_1(ii,1),C1_1(ii,2),C1_1(ii,3),C1_1(ii,4),C1_1(ii,5));
    fprintf(fid2,'%f,%f,%f,%f,%f\n',C1_2(ii,1),C1_2(ii,2),C1_2(ii,3),C1_2(ii,4),C1_2(ii,5));
    fprintf(fid3,'%f,%f,%f,%f,%f\n',C2(ii,1),C2(ii,2),C2(ii,3),C2(ii,4),C2(ii,5));
end
fclose(fid1);fclose(fid2);fclose(fid3);
```

Es wurden 2 Varianten dieses Szenarios gerechnet, welche sich nur in der Position des Punktes P4 unterscheiden:

a) P4 = [1/3,1/3,h(ii)] → Bilder : szen41.eps bis szen43.eps

b) P4 = [1/10,1/10,h(ii)] → Bilder : szen51.eps bis szen53.eps

Die Resultate sind aus den folgenden Grafiken ersichtlich. Dabei geht deutlich hervor, dass gestauchte Tetraeder gefährlich sind, da je stärker die Stauchung ist, desto schlechter die Funktion interpoliert wird.

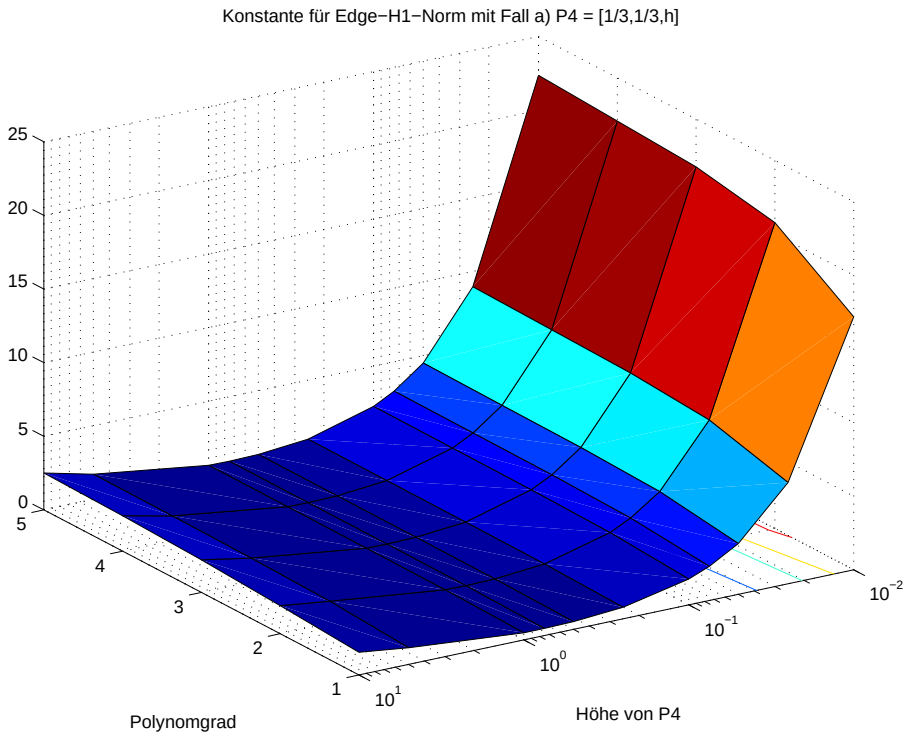

Abbildung 3.6: Datei szen41.eps

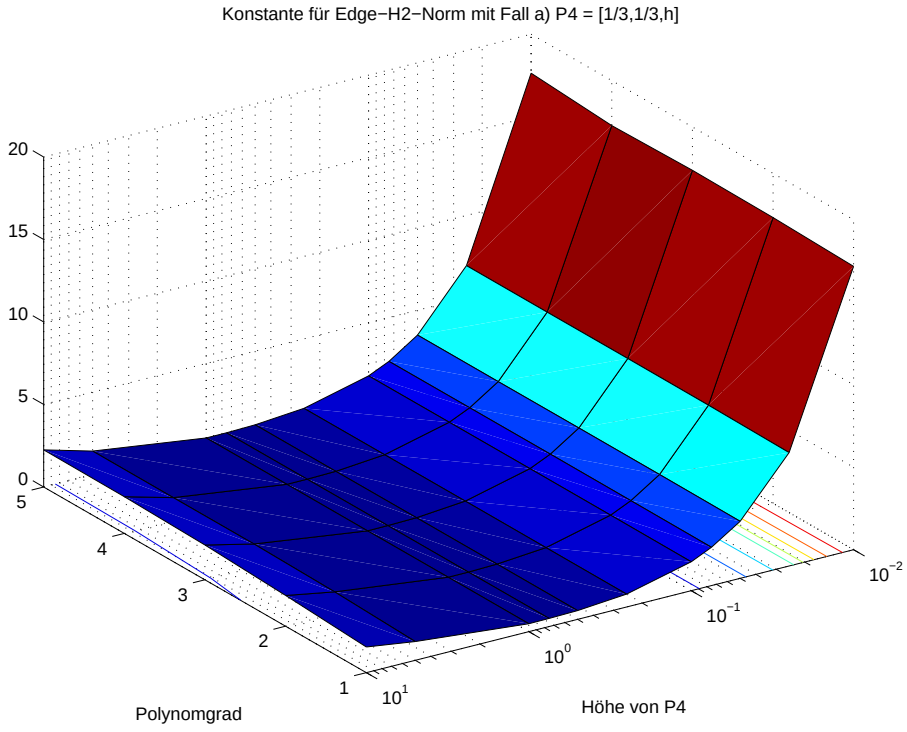

Abbildung 3.7: Datei szen42.eps

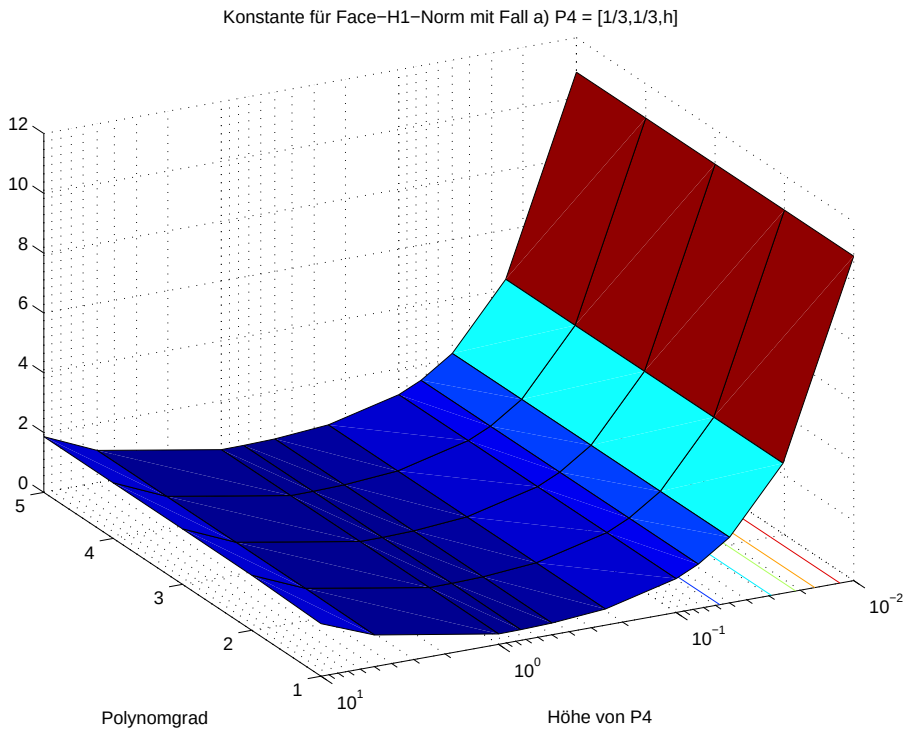

Abbildung 3.8: Datei szen43.eps

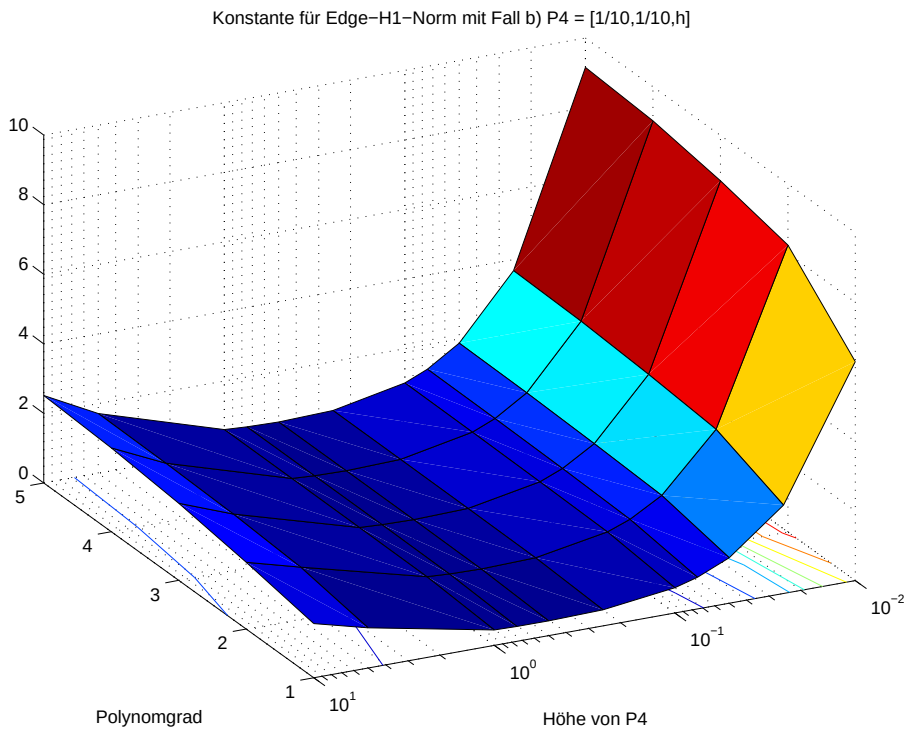

Abbildung 3.9: Datei szen51.eps

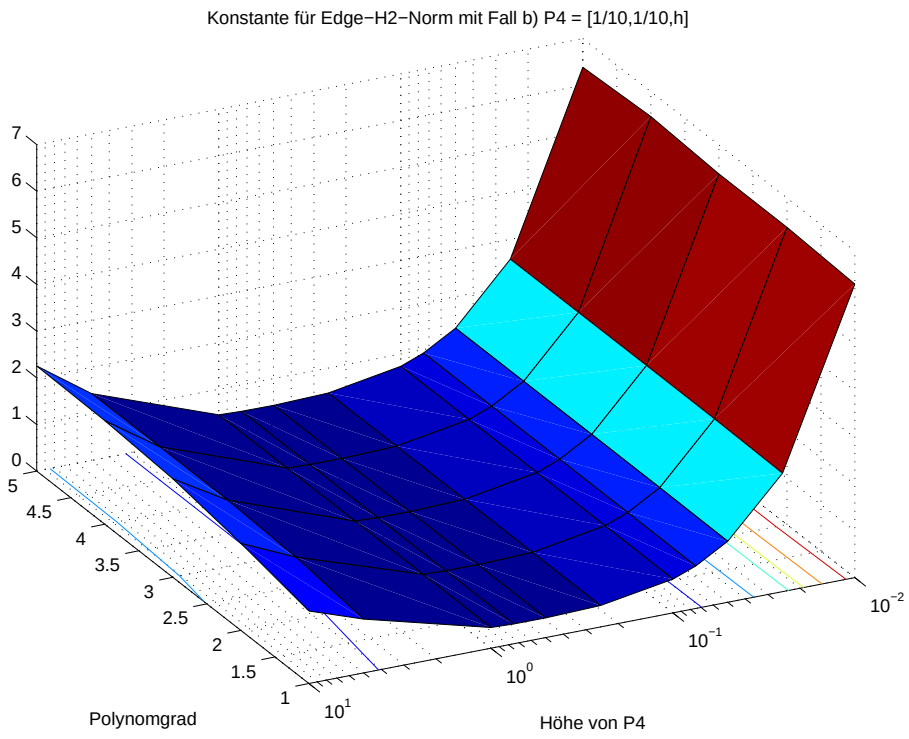

Abbildung 3.10: Datei szen52.eps

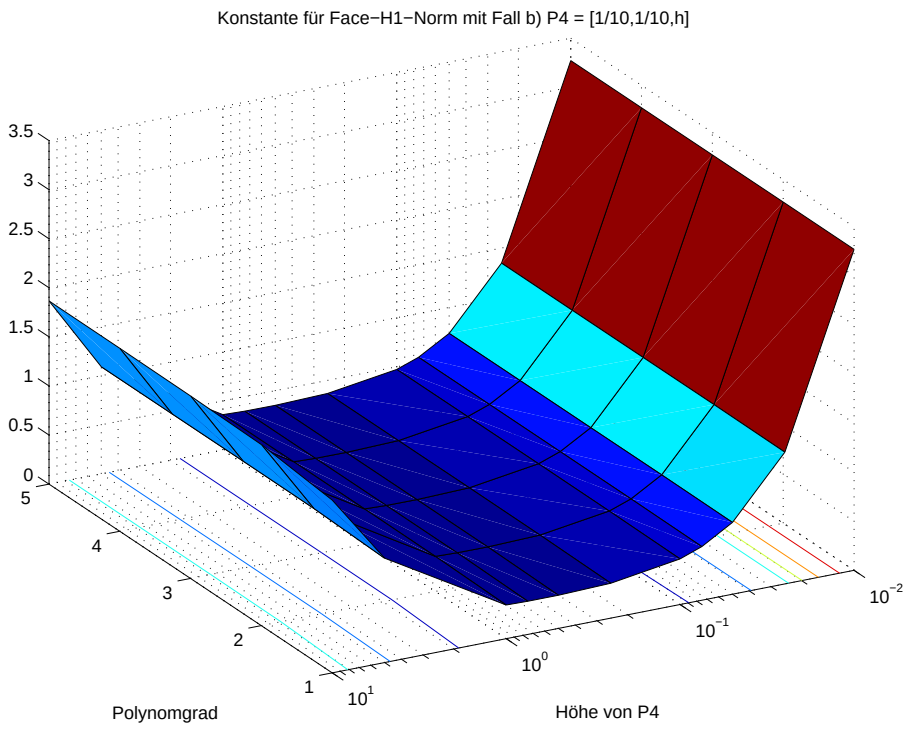

Abbildung 3.11: Datei szen53.eps

Kapitel 4

Winkel- und Längenabhängigkeit

4.1 Berechnung

Mit der Datei `get_Tetraeders.m` wird ein Datensatz von 11'054 Tetraedern generiert, bei denen die Punkte $P_1 = (0, 0, 0)$, $P_2 = (1, 0, 0)$ fest sind, sowie die Kante P_1P_2 die längste des Tetraeders ist, welche in Tetraeder.prot gespeichert werden. Für diese Tetraeder werden mit `get_const.m` die Konstanten für m=1 bis 6 berechnet und in `Konstants_m1.prot` bis `Konstants_m6.prot` gespeichert. Die verschiedenen Winkel, Kantenlängen und Flächen des Tetraeders werden mit `get_Geometrie.m` berechnet und in Geometrie.prot gespeichert.

Datei `get_const.m`

```
function f = get_const(f)
% Berechnet die Konstanten für die Tetraeder
% im file Tetraeder.prot
file = 'Tetraeder.prot';
TT = dlmread(file,',');
P1 = [0,0,0];P2 = [1,0,0];
m=1;% 2,3,4,5,6
file = 'Konstants_m1.prot';
fid = fopen(file,'w');
for i=1:length(TT)
    P3 = TT(i,1:3);P4 = TT(i,4:6);
    [C1_1(i),C1_2(i),C2(i)]= get_C1C2(P1,P2,P3,P4,m);
    fprintf(fid,'%f,%f,%f\n',C1_1(i),C1_2(i),C2(i));
```

```
end
fclose(fid);
```

Datei `get_Geometrie.m`

```
function T = get_Geometrie(f)
% Output :
% T : Matrix(11054 x 34)
% T(i,1:6)  = Flächenwinkel  (6)(ansteigend sortiert)
% T(i,7:18) = Kantenwinkel   (12)(ansteigend sortiert)
% T(i,19:24) = Kantenlängen   ( 6)(ansteigend sortiert)
% T(i,25:28) = Dreiecksflächen( 4)(ansteigend sortiert)
% Input :
% TT : Matrix(11054 x 6 )
% P1 = (0,0,0), P2 = (1,0,0)
% TT(i,1:3)   = P3
% TT(i,4:6)   = P4
file = 'Tetraeder.prot';
TT = dlmread(file,',');
P1 = [0,0,0];P2 = [1,0,0];
for i=1:length(TT)
   P3 = [TT(i,1:3)];P4 = [TT(i,4:6)];
   T(i,1:6)  = sort([winkel_eb_eb(P1,P2,P3,P4)]);
   T(i,7:18) = sort([winkel_ka_ka(P1,P2,P3,P4)]);
   T(i,19:24) = sort([norm(P1-P2),norm(P1-P3),norm(P1-P4), ...
                  norm(P2-P3),norm(P2-P4),norm(P3-P4)] );
   T(i,25:28) = sort([area(P1,P2,P3),area(P1,P2,P4), ...
                  area(P1,P4,P3),area(P4,P2,P3)] );
end
```

4.2 Auswertung

Die Auswertung der 11'054 Tetraeder wird mit der Datei auswertung.m durchgeführt. Es wurden Vergleiche mit einer Variablen (Winkel, Kantenlänge) sowie auch mit zweien Variablen gemacht.

Datei auswertung.m

```
% Daten einlesen
m1  = dlmread('Konstants_m1.prot',',');
m2  = dlmread('Konstants_m2.prot',',');
m3  = dlmread('Konstants_m3.prot',',');
m4  = dlmread('Konstants_m4.prot',',');
m5  = dlmread('Konstants_m5.prot',',');
m6  = dlmread('Konstants_m6.prot',',');
geo = dlmread('Geometrie.prot',',');
Tet = dlmread('Tetraeder.prot',',');
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Konstanten
%          m=1     m=2     m=3     m=4     m=5     m=6
Edge_H1 = [m1(:,1),m2(:,1),m3(:,1),m4(:,1),m5(:,1),m6(:,1)];
Edge_H2 = [m1(:,2),m2(:,2),m3(:,2),m4(:,2),m5(:,2),m6(:,2)];
Face_H1 = [m1(:,3),m2(:,3),m3(:,3),m4(:,3),m5(:,3),m6(:,3)];
%-------------------------------------------------------------
% Geometrie
% geo : Matrix(11054 x 34)
% geo(i,1:6)   = Flächenwinkel  ( 6)(ansteigend sortiert)
% geo(i,7:18)  = Kantenwinkel   (12)(ansteigend sortiert)
% geo(i,19:24) = Kantenlängen   ( 6)(ansteigend sortiert)
%                ( Längste Kante == 1 )
% geo(i,25:28) = Dreiecksflächen( 4)(ansteigend sortiert)
%-------------------------------------------------------------
% Tetraeder
% Tet : Matrix(11054 x 6)
% P1 = (0,0,0) , P2 = (1,0,0)
% Tet(i,1:3) = P3
% Tet(i,4:6) = P4
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

```
% Auswertungen in 3D
%figure(1);
%index1=1;index2=2;
%ff=myplot3dmin(geo(:,index1 ),geo(:,index2),Edge_H1(:,6),25);
%title('Kantenelement-H1Norm');grid on;
%xlabel('Argument: 1');ylabel('Argument: 2');
% Auswertungen in 2D
%figure(2)
%index=25;
%ff=myplot2dmin( geo(:,index ) , Edge_H1(:,6),Edge_H2(:,6),Face_H1(:,6),15)
%title('Bild 1');legend('Edge-H1','Edge-H2','Face-H1');grid on;
%xlabel('Kleinste Dreiecksfläche');ylabel('Konstanten');
clear all
```

Betrachten wir zuerst die Abhängigkeiten der Konstanten von einer Variablen (Abb.4.1 bis Abb.4.4). Dabei interessiert uns, inwiefern stumpfe beziehungsweise spitze Kanten- und Flächenwinkel problematisch sind.

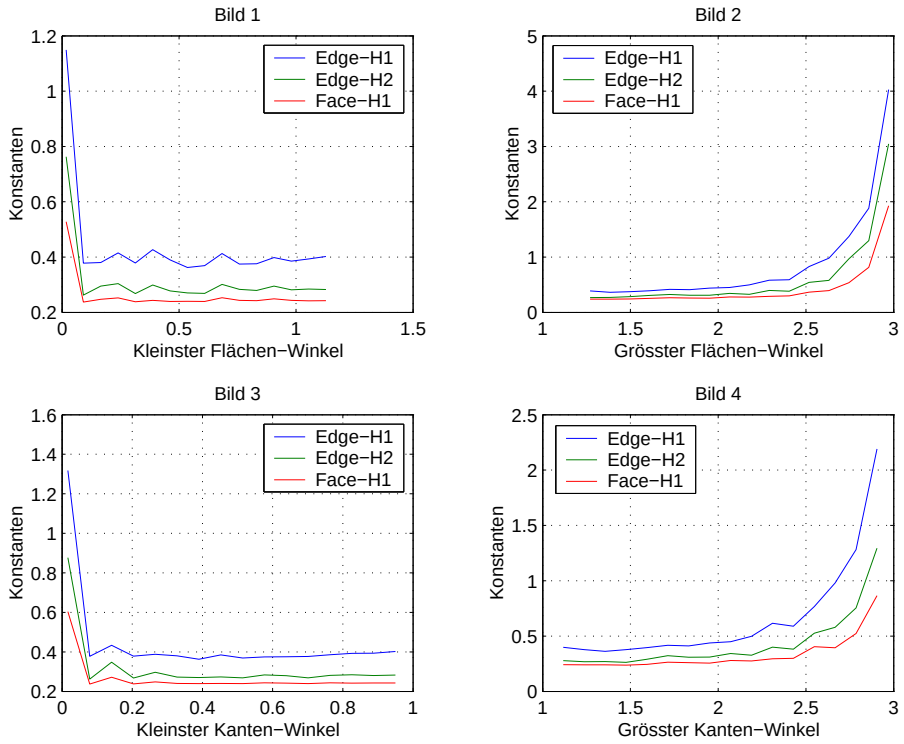

Abbildung 4.1: Datei 2d1.eps : Minimalwerte

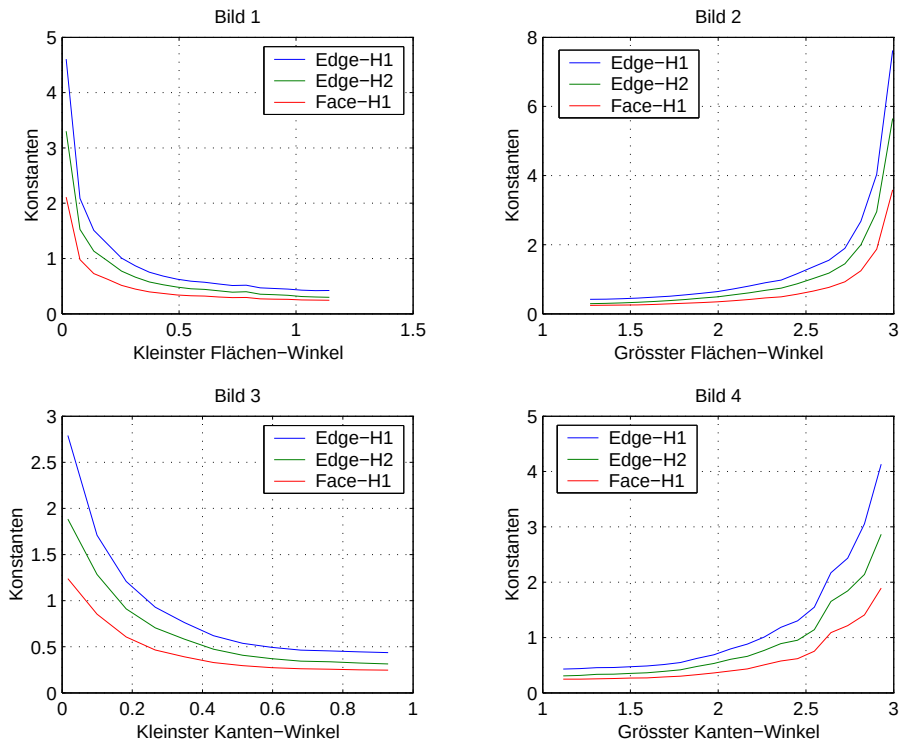

Abbildung 4.2: Datei 2d3.eps : Durchschnittswerte

Je grösser die stumpfesten Kanten- und Flächenwinkel werden, desto schlechter werden die Interpolationskonstanten. Bei der Betrachtung der Minimalwerte der Konstanten sieht man, dass die spitzen Winkel erst ab einer bestimmten Grösse kritisch werden. Man kann aber für die spitzesten Winkel sagen, dass je kleiner der Winkel, desto grösser werden die Interpolationskonstanten.

Die folgenden Visualisierungen der Fehlerkonstanten für 2 Variablen auf den nächsten Seiten werden nur für die Abschätzungen der Kantenelemente mit der H1-Norm dargestellt, da sie für die Kantenelemente mit der H2-Norm sowie für die Flächenelemente mit der H1-Norm ähnlich sind. Für Werte der Konstanten gleich Null in den Graphiken existieren keine Tetraeder, welche diese Bedingungen erfüllen, und sind nur aus Visualisierungsgründen vorhanden.

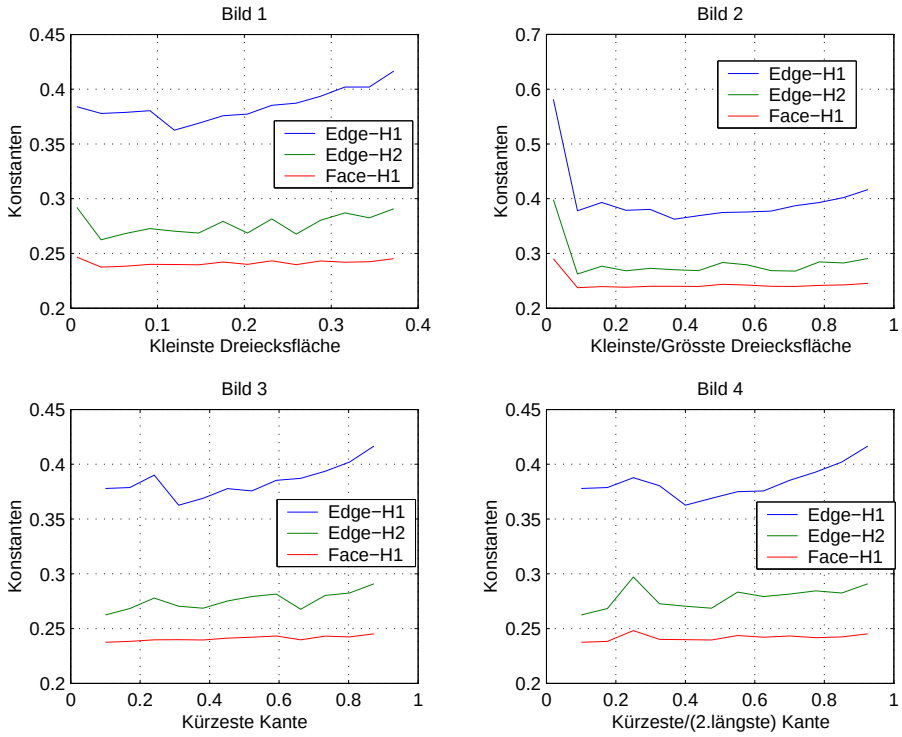

Abbildung 4.3: Datei 2d3.eps : Minimalwerte

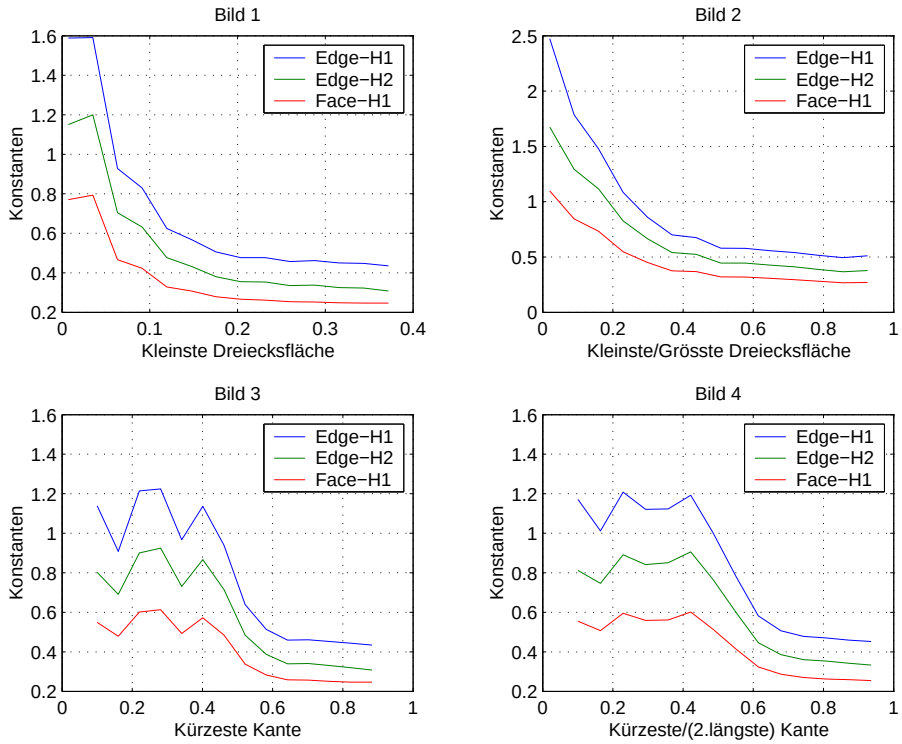

Abbildung 4.4: Datei 2d4.eps : Durchschnittswerte

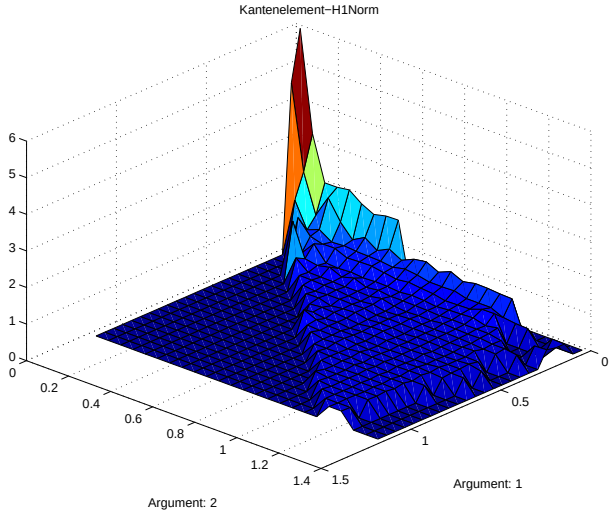

Abbildung 4.5: Datei 3d1.eps

Argument 1 : Kleinster Flächen-Winkel

Argument 2 : 2.kleinster Flächen-Winkel

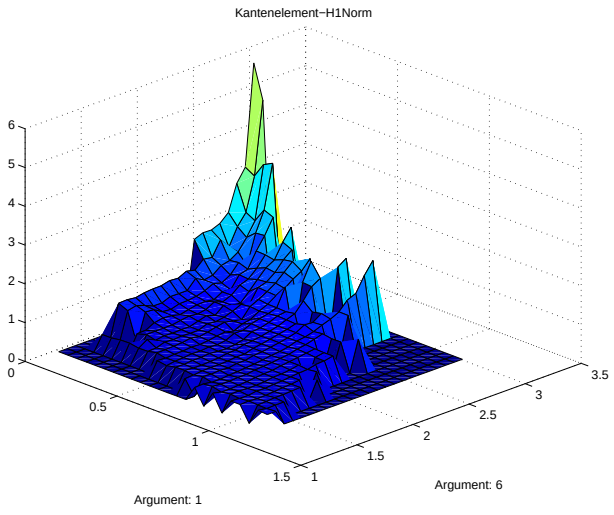

Abbildung 4.6: Datei 3d2.eps

Argument 1 : Kleinster Flächen-Winkel

Argument 6 : Grösster Flächen-Winkel

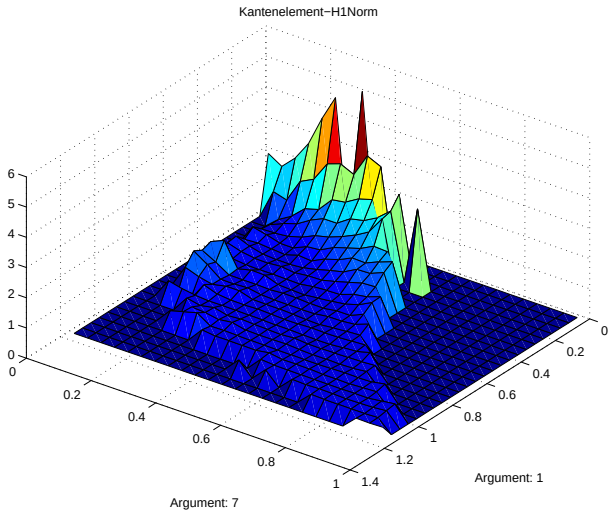

Abbildung 4.7: Datei 3d3.eps

Argument 1 : Kleinster Flächen-Winkel

Argument 7 : Kleinster Kanten-Winkel

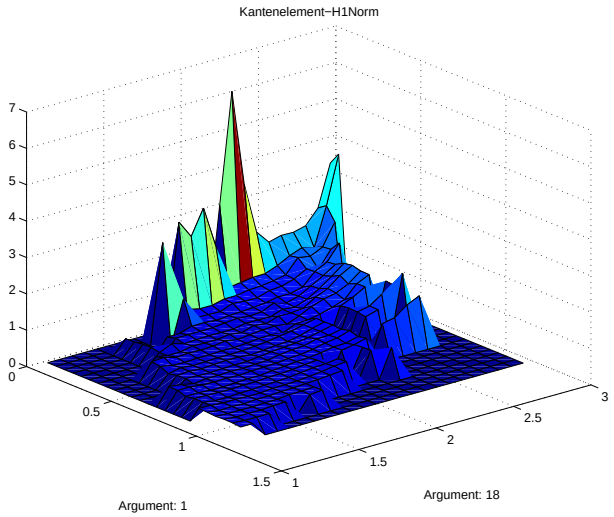

Abbildung 4.8: Datei 3d4.eps

Argument 1 : Kleinster Flächen-Winkel

Argument 18 : Grösster Kanten-Winkel

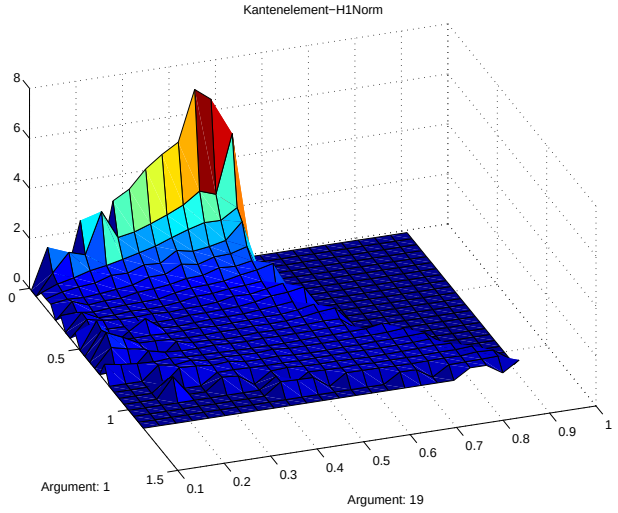

Abbildung 4.9: Datei 3d5.eps

Argument 1 : Kleinster Flächen-Winkel

Argument 19 : Kürzeste Kante

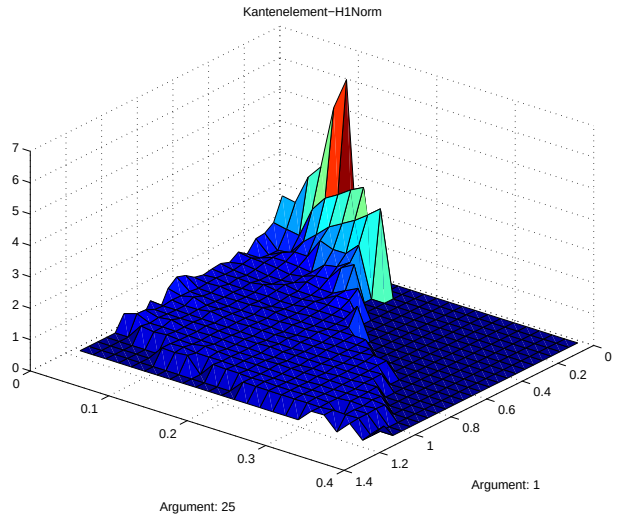

Abbildung 4.10: Datei 3d6.eps

Argument 1 : Kleinster Flächen-Winkel

Argument 25 : Kleinste Fläche

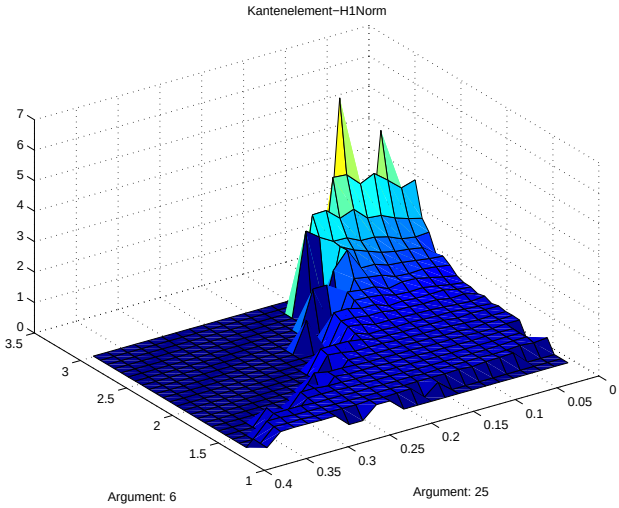

Abbildung 4.11: Datei 3d7.eps

Argument 6 : Grösster Flächen-Winkel

Argument 25 : Kleinste Fläche

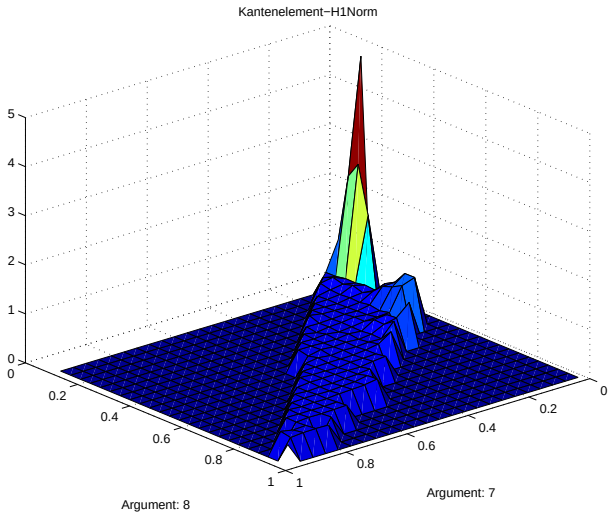

Abbildung 4.12: Datei 3d8.eps

Argument 7 : Kleinster Kanten-Winkel

Argument 8 : 2.kleinster Kanten-Winkel

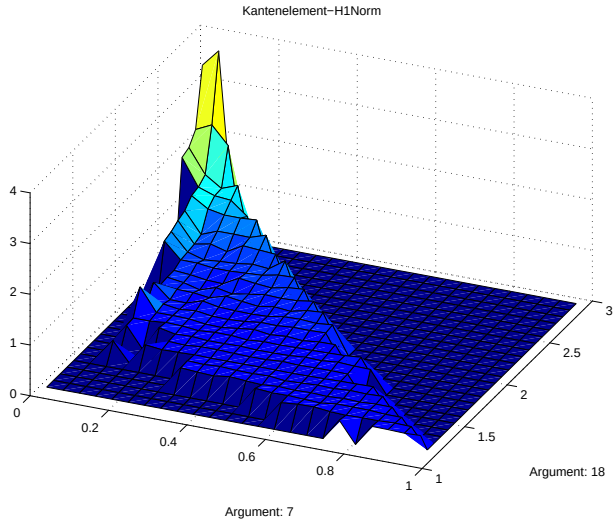

Abbildung 4.13: Datei 3d9.eps

Argument 7 : Kleinster Kanten-Winkel

Argument 18 : Grösster Kanten-Winkel

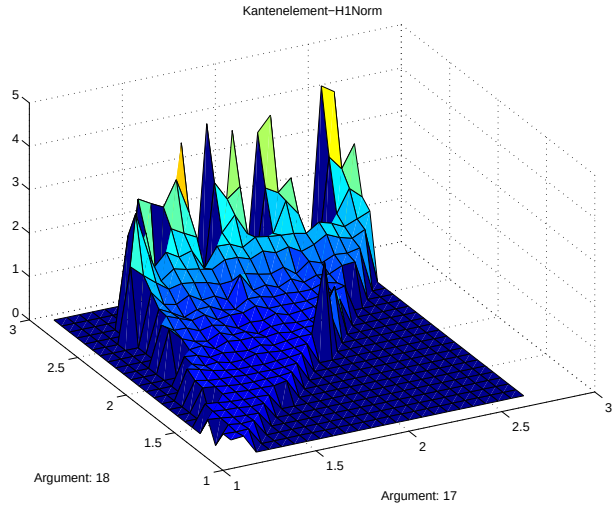

Abbildung 4.14: Datei 3d10.eps

Argument 17 : 2.grösster Kanten-Winkel

Argument 18 : Grösster Kanten-Winkel

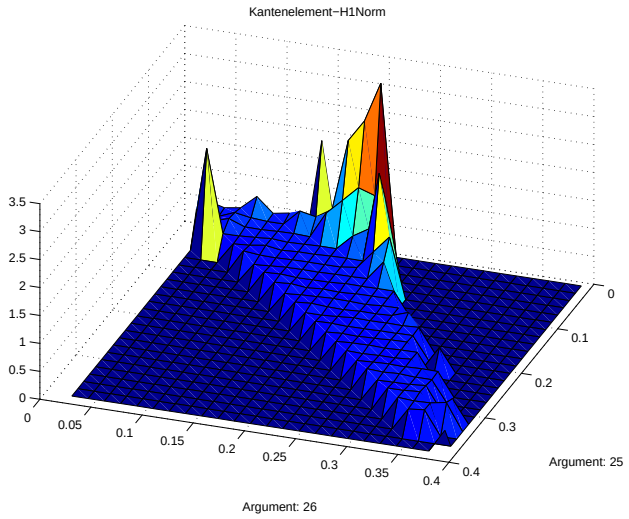

Abbildung 4.15: Datei 3d11.eps

Argument 25 : Kleinste Fläche

Argument 26 : 2.kleinste Fläche

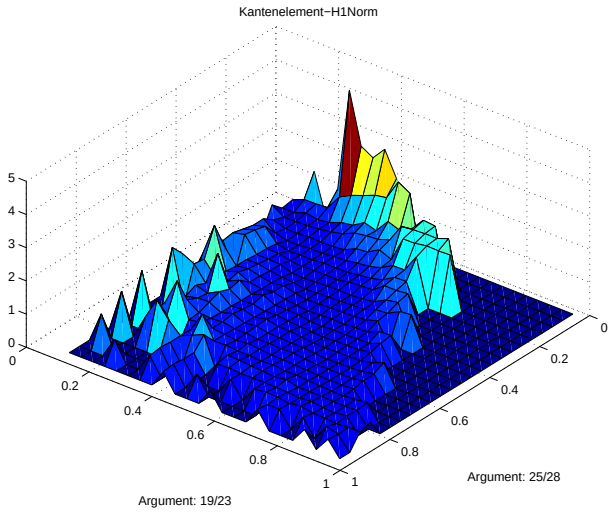

Abbildung 4.16: Datei 3d12.eps

Argument 25/28 : Kleinste Fläche / Grösste Fläche

Argument 19/23 : Kürzeste Kante / 2.längste Kante

Anhang A

Literatur

1. R. Hiptmair, *Finite elements in computational electromagnetism*, Acta Numerica,(2002), pp. 237-339

2. Vorlesungsskript der Numerik der partiellen Differentialgleichungen von R. Hiptmair