3. Lineare Finite Elemente am Beispiel des Poisson-Problems

Modellproblem: Poisson-Gleichung aus 2.1

$$\begin{cases} -\Delta u = f & \text{in } \Omega \subseteq \mathbb{R}^2 \quad (d=2) \\ u = 0 & \text{auf } \Gamma \end{cases}$$

Variationelle Formulierung: Finde $u \in H_0^1(\Omega)$ mit

$$a(u,v) = \ell(v) \qquad \forall v \in H_0^1(\Omega)$$

wobei

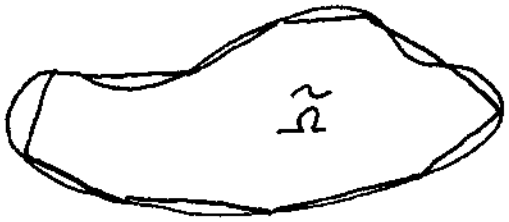
$$a(u,v) = \int_\Omega \partial_x u(x,y) \cdot \partial_x v(x,y) + \partial_y u(x,y) \cdot \partial_y v(x,y) \, d(x,y)$$

$$\ell(v) = \int_\Omega f(x,y) v(x,y) \, d(x,y)$$

(Schreibe jetzt (x,y) anstelle von (x_1, x_2))

(a) Triangulierung

Approximiere den Rand von Ω durch ein Polygon mit Inneren $\tilde{\Omega}$:



Unterteile $\tilde{\Omega}$ in $m \in \mathbb{N}$ Dreiecke. Die Ecken eines Dreiecks müssen auf den Ecken anderer Dreiecke liegen.

Richtig:  Falsch: 

Bezeichnung: T_k k-tes Dreieck, $\overline{\tilde{\Omega}} = \bigcup_{k=1}^{m} T_k$

$P_i = (x_i, y_i)$ Ecken in $\tilde{\Omega}$ (ohne Rand), $i = 1, \dots, N$

Algorithmen zur Erzeugung von Triangulierungen $\longrightarrow$ später

(b) Basisfunktionen

Sei φ_i Hutfunktion, d.h. $\varphi_i|_{T_k}$ ist linear und

$$\varphi_i(P_j) = \begin{cases} 1 & \text{falls } i=j \\ 0 & \text{sonst} \end{cases}$$

Wähle $V_N = \operatorname{span}\{\varphi_1, \dots, \varphi_N\}$

Suche Lösung der Form $u_N = \sum_{i=1}^{N} \hat{u}_i \varphi_i$

Es gilt $u_N(P_j) = \sum_{i=1}^{N} \hat{u}_i \varphi_i(P_j) = \hat{u}_j$.

Man kann zeigen, dass $V_N \subseteq H_0^1(\tilde{\Omega})$.

(c) Aufstellen des Gleichungssystems

Steifigkeitsmatrix:

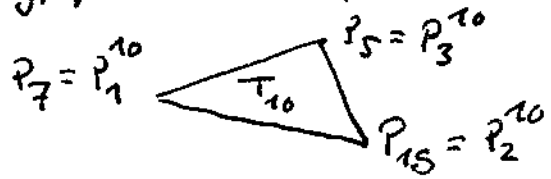
$$a(\varphi_i, \varphi_j) = \sum_{k=1}^{m} \underbrace{\int_{T_k} \partial_x \varphi_i(x,y) \cdot \partial_x \varphi_j(x,y) + \partial_y \varphi_i(x,y) \cdot \partial_y \varphi_j(x,y) \, d(x,y)}_{=: a_{T_k}(\varphi_i, \varphi_j)}$$

Es gilt $a_{T_k}(\varphi_i, \varphi_j) = 0$ falls $\varphi_i|_{T_k} \equiv 0$ oder $\varphi_j|_{T_k} \equiv 0$.

$\Rightarrow$ Die Steifigkeitsmatrix ist dünn besetzt.

Betrachte nun ein einzelnes Dreieck T_k mit $k \in \{1, \dots, m\}$.

Verwende lokale Indizierung: Die drei Ecken von T_k seien $P_i^k = (x_i^k, y_i^k)$ mit entsprechenden Basisfunktionen φ_i^k $(i = 1,2,3)$.



Sei $\varphi_i^k(x,y) = \alpha_i + \beta_i x + \gamma_i y$ für $(x,y) \in T_k$, $i = 1,2,3$

Fordere $\varphi_i^k(P_j^k) = \begin{cases} 1 & \text{falls } i = j \\ 0 & \text{sonst} \end{cases}$

$$\Leftrightarrow \alpha_i + \beta_i x_j^k + \gamma_i y_j^k = \begin{cases} 1 & \text{falls } i = j \\ 0 & \text{sonst} \end{cases}$$

$$\Leftrightarrow \begin{pmatrix} 1 & x_1^k & y_1^k \\ 1 & x_2^k & y_2^k \\ 1 & x_3^k & y_3^k \end{pmatrix} \begin{pmatrix} \alpha_1 & \alpha_2 & \alpha_3 \\ \beta_1 & \beta_2 & \beta_3 \\ \gamma_1 & \gamma_2 & \gamma_3 \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} \qquad (*)$$

$$\Leftrightarrow \begin{pmatrix} \alpha_1 & \alpha_2 & \alpha_3 \\ \beta_1 & \beta_2 & \beta_3 \\ \gamma_1 & \gamma_2 & \gamma_3 \end{pmatrix} = \begin{pmatrix} 1 & x_1^k & y_1^k \\ 1 & x_2^k & y_2^k \\ 1 & x_3^k & y_3^k \end{pmatrix}^{-1}$$

$\partial_x \varphi_i^k(x,y) = \beta_i$ für $(x,y) \in \overline{T_k}$

$\partial_y \varphi_i^k(x,y) = \gamma_i$

$$\Rightarrow a_{\overline{T_k}}(\varphi_i^k, \varphi_j^k) = \int_{\overline{T_k}} \beta_i \beta_j + \gamma_i \gamma_j \, d(x,y) = |\overline{T_k}| \cdot (\beta_i \beta_j + \gamma_i \gamma_j)$$

mit $|\overline{T_k}| = \frac{1}{2} \left| \det \begin{pmatrix} 1 & x_1^k & y_1^k \\ 1 & x_2^k & y_2^k \\ 1 & x_3^k & y_3^k \end{pmatrix} \right|$ Fläche von $\overline{T_k}$

Lastvektor:

$$\ell(\varphi_i) = \sum_{k=1}^{m} \underbrace{\int_{\overline{T_k}} f(x,y) \varphi_i(x,y) \, d(x,y)}_{=: \ell_{\overline{T_k}}(\varphi_i)}$$

Approximiere das Integral durch eine Quadraturformel:

$$\int_{\overline{T_k}} g(x,y) \, d(x,y) \approx |\overline{T_k}| \cdot g(x_*^k, y_*^k)$$

wobei $x_*^k = \frac{1}{3}(x_1^k + x_2^k + x_3^k)$ und $y_*^k = \frac{1}{3}(y_1^k + y_2^k + y_3^k)$ die Koordinaten des Schwerpunkts sind. Es gilt für $\varphi_i|_{\overline{T_k}} \neq 0$

$$\varphi_i(x_*^k, y_*^k) = \alpha_i + \beta_i x_*^k + \gamma_i y_*^k$$

$$= \left(\tfrac{1}{3}, \tfrac{1}{3}, \tfrac{1}{3}\right) \underbrace{\begin{pmatrix} 1 & x_1^k & y_1^k \\ 1 & x_2^k & y_2^k \\ 1 & x_3^k & y_3^k \end{pmatrix} \begin{pmatrix} \alpha_i \\ \beta_i \\ \gamma_i \end{pmatrix}}_{= e_i \text{ wegen } (*)} = \frac{1}{3}$$

Approximiere also

$$l_{T_k}(\varphi_i) = \int_{T_k} f(x,y)\,\varphi_i(x,y)\,d(x,y) \approx |T_k|\, f(x_*^k, y_*^k)\cdot \frac{1}{3}$$

$$\Rightarrow l(\varphi_i) = \sum_{k=1}^{m} l_{T_k}(\varphi_i) \approx \frac{1}{3} \sum_{\substack{k=1 \\ P_i \in T_k}}^{m} |T_k| \cdot f(x_*^k, y_*^k)$$

(d) Lösung des Gleichungssystems

Falls N klein: Cholesky-Verfahren

Falls N groß: Mehrgitterverfahren (→ später)
CG-Verfahren mit Vorkonditionierung

(e) Implementierung in Matlab

Triangulierung: Wird von uns bereitgestellt

Hier: load MyGrids\mygrid01.mat
→ EdgesDir, EdgesNeu, P, T

$P \in \mathbb{R}^{\bar{N}\times 2}$, $\bar{N}$ Anzahl aller Punkte

$P(k,:)$ Koordinaten des k-ten Punkts

$T \in \mathbb{N}^{m\times 3}$ gibt an, welche Punkte zu einem Dreieck gehören

$T(17,:) = [6\ 10\ 7]$ → $P(6,:)$, $P(10,:)$, $P(7,:)$

EdgesDir $\in \mathbb{R}^{\dots \times 2}$ Kanten, auf denen Dirichlet-Randbedingungen gesetzt werden

EdgesNeu $\in \mathbb{R}^{\dots \times 2}$ analog für Neumann-Randbedingungen

In Aufgabe 2: EdgesNeu = [] (leere Liste, wird nicht verwendet)

Visualisierung:

trimesh(T, x, y, u) x, y Koordinaten der Punkte
trisurf(T, x, y, u) u Werte der Funktion an den Punkten
view(2) → Ansicht „von oben"

Berechnung der Steifigkeitsmatrix und des Lastvektors:
Gehe nicht punktweise vor, sondern dreiecksweise

for $k = 1 : m$ % m = Anzahl der Dreiecke

Berechne $\tilde{a}_{ij} = a_{T_k}(\varphi_i^k, \varphi_j^k)$, $i,j = 1,2,3$

$\tilde{b}_i = \ell_{T_k}(\varphi_i^k)$

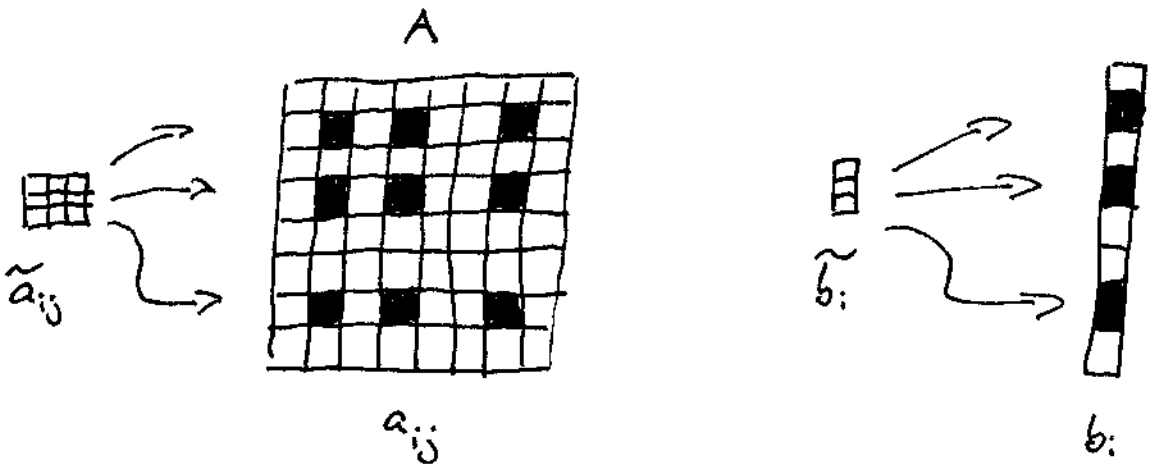
und addiere zu den entsprechenden Einträgen hinzu:

$a_{\psi(i),\psi(j)} = a_{\psi(i),\psi(j)} + \tilde{a}_{ij}$ $i,j = 1,2,3$

$b_{\psi(i)} = b_{\psi(i)} + \tilde{b}_i$

Dabei ist ψ die Abbildung vom lokalen auf den globalen Index, d.h.

$\varphi_i^k = \varphi_{\psi(i)}$, $P_i^k = P_{\psi(i)}$



Man kann zuerst die Steifigkeitsmatrix für alle $\tilde{N}$ Punkte aufstellen und dann die Spalten und Zeilen, die zu den $\tilde{N}-N$ Randpunkten gehören, wieder entfernen.

Kein großer Effizienzverlust, falls $\tilde{N} \approx N$

Funktioniert so nur bei homogenen Dirichlet-Randbedingungen

Weitere Hilfreiche Befehle: size, length, det, inv, unique, setdiff, ismember, x.*y, x.^2

Approximation des L^2-Fehlers durch Quadratur:

$$\varepsilon(x,y) := |\underbrace{u(x,y)}_{\text{exakt}} - \underbrace{u_N(x,y)}_{\text{Approx}}|^2$$

$$\|u-u_N\|_{L^2(\Omega)} = \left(\int_\Omega \varepsilon(x,y)\, d(x,y) \right)^{1/2} = \left(\sum_{k=1}^{m} \int_{T_k} \varepsilon(x,y)\, d(x,y) \right)^{1/2}$$

$$\approx \left(\sum_{k=1}^{m} \frac{|T_k|}{3} \left(\varepsilon(P_1^k) + \varepsilon(P_2^k) + \varepsilon(P_3^k) \right) \right)^{1/2}$$