Prof. Dr. Angela Kunoth
Samuel Leweke

Abschlussprojekt: Immersed Penalized Boundary Methode

In Anwendungen aus Naturwissenschaften und Ingenieurwesen sind PDEs meistens auf komplexen Gebieten Ω gegeben, die nicht $[0,1]^d$ entsprechen. Folglich können Tensorprodukt Ansätze für die diskreten Funktionenräume nicht verwendet werden, da das Gebiet nicht rechteckig ist. Dieses Problem kann umgangen werden, indem das *physikalische Gebiet* Ω in ein erweitertes Gebiet $\hat{\Omega} \supset \Omega$ eingebettet wird, welches das kleinste Ω enthaltende Rechteck ist. Nun müssen jedoch die Randbedingungen für die Lösung im erweiterten Gebiet $\hat{\Omega}$ erzwungen werden. Neben Methoden mit Lagrange-Multiplikator und schwacher Aufprägung der Randbedingung eignet sich dafür die Strafmethode. Hierbei wird die PDE als Optimierungsproblem formuliert und der Fehler auf dem Rand als weiterer (Straf-)Term an die Zielfunktion angehangen.

Die Kombination dieser beiden Ideen, die eingebettete (*immersed*) Rand-Strafmethode (*penalized boundary*), soll in diesem Projekt zur Lösung der PDE

$$\begin{aligned} -\Delta u &= f \quad && \text{in } \Omega \subset \hat{\Omega} \coloneqq [0,1]^2 \\ u &= g \quad && \text{auf } \Gamma \coloneqq \partial\Omega \end{aligned} \tag{1}$$

verwendet werden. Das Gebiet Ω wird dabei in das Rechengebiet $\hat{\Omega} = [0,1]^2$ eingebettet. Der Rand Γ von Ω ist durch eine B-Spline Kurve $\mathbf{G}\colon [0,1] \to \mathbb{R}^2$ der Ordnung k_G parametrisiert:

$$\mathbf{G}(t) = \sum_{i=1}^{N_G} \mathbf{C}_i N_{i,k_G}(t), \qquad t \in [0,1], \tag{2}$$

wobei $\mathbf{C}_i \in \mathbb{R}^2$ *Kontrollpunkte* sind und N_G die Anzahl der Basisfunktionen bezeichnet. Zu dieser Splinekurve gehört ein entsprechender erweiterter Knotenvektor θ_G mit $N_G + k_G$ Einträgen.

Als diskreter Funktionenraum $V_J \subset H^1(\Omega)$ wird ein Tensorprodukt B-Spline der Ordnung k auf einem uniformen Gitter der Feinheit 2^{-J} verwendet:

$$V_J \coloneqq \operatorname{span}\left\{\mathbf{x} \mapsto N_{\mathbf{i},k}(\mathbf{x}) = N_{i_1,k}(x_1) N_{i_2,k}(x_2) : \mathbf{i} = (i_1, i_2) \in \mathcal{I}_J \times \mathcal{I}_J,\, \mathcal{I}_J \coloneqq \{1, \dots, 2^J + k - 1\}^2\right\}. \tag{3}$$

Der zugehörige Knotenvektor θ_J ist gegeben durch

$$\theta_{J,1} = \ldots = \theta_{J,k} = 0 < 2^{-J} < 2 \cdot 2^{-J} < 3 \cdot 2^{-J} < \ldots < 1 - 2^{-J} < 1 = \theta_{J,n+1} = \ldots = \theta_{J,n+k}. \tag{4}$$

Ein Galerkin-Verfahren für (1) auf dem erweiterten Gebiet $\hat{\Omega}$ mit dem Ansatz

$$u_J = \sum_{\mathbf{i} \in \mathcal{I}_J \times \mathcal{I}_J} (\mathbf{u}_J)_{\mathbf{i}}\, N_{\mathbf{i},k} \tag{5}$$

führt auf ein lineares Gleichungssystem $\mathbf{A}\mathbf{u}_J = \mathbf{b}$ für die Koeffizienten $\mathbf{u}_J$ mit

$$a_{\mathbf{i},\mathbf{j}} = \int_{\hat{\Omega}} \nabla N_{\mathbf{j}} \cdot \nabla N_{\mathbf{i}}\, \mathrm{d}\mathbf{x}, \qquad b_{\mathbf{i}} = \int_{\hat{\Omega}} f N_{\mathbf{i}}\, \mathrm{d}\mathbf{x} \qquad \forall \mathbf{i}, \mathbf{j} \in \mathcal{I}_J^2. \tag{6}$$

Bei der *Immersed Penalized Boundary Methode* wird die numerische Lösung des Problems (1) auf dem physikalischen Gebiet Ω als Minimierer eines Funktionals $\Phi\colon \mathbb{R}^{\left(2^J+k-1\right)^2} \to \mathbb{R}$ formuliert:

$$\text{Finde } \mathbf{u}_J \in \mathbb{R}^{\left(2^J+k-1\right)^2}: \quad \Phi(\mathbf{u}) \coloneqq \sum_{i=1}^{2^J+k-1} \left[(\mathbf{A}\mathbf{u} - \mathbf{b})_i\right]^2 + \lambda \sum_{i=1}^{N_\Gamma} \left[g(\gamma_i) - u_J(\gamma_i)\right]^2 \to \min. \tag{7}$$

Hierbei ist $\lambda > 0$ der Strafparameter und $(\gamma_i)_{i=1}^{N_\Gamma} \subset \Gamma$ eine endliche Menge von Punkten auf dem Rand Γ.

Aufgaben

a) Bestimmen Sie den Gradienten von Φ aus (7) und folgern Sie das lineare Gleichungssystem

$$\left(\mathbf{A}^T\mathbf{A} + \lambda\mathbf{K}^T\mathbf{K}\right)\mathbf{u}_J = \mathbf{A}^T\mathbf{b} + \lambda\mathbf{K}^T\mathbf{g} \tag{8}$$

für den Minimierer $\mathbf{u}_J$ mit der (verallgemeinerten) Vandermonde-Matrix $\mathbf{K}$ und dem Vektor $\mathbf{g}$:

$$\mathbf{K} = \begin{pmatrix} N_{(1,1),k}(\gamma_1) & \dots & N_{(2^J+k-1,2^J+k-1),k}(\gamma_1) \\ \vdots & \ddots & \vdots \\ N_{(1,1),k}(\gamma_{N_\Gamma}) & \dots & N_{(2^J+k-1,2^J+k-1),k}(\gamma_{N_\Gamma}) \end{pmatrix}, \qquad \mathbf{g} = \begin{pmatrix} g(\gamma_1) \\ \vdots \\ g(\gamma_{N_\Gamma}) \end{pmatrix}. \tag{9}$$

b) Schreiben Sie eine Matlab-Funktion `K = VandermondeNik2D(gammaX, gammaY, knots, order)`, welche die Matrix `K` $= \mathbf{K}$ aus (9) mit den Tensorprodukt B-Spline Basisfunktionen aus (3) aufbaut. Die Auswertungspunkte (`gammaX`, `gammaY`) sind durch die Vektoren `gammaX`, `gammaY` gegeben.

Hinweis: Verwenden Sie die lexikographische Sortierung für die Indexmenge $\mathcal{I}_J \times \mathcal{I}_J$.

c) Schreiben Sie eine Matlab-Funktion `v = EvalSpline2D(x, y, knots, order, coeffs)`, welche den B-Spline

$$\sum_{i_1,i_2=1}^{\#\texttt{knots}-\texttt{order}} \texttt{coeffs}_{(i_1,i_2)}\, N_{i_1,\texttt{order}}(x) N_{i_2,\texttt{order}}(y) \tag{10}$$

an den Punkten (`x`, `y`) auswertet. Die Punkte sind durch beliebige Arrays `x`, `y` (mit gleicher Größe) gegeben und der Rückgabewert `v` ist ein Array gleicher Größe.

d) Schreiben Sie eine Matlab-Funktion `[x, y] = SplineCurve(t, knotsG, orderG, ctrlPts)`, die die Splinekurve $\mathbf{G}$ aus (2) auswertet. Die Kontrollpunkte $\mathbf{C}_i$ sind in der zweispaltigen Matrix `ctrlPts` gegeben, wobei jede Zeile einen Punkt enthält. Die Abbildung $\mathbf{G}$ soll an den Punkten im Vektor `t` ausgewertet werden Das Ergebnis sind die Vektoren `x` und `y`, welche die gleiche Größe wie `t` haben.

e) Schreiben Sie eine Matlab-Funktion `coeffs = ImmersedPenaltyBoundary(J, order, knotsG, orderG, ctrlPts, nBnd, f, g)`, die (8) löst und den Koeffizientenvektor $\mathbf{u}_J$ in `coeffs` zurückgibt. Die Parameter J und $k =$ `order` bestimmen den B-Spline Raum V_J aus (3). Die Randkurve $\mathbf{G}$ der Ordnung $k_G =$ `orderG` ist durch den Knotenvektor `knotsG` und die Kontrollpunkte `ctrlPts` gegeben.

Die Randpunkte $(\gamma_i)_{i=1}^{\texttt{nBnd}}$ sollen durch das Bild $\gamma_i = G(t_i)$ der `nBnd` äquidistanten Punkte $t_i = (i-1)/\texttt{nBnd}$ mit $i = 1, \dots,$ `nBnd` erzeugt werden.

Die rechte Seite der PDE f ist als Function-Handle `f` und die Dirichlet-Randfunktion g als Function-Handle `g` gegeben.

Hinweis: Verwenden Sie die lexikographische Sortierung für `coeffs` $= \mathbf{u}_J$.

f) Schreiben Sie eine Matlab-Funktion `[errInt, errBnd] = ErrorEstimate(J, order, knotsG, orderG, ctrlPts, JErr, nBndErr, coeffs, u)`, die den Fehler der numerischen Lösung u_J in der L_∞-Norm auf Ω und dem Rand Γ schätzt. Werten Sie dazu $|u - u_J|$ für eine große Menge an Punkten aus und nehmen Sie das Maximum dieser Funktionsauswertungen. Die numerische Lösung u_J ist dabei durch `J`, `order` und `coeffs` gegeben und die exakte Lösung u durch das Function-Handle `u`.

Gehen Sie für den Randfehler wie in Teil e) vor, indem Sie `nBndErr` äquidistante Punkte auf $[0,1]$ mit $\mathbf{G}$ nach Γ abbilden und dort den absoluten Fehler auswerten. Das Maximum dieser Auswertungen soll in `errBnd` zurückgegeben werden.

Für den inneren Fehler auf Ω erstellen Sie zunächst ein äquidistantes Gitter auf $\hat{\Omega}$ zum Level `Jerr` (Gitterweite $2^{-\texttt{Jerr}}$). Für ein konvexes Gebiet Ω stellt die konvexe Hülle der zuvor berechneten `nBndErr` Randpunkte eine gute Approximation dieses Gebiets dar. Werten Sie nun den absoluten Fehler nur an denjenigen Punkten aus dem Gitter aus, die in der konvexen Hülle liegen (verwenden Sie `inpolygon()` mit den Randpunkten). Geben Sie das Maximum dieser Funktionsauswertungen in `errInt` zurück.

Hinweis: Vermeiden Sie, soweit möglich, dass die Auswertungspunkte für den $L_\infty(\Omega)$-Fehler auf dem Rand Ω liegen. Speichern Sie die Auswertungspunkte in Ω für spätere Fehlerauswertungen zwischen (das

Filtern der Punkte mittels `inpolygon()` kann länger dauern und erfordert eventuell eine geringere Zahl an Randpunkten).

g) Betrachten Sie das Problem (1) mit

$$\begin{aligned} u(\mathbf{x}) &= \sin\left(2\pi\left(x_1^2 - x_2^2\right)\right), \\ f(\mathbf{x}) &= 16\pi^2\left(x_1^2 + x_2^2\right)\sin\left(2\pi\left(x_1^2 - x_2^2\right)\right) = -\Delta u(\mathbf{x}), \\ g(\mathbf{x}) &= u(\mathbf{x}), \\ \theta_G &= (0,0,0,0.35,0.5,0.75,1,1,1)^T, \qquad k_G = 3, \\ \texttt{ctrlPts} &= [0,0;1,0;1,1;0,1;0,0.5;0,0]. \end{aligned} \tag{11}$$

Lösen Sie das Problem numerisch für $k = 3, \ldots, 6$ und $J = 3, \ldots, 6$.

Bestimmen Sie dafür zunächst eine geeignete Zahl an Randpunkten N_Γ und einen Strafparameter $\lambda > 0$. Fertigen Sie dazu jeweils eine Parameterstudie an, in der Sie für $k = 4$ und $J = 6$ den jeweiligen Parameter variieren und den anderen konstant halten. Betrachten Sie den Fehler der numerischen Lösung $u_{J,k}$ in Ω und auf Γ und wählen Sie denjenigen Parameterwert, der die Fehler minimiert. Geben Sie eine Tabelle pro Studie mit den Fehlern für jeden untersuchten Parameterwert aus. Wählen Sie für die N_Γ-Studie $\lambda = 1$ und für die λ-Studie $N_\Gamma = 400$, wobei $\lambda = 10^\ell$ als Zehnerpotenz angesetzt werden soll. Benutzen Sie `Jerr` $= 8$ und `nBndErr` $= 800$ für die Auswertung des Fehlers.

Berechnen Sie nun mit den zuvor bestimmten Parametern N_Γ, λ für jede Kombination (J, k) den Fehler der numerischen Lösung $u_{J,k}$ bezüglich der exakten Lösung u in Ω und auf Γ.

Plotten Sie für jede Ordnung k die beiden Fehlerkurven (Gitterweite 2^{-J} gegen Fehlernorm für $J = 3, \ldots, 6$) in einen gemeinsamen doppelt-logarithmischen (`loglog`) Konvergenzplot.

Schätzen Sie für jede Ordnung k und jeweils zwei aufeinanderfolgende Level $J-1$, J die Konvergenzrate

$$\operatorname{eoc}(J,k) = \frac{\ln\left(\frac{\|u_{J,k}-u\|}{\|u_{J-1,k}-u\|}\right)}{\ln\left(\frac{2^{-J}}{2^{-(J-1)}}\right)}$$

bezüglich beider Fehlernormen. Geben Sie eine Tabelle mit den (empirischen) Konvergenzraten $\operatorname{eoc}(J,k)$ für jede Norm aus.

Literatur

[Sch19] Larry L. Schumaker. „Solving Elliptic PDE's on Domains with Curved Boundaries with an Immersed Penalized Boundary Method“. *Journal of Scientific Computing* 80.3 (Sep. 2019), S. 1369–1394. DOI: 10.1007/s10915-019-00978-3.