Prof. Dr. Angela Kunoth
Samuel Leweke
Numerik partieller Differentialgleichungen II — SoSe 2020

Ausgabe: 29.04.2020

Übung 2

Abgabe: 06.05.2020, 12:00 Uhr

In diesem Projekt soll das Gitter zum Lösen einer PDE mit einem adaptiven Verfahren an die exakte Lösung angepasst werden. Um die Genauigkeit der numerischen Lösung einer PDE zu erhöhen, kann prinzipiell das Gitter gleichmäßig verfeinert werden. Auf diese Weise können Strukturen in der Lösung (große Änderungen in Bereichen des Rechengebiets) global aufgelöst werden. Dieses Vorgehen ist jedoch ineffizient, da auch Teilgebiete, in denen sich die Lösung kaum ändert, höher aufgelöst werden und die Gitterweite – und somit die Anzahl der Unbekannten – von den feinsten Strukturen der Lösung diktiert wird.

Ein effizienteres Vorgehen besteht also darin, nur dort die Auflösung des Gitters zu erhöhen, wo die Lösung besondere Strukturen aufweist. In den Teilbereichen, in denen die Lösung glatt ist, kann selbige auch mit einem groben Gitter ausreichend genau approximiert werden. Diese Anpassung des Gitters – und damit die angepasste Platzierung von Freiheitsgraden – wird von adaptiven Algorithmen realisiert.

Ein adaptives Verfahren besteht aus mehreren Subroutinen:

- `SOLVE`: Löse die diskretisierte PDE auf einem gegebenen Gitter.
- `ESTIMATE`: Schätze den lokalen Fehler der numerischen Lösung auf dem aktuellen Gitter. Dieser Schritt wird durch einen *a posteriori* Fehlerschätzer realisiert.
- `MARK`: Markiere die Gitterzellen, welche einen hohen Fehler aufweisen und verfeinert werden sollen.
- `REFINE`: Erzeuge aus der Markierung ein neues, lokal verfeinertes Gitter.

Beginnend mit einem groben (z. B. uniformen) Startgitter wird dieser Zyklus von Subroutinen so lange iteriert, bis eine vorgegebene Fehlerschranke unterschritten oder eine maximale Iterationszahl erreicht wird.

In diesem Projekt soll die eindimensionale PDE

$$\begin{aligned} -\Delta u &= f \quad && \text{in } \Omega = (0,1) \\ u &= 0 && \text{auf } \partial\Omega \end{aligned} \tag{1}$$

adaptiv mit B-Splines gelöst werden. Dazu wird die schwache Formulierung

$$\text{Finde } u \in H_0^1(\Omega): \qquad a(u,v) \coloneqq \int_\Omega \nabla u \cdot \nabla v \,\mathrm{d}x = \int_\Omega f v \,\mathrm{d}x =: L(v) \qquad \forall v \in H_0^1(\Omega) \tag{2}$$

verwendet. Der endlich-dimensionale Teilraum $V_h(\boldsymbol{\theta}) \subset H_0^1(\Omega)$ ist durch B-Splines $N_{i,k,\boldsymbol{\theta}}$ der Ordnung $k \geq 2$ mit nicht-uniformem Knotenvektor $\boldsymbol{\theta}$, der

$$\theta_1 = \ldots = \theta_k = 0 < \theta_{k+1} < \ldots < \theta_n < 1 = \theta_{n+1} = \ldots = \theta_{n+k} \tag{3}$$

erfüllt, gegeben. Um die Nullrandbedingung zu erfüllen, müssen die erste und letzte Basisfunktion entfernt werden:

$$V_h(\boldsymbol{\theta}) \coloneqq \operatorname{span}\{N_{i,k,\boldsymbol{\theta}} : i = 2, \ldots, \#\boldsymbol{\theta} - k - 1\} \subset H_0^1(\Omega). \tag{4}$$

Im Folgenden wird das Subskript k in $N_{i,k,\boldsymbol{\theta}}$ zur besseren Übersichtlichkeit weggelassen.

Die PDE wird nun zu gegebenem Gitter (repräsentiert durch den Knotenvektor $\boldsymbol{\theta}$) und zugehörigem Teilraum $V_h(\boldsymbol{\theta})$ durch Einschränken der schwachen Formulierung (2) auf $V_h(\boldsymbol{\theta})$ diskretisiert. Dies führt auf ein lineares Gleichungssystem

$$\boldsymbol{A}\boldsymbol{u}_{\boldsymbol{\theta}} = \boldsymbol{b} \tag{5}$$

mit

$$a_{ij} = \int_0^1 N'_{j,\boldsymbol{\theta}}(x) N'_{i,\boldsymbol{\theta}}(x)\,\mathrm{d}x, \qquad b_i = \int_0^1 f(x) N_{i,\boldsymbol{\theta}}(x)\,\mathrm{d}x. \tag{6}$$

Das Lösen von (5) realisiert `SOLVE`.

Als nächstes muss in `ESTIMATE` der Fehler von

$$u_{\boldsymbol{\theta}} = \sum_{i=2}^{\#\theta-k-1} (\boldsymbol{u}_{\boldsymbol{\theta}})_i\, N_{i,\boldsymbol{\theta}} \in V_h(\boldsymbol{\theta}) \tag{7}$$

geschätzt werden. Dazu wird das Residuum $r\colon \Omega \to \mathbb{R}$ definiert als $r = \Delta u_{\boldsymbol{\theta}} + f$. Es kann gezeigt werden, dass es für Splineräume $V_h(\boldsymbol{\theta}) \subset C^1(\Omega)$ eine vom Gitter $\boldsymbol{\theta}$ unabhängige konstante $C > 0$ gibt, sodass

$$\|u - u_{\boldsymbol{\theta}}\|_{H^1_0(\Omega)} \le C\varepsilon_{\boldsymbol{\theta}}(u_{\boldsymbol{\theta}}), \qquad \varepsilon^2_{\boldsymbol{\theta}}(u_{\boldsymbol{\theta}}) \coloneqq \sum_{i=1}^{\#\theta-1} (\theta_{i+1}-\theta_i)^2 \|r\|^2_{L_2([\theta_i,\theta_{i+1}])}, \tag{8}$$

wobei $u \in H^1_0(\Omega)$ die exakte Lösung von (1) ist. Die Größe

$$\varepsilon^2_{\boldsymbol{\theta},i}(u_{\boldsymbol{\theta}}) \coloneqq (\theta_{i+1}-\theta_i)^2 \|r\|^2_{L_2([\theta_i,\theta_{i+1}])}, \qquad i = 1,\dots,\#\boldsymbol{\theta}-1, \tag{9}$$

liefert somit einen Indikator für den Beitrag eines Knotenspanns $[\theta_i, \theta_{i+1}]$ zum Gesamtfehler.

Mit dem Fehlerindikator $\varepsilon_{\boldsymbol{\theta},i}(u_{\boldsymbol{\theta}})$ können diejenigen Teilintervalle $[\theta_i, \theta_{i+1}]$ verfeinert werden, die am meisten zum Fehler beitragen. Eine häufig gewählte Strategie für `MARK` ist das sogenannte *Dörfler-Marking*. Hierbei werden Teilintervalle $\mathcal{M} \subset \{1,\dots,\#\boldsymbol{\theta}-1\}$ mit den größten Beiträgen $\varepsilon_{\boldsymbol{\theta},i}(u_{\boldsymbol{\theta}})$ markiert bis

$$\sqrt{\sum_{i\in\mathcal{M}} \varepsilon^2_{\boldsymbol{\theta},i}(u_{\boldsymbol{\theta}})} \ge \varrho\, \varepsilon_{\boldsymbol{\theta}}(u_{\boldsymbol{\theta}}) \tag{10}$$

für ein gegebenes $\varrho \in (0,1]$ erfüllt ist.

Im letzten Schritt, `REFINE`, wird ein neues Gitter $\tilde{\boldsymbol{\theta}}$ aus $\boldsymbol{\theta}$ und den Markierungen $\mathcal{M}$ erzeugt. In diesem Projekt wird ein ausgewähltes Teilintervall $[\theta_i, \theta_{i+1}]$ mit $i \in \mathcal{M}$ einfach in zwei gleich große Stücke $[\theta_i, (\theta_i+\theta_{i+1})/2]$ und $[(\theta_i+\theta_{i+1})/2, \theta_{i+1}]$ zerteilt. Das heißt, zwischen θ_i und θ_{i+1} wird der Knoten $(\theta_i+\theta_{i+1})/2$ eingefügt.

Dieses Projekt wird in zwei Teilen in den Übungen 2 und 4 umgesetzt. Im ersten Teil werden `SOLVE` und `ESTIMATE` implementiert und getestet.

Aufgaben (20 Punkte)

a) Schreiben Sie eine Matlab-Funktion `v = EvalSpline(x, knots, order, coeffs, diff)`, welche die `diff`-te Ableitung (`diff` ≥ 0) des B-Splines

$$S_{\texttt{knots}}(x) = \sum_{i=1}^{\#\texttt{knots}-\texttt{order}} \texttt{coeffs}_i\, N^{(\texttt{diff})}_{i,\texttt{order},\texttt{knots}}(x) \tag{11}$$

auswertet. Die Auswertung erfolgt an allen Punkten des (beliebigen) Arrays `x` und der Rückgabewert `v` ist ein Array derselben Größe.

b) Schreiben Sie eine Matlab-Funktion `coeffs = Solve(knots, order, rhsF)`, welche zu einem gegebenen Gitter $\boldsymbol{\theta} =$ `knots` das zur PDE (1) gehörende lineare System (5) löst. Die rechte Seite f der PDE (1) wird als Function Handle in `rhsF` übergeben. Die Koeffizienten der diskreten Lösung $u_{\boldsymbol{\theta}}$ aus (7) werden als Vektor `coeffs` zurückgegeben.

c) Schreiben Sie eine Matlab-Funktion `err = Estimate(knots, order, coeffs, rhsF)`, die den Fehlerindikator (9) auf allen Teilintervallen $i = 1,\dots,\#\boldsymbol{\theta}-1$ auswertet und als Vektor `err` zurückliefert. Die Koeffizienten der diskreten Lösung $u_{\boldsymbol{\theta}}$ werden in `coeffs` und die rechte Seite f der PDE (1) wird als Function Handle in `rhsF` übergeben.

d) Schreiben Sie ein Matlab-Programm, welches die PDE (1) mit rechter Seite

$$f(x) = x^{-1/4} \tag{12}$$

und analytischer Lösung

$$u(x) = \frac{16}{21} x \left(1 - x^{3/4}\right), \qquad u'(x) = -\frac{4}{21}\left(7x^{3/4} - 4\right) \tag{13}$$

numerisch löst. Das Problem soll dabei mit uniformen Gittern der Gitterweite $h = 2^{-J}$ mit Level $J = 3, \dots, 6$ diskretisiert werden.

Berechnen Sie für jedes Level den a posteriori Fehlerschätzer $\varepsilon_{\boldsymbol{\theta}}(u_{\boldsymbol{\theta}})$ und den Fehler $\|u - u_{\boldsymbol{\theta}}\|_{H_0^1(\Omega)}$ in der $H_0^1(\Omega)$-Norm und tragen Sie beide in einem gemeinsamen doppelt-logarithmischen Plot gegen die Gitterweite $h = 2^{-J}$ auf.

Plotten Sie außerdem für das feinste Gitter den a posteriori Fehlerschätzer $\varepsilon_{\boldsymbol{\theta},i}(u_{\boldsymbol{\theta}})$ auf jedem Knotenspann $[\theta_i, \theta_{i+1}]$ für $i = 1, \dots, \#\boldsymbol{\theta} - 1$ gegen die jeweilige linke Grenze θ_i des Knotenspanns. Hierbei soll die y-Achse mit $\varepsilon_{\boldsymbol{\theta},i}(u_{\boldsymbol{\theta}})$ logarithmisch skaliert sein.

Führen Sie die obige Prozedur für quadratische ($k = \texttt{order} = 3$) und kubische Splines ($k = 4$) durch.

Hinweise

Aus dem vorherigen Semester werden auf der Homepage / im sciebo-Ordner die geschützten Implementierungen der folgenden Funktionen bereitgestellt:

- `I = gaussInt(f, a, b, nodes)`
  Berechnet das Integral über $[a, b]$ der Funktion $f\colon [a, b] \to \mathbb{R}$ mittels Gauß-Legendre Quadratur mit `nodes` Knoten. Polynome bis zum Grad $2 \cdot \texttt{nodes} - 1$ werden exakt integriert. Die Funktion f wird als Function Handle `f` übergeben.
- `v = Neville(x, coeffs, knots, order)`
  Wertet einen B-Spline, der durch die Ordnung $k = \texttt{order}$, den erweiterten Knotenvektor `knots` und die Koeffizienten `coeffs` gegeben ist, an den Stellen `x` aus. Das Ergebnis $\texttt{v} = \sum_i \texttt{coeffs}_i N_{i,k}(\texttt{x})$ ist ein Array mit dem Format von `x`.
- `v = Nik(x, i, knots, order, diff)`
  Wertet die `diff`-te Ableitung der `i`-ten B-Spline Basisfunktion an den Stellen `x` aus. Die Basisfunktionen sind dabei durch die Ordnung `order` und den erweiterten Knotenvektor `knots` festgelegt. Das Ergebnis ist ein Array mit dem Format von `x`.
- `A = MatAssembly(knots, order, diff, startI, endI)`
  Berechnet für eine B-Spline Basis identifiziert durch erweiterten Knotenvektor `knots` und Ordnung $k = \texttt{order}$ die $(\texttt{endI} - \texttt{startI} + 1) \times (\texttt{endI} - \texttt{startI} + 1)$ Matrix $\boldsymbol{A}$ mit

  $$a_{i-\texttt{startI}+1, j-\texttt{startI}+1} = \int_{\texttt{knots}_{\texttt{startI}}}^{\texttt{knots}_{\texttt{endI}+k}} N_{j,k}^{(\texttt{diff})}(x)\, N_{i,k}^{(\texttt{diff})}(x)\, \mathrm{d}x \qquad i, j = \texttt{startI}, \dots, \texttt{endI}.$$

  Die Indizes `startI` und `endI` geben den Index der ersten bzw. letzten Basisfunktion an. Die Quadratur wird intern automatisch so gewählt, dass die Integrale exakt berechnet werden.
- `b = VecAssembly(g, knots, order, startI, endI, gaussLegNodes)`
  Berechnet für eine B-Spline Basis identifiziert durch erweiterten Knotenvektor `knots` und Ordnung $k = \texttt{order}$ den Vektor $\boldsymbol{b}$ der Länge $(\texttt{endI} - \texttt{startI} + 1)$ mit

  $$b_{i-\texttt{startI}+1} = \int_{\texttt{knots}_{\texttt{startI}}}^{\texttt{knots}_{\texttt{endI}+k}} g(x)\, N_{i,k}(x)\, \mathrm{d}x \qquad i = \texttt{startI}, \dots, \texttt{endI}.$$

  Die Indizes `startI` und `endI` geben den Index der ersten bzw. letzten Basisfunktion an. Die Funktion g wird als Function Handle `g` und die Anzahl der Quadraturknoten wird in `gaussLegNodes` übergeben.

Literaturverzeichnis

[BG16] A. Buffa und C. Giannelli. Adaptive isogeometric methods with hierarchical splines: Error estimator and convergence. *Mathematical Models and Methods in Applied Sciences*, 26(01):1–25, 2016. DOI: 10.1142/S0218202516500019.