Unstetige Galerkin-Verfahren und die lineare Transportgleichung

Tobias G. Pfeiffer*

Freie Universität Berlin

26. Mai 2009

Zusammenfassung

Es soll eine Einführung in die Behandlung der linearen Transportgleichung $u_t + \nabla \cdot (\vec{a}\, u) = 0$ mittels unstetiger Galerkin-Verfahren (DG-Verfahren) gegeben werden. Nach der Entwicklung einer schwachen Formulierung des Problems und der Vorstellung verschiedener numerischer Flüsse soll der Fokus besonders auf Fragen der konkreten Implementierung liegen, wobei der Leser befähigt werden soll, das DG-Verfahren im Zweidimensionalen selbst zu implementieren. Des Weiteren wird ein Ausblick auf Techniken zur Beschleunigung und Parallelisierung des Verfahrens gegeben.

Inhaltsverzeichnis

* eMail: tobias.pfeiffer@fu-berlin.de

1 Die lineare Transportgleichung

Die lineare Transportgleichung beschreibt die zeitliche Entwicklung einer skalaren Größe u, die auf einem gewissen Gebiet $\Omega \subset \mathbb{R}^d$ definiert ist:

$$u_t + \nabla \cdot (\vec{a}\, u) = 0 \qquad \text{auf } [0,T] \times \Omega \tag{1.1}$$

$$u(0,x) = u_0(x) \qquad \text{auf } \Omega \tag{1.2}$$

Sie ist ein Spezialfall der allgemeinen Transportgleichung $u_t + \nabla \cdot f(t,x,u,\nabla u) = g(t,x,u)$ und beschreibt die Bewegung von u in Richtung bzw. mit Geschwindigkeit $\vec{a}$, wobei $\vec{a}$ auch von t und x abhängen kann. Die Transportgleichung beschreibt in ihrer allgemeinen Form verschiedene Transportphänomene, z. B. der Wärmeleitung oder der Fluiddynamik.

Ziel der numerischen Behandlung der Transportgleichung ist die Berechnung von u im Laufe der Zeit bzw. zu einem gegebenen Zeitpunkt T. Dafür muss insbesondere u_t aus den gegebenen Anfangsdaten $u_0(x)$ bestimmt und dann $u(T,x) = \int_0^T u_t(t,x)\,\mathrm{d}t$ mittels eines geeigneten Zeitintegrationsverfahrens (z. B. Euler, Runge-Kutta) berechnet werden.

2 Entwicklung einer schwachen Formulierung

2.1 Elementweise Ansatzräume

Seien $\{\Omega_i\}_{i=1,\dots,N}$ paarweise disjunkte Teilmengen des $\mathbb{R}^d$, so dass $\Omega = \bigcup_{i=1}^N \overline{\Omega_i}$ (d. h. anschaulich sind die $\{\Omega_i\}$ eine Partition, bis auf die Ränder). Mit einem geeigneten Funktionenraum $V \subset L^2(\Omega)$ kann man (1.1) durch Multiplikation mit $v \in V$ und Integration über Ω in eine schwache Formulierung bringen:

$$u_h \in V: \quad \int_\Omega (u_h)_t\, v + \int_\Omega \nabla \cdot (\vec{a}\, u_h)\, v = 0 \quad \forall\, v \in V \tag{2.1}$$

Die Idee der unstetigen Galerkin-Verfahren, die erstmals in [RH73] beschrieben wurde, ist es, V so zu wählen, dass an den Elementgrenzen Unstetigkeiten zugelassen sind, d. h. insbesondere kann prinzipiell auf jedem Ω_i ein anderer Ansatzraum $V(\Omega_i)$ mit $\forall\, v \in V(\Omega_i) : \operatorname{supp}(v) \subseteq \Omega_i$ gewählt werden. (Sinnvollerweise wählt man dabei einen Raum von Polynomen bis zu einem gewissen Grad m_i, dazu später mehr.) Dann kann man (2.1) auf jedes einzelne Element projizieren:

$$u_h \in V(\Omega_i): \quad \int_{\Omega_i} (u_h)_t\, v + \int_{\Omega_i} \nabla \cdot (\vec{a}\, u_h)\, v = 0 \quad \forall\, v \in V(\Omega_i) \tag{2.2}$$

Lemma 1. *Falls $\{v_{|\bigcup_i \Omega_i} \mid v \in V\}$ isomorph zum kartesischen Produkt $V(\Omega_1) \times \ldots \times V(\Omega_N)$ ist und jedes $V(\Omega_i)$ die konstanten Funktionen enthält, dann gilt:*

$$(2.1) \iff \forall\, i \in \{1,\dots,N\} : (2.2)$$

Beweis. „⇐“ ist klar, denn wenn alle Summanden 0 sind, ist auch die Summe 0.

Für „⇒“ sei angenommen, dass (2.2) für ein festes i nicht gelte. Wähle als Testfunktion für (2.1) ein v mit $v \equiv 1$ auf Ω_i und 0 sonst, dann ist $\int_\Omega (u_h)_t\, v + \nabla \cdot (\vec{a}\, u_h)\, v = \int_{\Omega_i} (u_h)_t\, v + \nabla \cdot (\vec{a}\, u_h)\, v \neq 0$ nach Annahme, d. h. (2.1) gilt nicht. □

2.2 Numerische Flüsse

In (2.2) ist jedes Element vollständig unabhängig von allen anderen. Da dies mit der physikalischen Anschauung nicht übereinstimmt (der Zu- und Abfluss in ein Gebiet aus seiner Umgebung bestimmt ganz wesentlich das Verhalten über die Zeit), muss für ein sinnvolles Ergebnis eine gewisse Kopplung mit anderen Elementen vorhanden sein. Um diese wieder in das Problem einfließen zu lassen, wird (2.2) partiell integriert und man erhält:

$$\int_{\Omega_i} (u_h)_t v - \int_{\Omega_i} (\vec{a}\, u_h) \cdot \nabla v + \int_{\partial\Omega_i} \widehat{\vec{a}\, u_h} \cdot \vec{n}\, v = 0 \tag{2.3}$$

Dabei ist $\widehat{\vec{a}\, u_h}$ ein Term, der nicht kanonisch gegeben ist, denn auf den Rändern der Elemente ist u_h nicht wohldefiniert. Die Wahl von $\widehat{\vec{a}\, u_h} : \bigcup_{i=1}^{N} \partial\Omega_i \to \mathbb{R}$, dem sog. *numerischen Fluss* (anschaulich ist $\widehat{\vec{a}\, u_h} \cdot \vec{n}$ tatsächlich ein Maß dafür, „wieviel über die Kante abfließt“) ist dabei essentiell für Konsistenz und Stabilität des Verfahrens. Es gibt hier zunächst zwei naheliegende Möglichkeiten, den *Upwind-Fluss* und den *Lax-Friedrichs-Fluss*.

Der Upwind-Fluss

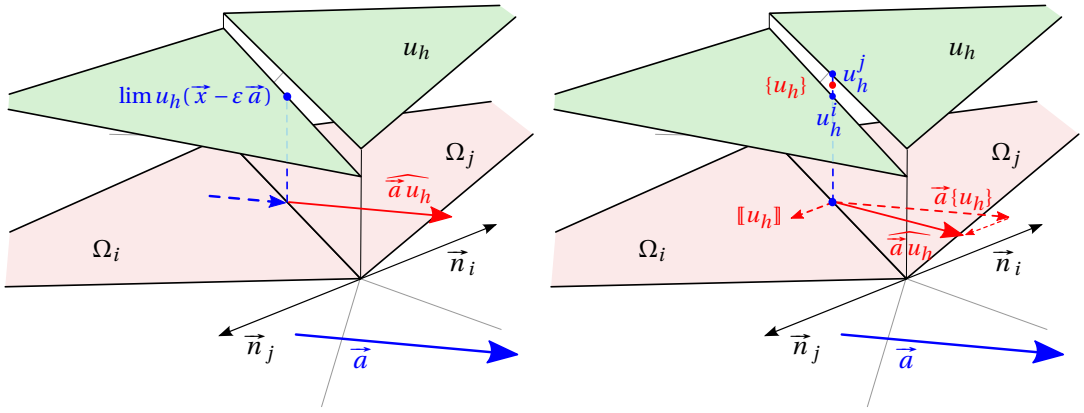
$$\widehat{\vec{a}\, u_h}(x) = \vec{a} \lim_{\varepsilon \searrow 0} u_h(\vec{x} - \varepsilon \vec{a}) \tag{2.4}$$

wird im Wesentlichen vom Grenzwert von u_h, wenn man sich aus der Richtung $-\vec{a}$ an den Rand eines Elementes annähert, bestimmt, und transportiert in Richtung $\vec{a}$; vgl. Abb. 1(a).

Zur Definition des Lax-Friedrichs-Flusses bedient man sich zweier zusätzlicher Definitionen. Sei x auf der Kante $e := \partial\Omega_i \cap \partial\Omega_j$, $\vec{n_i}$ bzw. $\vec{n_j}$ die Einheitsaußennormalen von Ω_i bzw. Ω_j auf e und u_h^i bzw u_h^j die Grenzwerte von u_h auf e aus Richtung des jeweiligen Elements,[1] dann definiert man den *Durchschnitt* $\{u_h\} = \frac{1}{2}(u_h^i + u_h^j)$ und den *Sprung* $[\![u_h]\!] = u_h^i \vec{n_i} + u_h^j \vec{n_j}$. Der Lax-Friedrichs-Fluss

$$\widehat{\vec{a}\, u_h} = \vec{a}\{u_h\} + \tfrac{1}{2} \left| \vec{a} \right| [\![u_h]\!] \tag{2.5}$$

besteht also anschaulich aus einer Komponente, die, ähnlich zum Upwind-Fluss, in $\vec{a}$-Richtung transportiert, aber zusätzlich noch in Normalenrichtung vom Element mit größerem Funktionswert zu dem mit kleinerem transportiert,[2] siehe Abb. 1(b).



(a) Upwind-Fluss (b) Lax-Friedrichs-Fluss

Abbildung 1: Numerische Flüsse über Elementgrenzen

Befindet man sich am Rand von Ω, muss man sich zur Berechnung des Flusses auf einen Wert von u_h außerhalb von Ω festlegen. Diese Wahl wird dabei maßgeblich durch die physikalische Situation, die die Berechnung simuliert, beeinflusst; so kann man z. B. „Wände“ um das Rechengebiet herum oder einen Zu- bzw. Abfluss simulieren.

[1] Häufig begegnet man hier auch der Schreibweise u_h^+ und u_h^- für die Beschreibung des Grenzwertes aus den beiden Gebieten.
[2] Im eindimensionalen Fall und falls $\vec{a} = a > 0$, sind die beiden Flüsse sogar identisch.

2.3 Stabilität

In [Coc03] wird gezeigt, dass das Problem (1.1) stabil ist, falls $-\nabla \cdot \vec{a} \leq L$ für eine Konstante L. Des Weiteren wird gezeigt, dass auch das DG-Verfahren selbst stabil ist, falls

$$\Theta_h := \sum_{i=1}^{N} \left(-\frac{1}{2} \int_{\Omega_i} \nabla \cdot (\vec{a}\, u_h) + \int_{\partial\Omega_i} \widehat{\vec{a}\, u_h} \cdot \vec{n}\, u_h \right) \geq 0$$

ist und dass dies der Fall ist, wenn der numerische Fluss die Gestalt

$$\widehat{\vec{a}\, u_h} = \vec{a}\{u_h\} + C[\![u_h]\!] \tag{2.6}$$

hat, wobei C eine nicht-negativ definite Matrix (oder insbesondere ein Skalar) ist. Da die beiden oben vorgestellten Flüsse von dieser Form sind, wobei $C = \frac{1}{2}\left|\vec{a} \cdot \vec{n}\right|$ (Upwind) bzw. $C = \frac{1}{2}\left|\vec{a}\right|$ (Lax-Friedrichs), folgt sofort die Stabilität. Dass der Wert von C für den Upwind-Fluss korrekt ist, zeigt sich allerdings erst bei Betrachtung von $\widehat{\vec{a}\, u_h} \cdot \vec{n}$.[3]

3 Lösen der schwachen Formulierung

3.1 Formulierung als lineares Gleichungssystem

Soll (2.3), also

$$\int_{\Omega_i} (u_h)_t\, v - \int_{\Omega_i} (\vec{a}\, u_h) \cdot \nabla v + \int_{\partial\Omega_i} \widehat{\vec{a}\, u_h} \cdot \vec{n}\, v = 0$$

nun numerisch gelöst werden, muss man zunächst auf Ω_i eine Basis $\{\varphi_1^i, \ldots, \varphi_{k_i}^i\}$ von $V(\Omega_i)$ gewählt werden. (Im Folgenden sollen die i-Indizes bei Funktionen und Koeffizienten der besseren Lesbarkeit wegen weggelassen werden.) Ziel ist die Bestimmung von $(u_h)_t$ als Linearkombination dieser Basisfunktionen, d. h. die Berechnung der Koeffizienten u_t^j, so dass $(u_h)_t = \sum_{j=1}^{k_i} u_t^j \varphi_j$. Die Bedingung „$\forall\, v \in V(\Omega_i)$“ kann man äquivalent durch die Bedingung „$\forall\, \varphi_l, l = 1, \ldots, k_i$“ ersetzen und erhält dann zunächst:

$$\vec{u_t} \in \mathbb{R}^{k_i}: \quad \int_{\Omega_i} \left(\sum_{j=1}^{k_i} u_t^j \varphi_j \right) \varphi_l - \int_{\Omega_i} \left(\vec{a}\, u_h\right) \cdot \nabla \varphi_l + \int_{\partial\Omega_i} \widehat{\vec{a}\, u_h} \cdot \vec{n}\, \varphi_l = 0 \quad \forall\, l = 1, \ldots k_i \tag{3.1}$$

u_h selbst steht in (3.1) noch nicht als Linearkombination einer Basis von $V(\Omega_i)$. Prinzipiell ist es hier auch möglich, u_h selbst ganz anders darzustellen als $(u_h)_t$ (z. B. in einer anderen Basis $\{\psi_j\}$ eines anderen Funktionenraumes $\tilde{V}(\Omega_i)$). Dafür kann es durchaus ein Bedürfnis geben, z. B. wenn es sich um die Anfangsdaten u_0 handelt, diese höher aufgelöst sind als $V(\Omega_i)$ erlaubt, und man diese Genauigkeit ausnutzen will. Andererseits bringt dies aber viele Schwierigkeiten bei der Zeitintegration mit sich; insbesondere kann es passieren, dass

1. $u_h^T := u_h + \int_0^T (u_h)_t$ (wie auch immer dies numerisch berechnet wird) gar nicht mehr in $\tilde{V}(\Omega_i)$ liegt, wenn nicht $V(\Omega_i) \subset \tilde{V}(\Omega_i)$, und dann im nächsten Zeitschritt das Verfahren verändert werden muss und
2. selbst falls doch $V(\Omega_i) \subset \tilde{V}(\Omega_i)$ man bei der Berechnung von u_h^T nicht einfach die Basiskoeffizienten von $u_h \in \tilde{V}(\Omega_i)$ und $\left(\int_0^T (u_h)_t\right) \in V(\Omega_i)$ addieren kann, sondern vorher noch eine Basistransformation durchführen müsste.

Es soll daher im Folgenden davon ausgegangen werden, dass u_h ebenfalls in der Basis $\{\varphi_j\}$ dargestellt werden kann (und ggf. u_0 vorher nach $V(\Omega_i)$ projiziert wurde), so dass es eine Darstellung $u_h = \sum_{j=1}^{k_i} u^j \varphi_j$ gibt. Dann lässt sich (3.1) wie folgt umschreiben:

$$\sum_{j=1}^{k_i} u_t^j \int_{\Omega_i} \varphi_j \varphi_l - \sum_{j=1}^{k_i} u^j \int_{\Omega_i} \left(\vec{a}\, \varphi_j\right) \cdot \nabla \varphi_l + \int_{\partial\Omega_i} \widehat{\vec{a}\, u_h} \cdot \vec{n}\, \varphi_l = 0 \tag{3.2}$$

[3] Einige Autoren arbeiten auch mit Flüssen, die keine vektoriellen Größen sind, sondern schon das Skalarprodukt mit $\vec{n}$ enthalten. [Li06] bspw. ersetzt den Term $\vec{F}(u) \cdot \vec{n}$ durch einen (skalaren) Fluss der Form $F_{\vec{n}}(u^-, u^+)$.

In Matrixform kann man dies als

$$M^i \overrightarrow{u_t} + K^i \overrightarrow{u} + F^i = 0 \tag{3.3}$$

schreiben, wobei dann $M^i_{lj} = \int_{\Omega_i} \varphi_j \varphi_l$ die Massenmatrix ist, $K^i_{lj} = -\int_{\Omega_i} \left(\vec{a}\varphi_j\right) \cdot \nabla\varphi_l$ die Steifigkeitsmatrix (nach [Li06, S. 30]) und $F^i_l = \int_{\partial\Omega_i} \widehat{\vec{a}\,u_h} \cdot \vec{n}\varphi_l$ die Rand- bzw. Flussterme darstellt. Dabei gilt:

- M^i ist nur von $V(\Omega_i)$ abhängig, muss also nur neu berechnet werden, falls sich der Funktionenraum ändert (z. B. p-Adaptivität) oder das Gebiet selbst (z. B. h-Adaptivität).
- K^i hängt zusätzlich von $\vec{a}$ ab. Falls $\vec{a}$ jedoch nicht von t abhängig ist, muss K^i ebenfalls nur bei Änderung von $V(\Omega_i)$ neu berechnet werden.
- F^i hängt in dieser allgemeinen Form von $V(\Omega_i)$ und dem numerischen Fluss ab, d. h. im Allgemeinen von $\vec{a}$ und u_h zum Zeitpunkt t auf Ω_i *und den benachbarten Elementen von* Ω_i. In der lokalen Form (3.3) muss daher F^i in jedem Zeitschritt neu assembliert werden und die Funktionswerte der Nachbarelemente kennen. Später soll aufgezeigt werden, wie man dieses Problem vermeidet.

Den gesuchten Vektor $\overrightarrow{u_t}$ erhält man also durch $\overrightarrow{u_t} = \left(M^i\right)^{-1} \cdot \left(-K^i \overrightarrow{u} - F^i\right)$.

3.2 Zeitintegration

Ist $u_h(t,x)$ zu einem festen Zeitpunkt t_0 bekannt, kann man also $\frac{d}{dt} u_h(t,x)_{|t=t_0}$ wie oben berechnen. Das Ziel von (1.1) ist die Bestimmung von u in einem gewissen Zeitintervall $[0,T]$, d. h. die Informationen über u_h und $(u_h)_t$ zum Zeitpunkt t_0 müssen nun genutzt werden, um u_h zum Zeitpunkt t_1 zu berechnen. Gemäß Hauptsatz der Differential- und Integralrechnung ist $u_h(t_1,x)$ gegeben durch:

$$u_h(t_1,x) = u_h(t_0,x) + \int_{t_0}^{t_1} (u_h)_t(t,x)\,\mathrm{d}t$$

Dies ist ein klassisches Anfangswertproblem, für das viele Standardverfahren existieren, z. B. Euler- oder Runge-Kutta-Verfahren. Mit $H(\vec{u},t) := \left(M^i\right)^{-1} \cdot \left(-K^i(t) \cdot \vec{u} - F^i(t,\vec{u})\right)$:

- Explizites Euler-Verfahren: $\overrightarrow{u_{|t_1}} = \overrightarrow{u_{|t_0}} + \Delta t \overrightarrow{(u_h)_{|t_0}} = \overrightarrow{u_{|t_0}} + \Delta t \cdot H(\overrightarrow{u_{|t_0}}, t_0)$
- Explizites Runge-Kutta-Verfahren der Ordnung 2:

$$\vec{f_1} = \Delta t H(\overrightarrow{u_{|t_0}}, t_0) \qquad \vec{f_2} = \Delta t H(\overrightarrow{u_{|t_0}} + f_1, t_0 + \Delta t) \qquad \overrightarrow{u_{|t_1}} = \overrightarrow{u_{|t_0}} + \tfrac{1}{2}\left(\vec{f_1} + \vec{f_2}\right)$$

Anschaulich wird also für jeden Basiskoeffizienten eine gewöhnliche Differentialgleichung gelöst. Es sei angemerkt, dass es prinzipiell für die Zeitintegration ebenfalls möglich ist, DG-Verfahren einzusetzen; es gibt z. B. auch DG-ODE-Löser, die äquivalent zu bestimmten Runge-Kutta-Verfahren sind.

4 Implementierung des DG-Verfahrens

Das bisher vorgestellte Verfahren macht noch keine Annahmen über

1. Form und Gestalt der Ω_i (bisher kann jedes Ω_i ein beliebiges zusammenhängendes Gebiet mit Lipschitz-Rand sein) und Dimension des umgebenden Raumes
2. Aufbau der Funktionenräume $V(\Omega_i)$.

Dies soll in diesem Abschnitt geschehen und dabei Vor- und Nachteile bei der Wahl für die Implementierung dargestellt werden.

4.1 Wahl der Elemente

Für $\Omega \subset \mathbb{R}$ vereinfachen sich viele Notationen und Konzepte, so ist z. B. die Behandlung von „Randintegralen“ und „Normalen“ dort kein erwähnenswertes Problem. Es soll im Folgenden jedoch der Fokus gerade deswegen auf $\Omega \subset \mathbb{R}^2$ liegen, um ein umfassendes Verständnis der notwendigen Schritte zu erzeugen. Des Weiteren kann man erst dort viele der Vorteile des DG-Verfahrens im Vergleich zur klassischen Finite-Elemente-Methode deutlich erkennen, wie z. B. die einfachere Behandlung von nichtkonformen Gittern oder unterschiedlichen Ansatzräumen auf benachbarten Elementen.

Bisher wurde an die Gestalt der Ω_i keine Einschränkungen gemacht. Wie in den meisten numerischen Verfahren bringt jedoch die Wahl von polygonal berandeten Gebieten deutliche Vereinfachungen in der Implementierung, so sind z. B. die Normalen stückweise konstant und es ist einfacher, Basen für polynomielle Ansatzräume anzugeben. Die nächsten Abschnitte sollen sich daher ebenfalls nur mit polygonal berandeten Elementen beschäftigen und es soll sogar die Einschränkung auf reine Dreiecksnetze gemacht werden, da dies die Behandlung verschiedener Ansatzfunktionen vereinfacht.

4.2 Wahl des Ansatzraumes

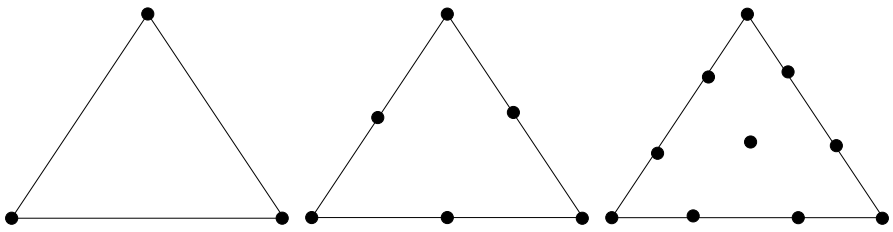
In diesem Abschnitt sollen verschiedene Möglichkeiten zur Wahl der Ansatzräume $V(\Omega_i)$ vorgestellt werden. Für solche Räume sind im Wesentlichen die Räume $P^m(\Omega_i)$ der Polynome vom Grad $\leq m$ interessant.

Die einfachste Wahl, die man dort treffen kann, sind die auf Ω_i konstanten Funktionen. Die Formulierung (3.2) vereinfacht sich dann zu:

$$u_t \cdot \operatorname{area}(\Omega_i) + \int_{\partial\Omega_i} \widehat{\vec{a}\, u_h} \cdot \vec{n} = 0 \qquad \Longleftrightarrow \qquad u_t = -\frac{1}{\operatorname{area}(\Omega_i)} \int_{\partial\Omega_i} \widehat{\vec{a}\, u_h} \cdot \vec{n}$$

Dies ist bei Wahl eines geeigneten Flusses gerade das Verfahren der Finiten Volumen und der eine Basiskoeffizient, den es dann pro Element gibt, repräsentiert das Volumen in der entsprechenden Zelle.

Knotenbasis Für Polynomräume P^m mit $m \geq 1$ kommt es zunächst darauf an, eine geeignete Basis zu wählen. Zunächst bietet sich die Wahl der *Knotenbasis* an, also solcher Funktionen, die zu einer Menge von Knotenpunkten $\{p_1, \ldots, p_k\}$ in Ω_i die Eigenschaft $\varphi_j(p_l) = \delta_{jl}$ haben. Dabei sollte die Menge der Knotenpunkte jedoch so beschaffen sein, dass dies auch möglich ist. Nach [KS07, Lemma 6.4] hat die Menge der $k = \frac{(m+1)(m+2)}{2}$ Punkte, auf $m+1$ parallelen Linien im Dreieck verteilt (wie in Abb. 2 gezeigt), die Eigenschaft, eine eindeutige Knotenbasis von P^m zu besitzen.



(a) $m = 1$ (b) $m = 2$ (c) $m = 3$

Abbildung 2: Knotenpunkte für eindeutige Knotenbasis

Zur konkreten Darstellung der Basisfunktionen erweist es sich als praktisch, jeden Punkt x im Dreieck durch seine *normalisierten Koordinaten* $\alpha_1, \alpha_2, \alpha_3$ (mit $\alpha_1 + \alpha_2 + \alpha_3 = 1$) in Bezug zu den drei Eckpunkten v_1, v_2, v_3 darzustellen, d. h. $x = \alpha_1 v_1 + \alpha_2 v_2 + \alpha_3 v_3$. Die Knotenbasis zu P^1 hat dann die einfache Form:

$$\varphi_j(\alpha_1, \alpha_2, \alpha_3) = \alpha_j \text{ für } j = 1, 2, 3$$

Basen zu Polynomräumen höheren Grades kann man durch eine Rekursionsformel erhalten, siehe [Li06]:

$$\varphi_\tau(\alpha_1, \alpha_2, \alpha_3) = \psi_I^I(\alpha_1)\psi_J^J(\alpha_2)\psi_K^K(\alpha_3) \qquad \text{für alle Tripel } \tau = (I, J, K) \text{ mit } I + J + K = m \tag{4.1}$$

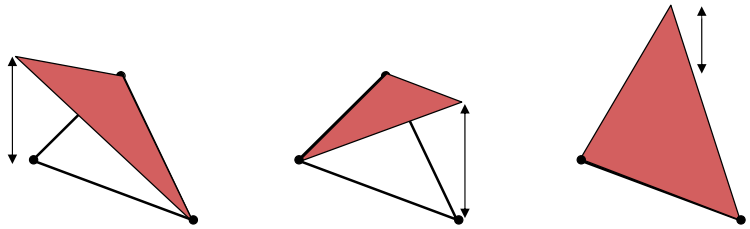


Abbildung 3: Lineare Basisfunktionen

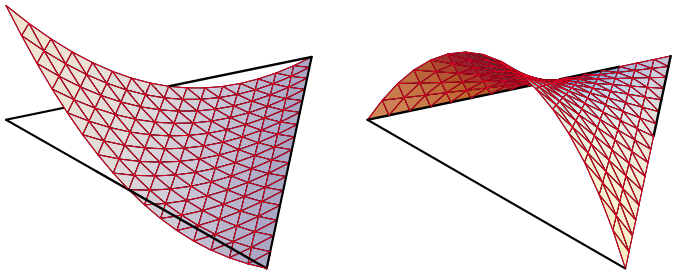
Dabei sind die $\psi_l^n(\alpha)$ die eindimensionalen Lagrange-Polynome:

$$\psi_l^n(\alpha) = \prod_{i=0, i\neq l}^{n} \frac{\alpha - \beta_i}{\beta_l - \beta_i},$$

wobei die β_i (mit $i = 0, \ldots, m$) $m+1$ äquidistant verteilte Stützstellen im Intervall $[0,1]$ sind.

Beispiel: Sei $m = 2$ (damit ist $\beta_0 = 0, \beta_1 = \frac{1}{2}, \beta_2 = 1$) und zu bestimmen ist die Basisfunktion, die auf dem Knotenpunkt $\frac{1}{2}v_1 + \frac{1}{2}v_2$ (also dem Kantenmittelpunkt zwischen v_1 und v_2) liegt. Dies entspricht dem Tupel $\tau = (1,1,0)$ und es folgt nach (4.1):

$$\varphi_{(1,1,0)}(\alpha_1, \alpha_2, \alpha_3) = \psi_1^1(\alpha_1) \cdot \psi_1^1(\alpha_2) = \frac{\alpha_1 - \beta_0}{\beta_1 - \beta_0} \cdot \frac{\alpha_2 - \beta_0}{\beta_1 - \beta_0} = \frac{\alpha_1 - 0}{\frac{1}{2} - 0} \cdot \frac{\alpha_2 - 0}{\frac{1}{2} - 0} = 4\alpha_1\alpha_2$$



(Quelle: Vorlesungsskript *Introduction to Finite Element Methods*, University of Colorado at Boulder)

Abbildung 4: Quadratische Basisfunktionen (Auswahl)

Orthogonalbasis Da in jedem Zeitschritt die Massenmatrix $(M_i)^{-1}$ mit $M_{lj}^i = \int_{\Omega_i} \varphi_j \varphi_l$ zur Lösung von (3.3) benötigt wird, ist es wünschenswert, eine bezüglich des L^2-Skalarproduktes orthogonale Basis von P^k zu kennen, denn dann ist M^i diagonal. Im linearen Fall erhält man sehr einfach eine orthogonale Basis durch diejenigen Funktionen, die auf dem Mittelpunkt einer Kante den Wert 1 haben, auf den Mittelpunkten der beiden anderen Kanten den Wert 0, d. h. $\varphi_j(\alpha_1, \alpha_2, \alpha_3) = 1 - 2\alpha_j$.

4.3 Integration

Für das Assemblieren der Matrizen M^i und K^i und des Vektors F^i sind Integrale über Ω_i bzw. über $\partial\Omega_i$ auszuwerten:

- $M_{lj}^i = \int_{\Omega_i} \varphi_j \varphi_l$
- $K_{lj}^i = -\int_{\Omega_i} \left(\vec{a}\varphi_j\right) \cdot \nabla\varphi_l = -\int_{\Omega_i} \left(\vec{a} \cdot \nabla\varphi_l\right)\varphi_j$
- $F_l^i = \int_{\partial\Omega_i} \widehat{\vec{a}\, u_h} \cdot \vec{n}\varphi_l = \sum_{e\in\partial\Omega_i} \int_e \widehat{\vec{a}\, u_h} \cdot \vec{n_e}\varphi_l$

Für die Standard-Massenmatrix ist es prinzipiell möglich, alle auftretenden Integrale analytisch zu berechnen. Existiert eine Darstellung in normalisierten Koordinaten, kann die Formel

$$\int_{\Omega_i} \alpha_1^a \alpha_2^b \alpha_3^c = \frac{a!\,b!\,c!}{(a+b+c+2)!} 2\,\mathrm{area}(\Omega_i)$$

aus [ZTZ05] dabei hilfreich sein. Falls $\vec{a}$ nicht konstant ist, ist diese Methode für die Berechnung von K^i und F^i jedoch im Allgemeinen nicht mehr möglich. Deswegen, und weil es oft leichter zu implementieren ist, verwendet man stattdessen oft Quadraturformeln zur approximativen Bestimmung der Integrale.

Quadraturformeln für Funktionen der Form $f : [-1,1] \to \mathbb{R}$, wie man sie zur Berechnung der Kantenintegrale benötigt, sind gut bekannt und haben die Form

$$\int_{-1}^{1} f(s)\,\mathrm{d}s \approx \sum_{i=1}^{n} w_i f(a_i).$$

Um die Voraussetzungen für gewisse Konvergenzaussagen (vgl. [Coc97, Proposition 3.1]) zu erfüllen, müssen bei Verwendung von Basisfunktionen aus P^m die Kantenintegrale durch eine für alle Polynome aus P^{2m+1} exakten Quadraturformel approximiert werden. Werte für w_i und a_i für eine Approximation einer bestimmten Güte lassen sich leicht in der Literatur finden; im Folgenden die Werte aus [Coc97], die für die Verwendung von P^1- bzw. P^2-Funktionen ausreichend sind:

Stützstellen a_i	entsprechende Gewichte w_i	exakte Integration von
$\pm\frac{1}{\sqrt{3}}$	1	P^3
$\pm\frac{3}{5}, 0$	$\frac{5}{9}, \frac{8}{9}$	P^5

Für das Flächenintegral über einem Dreieck T haben Quadraturformeln die Form

$$\int_T f(x)\,\mathrm{d}x \approx \sum_{i=1}^{n} w_i f\Big(\sum_{j=1}^{3} \tau_{ij} \overrightarrow{v_j}\Big),$$

wobei $\overrightarrow{v_j}$ die Eckpunkte des Dreiecks sind, $\overrightarrow{\tau_{ij}}$ die normalisierten Koordinaten des Auswertungspunktes und w_i wieder Gewichte. Hier sind bei Verwendung von P^m Quadraturformeln notwendig, die Polynome vom Grad $2m$ noch exakt berechnen. Die entsprechenden Daten dazu finden sich in [Li06]:

normalisierte Koordinaten τ_i	entsprechende Gewichte w_i	exakte Integration von
$\left(0,\frac{1}{2},\frac{1}{2}\right), \left(\frac{1}{2},0,\frac{1}{2}\right), \left(\frac{1}{2},\frac{1}{2},0\right)$	$\frac{1}{3}$	P^2
$\left(\frac{1}{3},\frac{1}{3},\frac{1}{3}\right)$,	0,225	P^5
$(\mu_1,\eta_1,\eta_1), (\eta_1,\mu_1,\eta_1), (\eta_1,\eta_1,\mu_1)$	0,1323941527	
$(\mu_2,\eta_2,\eta_2), (\eta_2,\mu_2,\eta_2), (\eta_2,\eta_2,\mu_2)$	0,1259391805	

mit $\mu_1 = 0{,}059715$, $\eta_1 = 0{,}470142$, $\mu_2 = 0{,}797426$, $\eta_2 = 0{,}101286$

4.4 Optimierung der Flussberechnung

In Abschnitt 3.1 wurde die Matrixgleichung

$$M^i \overrightarrow{u_t} + K^i \overrightarrow{u} + F^i = 0$$

eingeführt. Dies ist eine rein lokale Beschreibung, d. h. auf Ω_i beschränkt, und M^i und K^i sind im Allgemeinen dicht besetzte Matrizen der Größe $k_i \times k_i$, wobei k_i die Dimension des Funktionenraumes $V(\Omega_i)$ ist, also „klein“. Diese lokale Formulierung ist ein wesentliches Ziel der DG-Verfahren, birgt jedoch ein Problem: Das eigentliche Lösen des Systems geht sehr schnell, weil M^i klein ist, das Aufstellen der rechten Seite kann durch die Berechnung von F^i jedoch lange dauern. Selbst bei numerischen Flüssen, die dies wegen Linearität eigentlich erlauben würden, kann diese Berechnung in der lokalen Formulierung

nicht durch eine Matrixmultiplikation geschehen, da auf Werte von Nachbarelementen zugegriffen werden muss.

Dieses Probleme kann teilweise umgangen werden, wenn man aus den lokalen Matrizen eine größere Matrix assembliert. Li schreibt in [Li06] zwar explizit: „Because of the local formulation, a discontinuous finite element algorithm will not result in an assembled global matrix and this the in-core memory demand is not as strong“, Cockburn schreibt in [Coc03] jedoch von „blockdiagonalen“ Massenmatrizen, geht also offenbar durchaus von globalen Matrizen aus. Setzt man die Matrizen in einem großen System zusammen, erhält man eine Gleichung der Gestalt

$$M\vec{u_t} + K\vec{u} + F = \begin{pmatrix} M^0 & & & \\ & M^1 & & \\ & & \ddots & \\ & & & M^N \end{pmatrix} \cdot \begin{pmatrix} \vec{u_t^0} \\ \vec{u_t^1} \\ \vdots \\ \vec{u_t^N} \end{pmatrix} + \begin{pmatrix} K^0 & & & \\ & K^1 & & \\ & & \ddots & \\ & & & K^N \end{pmatrix} \cdot \begin{pmatrix} \vec{u^0} \\ \vec{u^1} \\ \vdots \\ \vec{u^N} \end{pmatrix} + \begin{pmatrix} F^0 \\ F^1 \\ \vdots \\ F^N \end{pmatrix} = 0 \tag{4.2}$$

Diese Umformung bringt in der Tat noch keinen Vorteil; hat man jedoch einen numerischen Fluss, der linear in u_h ist (insbesondere sind alle Flüsse der Form (2.6) linear), kann man F^i wie folgt umschreiben, wobei $N(\Omega_i)$ die Menge der Indizes der Nachbarelemente von Ω_i sein soll:

$$\begin{aligned} F_l^i &= \sum_{e\in\partial\Omega_i} \int_e \widehat{\vec{a}\,u_h} \cdot \vec{n_e}\varphi_l = \sum_{z\in N(\Omega_i)} \left(\sum_{j=1}^{k_i} \left(u_j^i \int_{e_{iz}} \widehat{\vec{a}\varphi_j^i} \cdot \vec{n_{e_{iz}}}\varphi_l \right) + \sum_{j=1}^{k_z} \left(u_j^z \int_{e_{iz}} \widehat{\vec{a}\varphi_j^z} \cdot \vec{n_{e_{iz}}}\varphi_l \right) \right) \\ &= \Bigg(\cdots \underbrace{\int_{e_{iz_1}} \widehat{\vec{a}\varphi_j^{z_1}} \cdot \vec{n_{e_{iz_1}}}\varphi_l}_{j=1,\ldots,k_{z_1}} \cdots \underbrace{\int_{e_{iz_2}} \widehat{\vec{a}\varphi_j^{z_2}} \cdot \vec{n_{e_{iz_2}}}\varphi_l}_{j=1,\ldots,k_{z_2}} \cdots \underbrace{\int_{e_{iz_3}} \widehat{\vec{a}\varphi_j^{z_3}} \cdot \vec{n_{e_{iz_3}}}\varphi_l}_{j=1,\ldots,k_{z_3}} \cdots \underbrace{\sum_{z\in N(\Omega_i)} \int_{e_{iz}} \widehat{\vec{a}\varphi_j^i} \cdot \vec{n_{e_{iz}}}\varphi_l}_{j=1,\ldots,k_i} \cdots \Bigg) \\ &\qquad \cdot \left(\cdots \vec{u^{z_1}} \cdots \vec{u^{z_2}} \cdots \vec{u^{z_3}} \cdots \vec{u^i} \cdots \right) \end{aligned} \tag{4.3}$$

In diesem Fall existiert also ebenfalls eine von u_h unabhängige Matrix $\hat{F}$ mit $F = \hat{F}\vec{u}$! Nun kann man (4.2) umschreiben in

$$M\vec{u_t} + K\vec{u} + \hat{F}\vec{u} = 0. \tag{4.4}$$

Dabei sind M und K blockdiagonal (d. h. hier gibt es nicht mehr Rechenaufwand als zuvor); $\hat{F}$ zwar nicht, aber dafür hat man die Berechnung des Flusses in jedem Zeitschritt (unter der Prämisse, dass $\vec{a}$ von t unabhängig ist) auf eine Multiplikation einer dünn besetzten Matrix mit einem Vektor reduziert.

4.5 Parallelisierung

Auf ersten Blick scheint es so, als würde die im vorherigen Abschnitt vorgestellte Idee der Assemblierung globaler Matrizen der natürlichen parallelen Struktur des DG-Verfahren diametral gegenüberstehen. In der Theorie ist das richtig, in der Praxis würde jedoch die *kanonische* Parallelisierung, d. h. die Berechnung verschiedener Elemente auf verschiedenen Prozessoren/Rechnern, nicht effizient funktionieren, denn für jede Berechnung von F^i müssen die Funktionswerte der Nachbarn zum gleichen Zeitpunkt bekannt sein, d. h. jeder Zeitschritt ist eine Barriere, an der Prozesse aufeinander warten und Daten austauschen müssen.

Der in [BAK99] vorgeschlagene Ansatz arbeitet daher mit einer Aufteilung von Ω in verschiedene Partitionen $P_1 \ldots, P_M$, die z. B. der Anzahl der verfügbaren Prozessoren oder Rechner entsprechen, und jede eine Menge von räumlich nah beieinanderliegenden Elementen enthalten. *Innerhalb* jeder Partition kann man nun in jedem Zeitschritt wie folgt vorgehen:

1. berechne F_{in} für alle *inneren* Elemente (die nicht am Rand der Partition liegen)
2. sende die u_h-Werte aller Randelemente an die benachbarten Partitionen (asynchron)
3. berechne $M^{-1}K\vec{u}$ für *alle* Elemente
4. empfange die u_h-Werte aller Randelemente benachbarter Partitionen
5. berechne F_{bound} für alle Randelemente Ω_i mit den empfangenen Werten
6. berechne $M^{-1}F$ und $\vec{u_t} = M^{-1}K\vec{u} + M^{-1}F$

Dieser Ansatz hat die folgenden Merkmale:

- Die (unumgängliche) Kommunikation zwischen benachbarten Elementen in jedem Zeitschritt findet im Wesentlichen nur innerhalb eines Prozesses bzw. auf dem gleichen Rechner statt; Interprozesskommunikation ist nur für die Randelemente der Partitionen nötig.
- Der Versand und Empfang dieser Daten kann asynchron während anderer Berechnungen stattfinden, ist also nicht notwendigerweise der dominierende Teil des Verfahrens.

5 Abschließende Bemerkungen

Im vorliegenden Text wurden die lineare Transportgleichung und die Methode der DG-Verfahren zur approximativen Lösung dieser Gleichung vorgestellt. Es zeigte sich, dass diese Verfahren nicht erst bei Besonderheiten wie hängenden Knoten leichter zu implementieren sind als klassische Finite-Elemente-Verfahren, sondern auch schon auf normalen Triangulierungen, und dabei eine sehr gut parallelisierbare Struktur aufweisen.

In dieser Einführung wurde bewusst der Fokus auf Techniken gelegt, die bei der (einfachen) Implementierung benötigt werden und dabei Stabilität und Konvergenz nur am Rande erwähnt. Bei einer tiefergehenden Beschäftigung mit DG-Verfahren für hyperbolische Gleichungen ist es allerdings unbedingt notwendig, sich mit diesen Fragen auseinanderzusetzen; insbesondere bei der Zeitintegration mittels expliziter Verfahren können sonst Stabilitätsprobleme auftreten, die die hohe Ordnung der eigentlichen DG-Verfahren wieder zunichte machen.

Literatur

[BAK99] BAGGAG, ABDELKADER, HAROLD ATKINS und DAVID KEYES: *Parallel implementation of the discontinuous Galerkin method.* Technischer Bericht, 1999.

[Coc97] COCKBURN, BERNARDO: *An introduction to the Discontinuous Galerkin method for convection-dominated problems.* In: *Advanced Numerical Approximation of Nonlinear Hyperbolic Equations*. Springer, 1997.

[Coc03] COCKBURN, BERNARDO: *Discontinuous Galerkin methods.* Z. Angew. Math. Mech., 2003.

[KS07] KORNHUBER, RALF und CHRISTOF SCHÜTTE: *Numerik von partiellen Differentialgleichungen (Vorlesungsskript)*, 3. Auflage, 2007.

[Li06] LI, BEN Q.: *Discontinuous Finite Elements in Fluid Dynamics and Heat Transfer*. Springer, 2006.

[RH73] REED, W H und T R HILL: *Triangular mesh methods for the neutron transport equation.* Technischer Bericht, 1973.

[ZTZ05] ZIENKIEWICZ, O. C., R. L. TAYLOR und J. Z. ZHU: *The Finite Element Method: Its Basis and Fundamentals*. Butterworth-Heinemann, 6. Auflage, 2005.