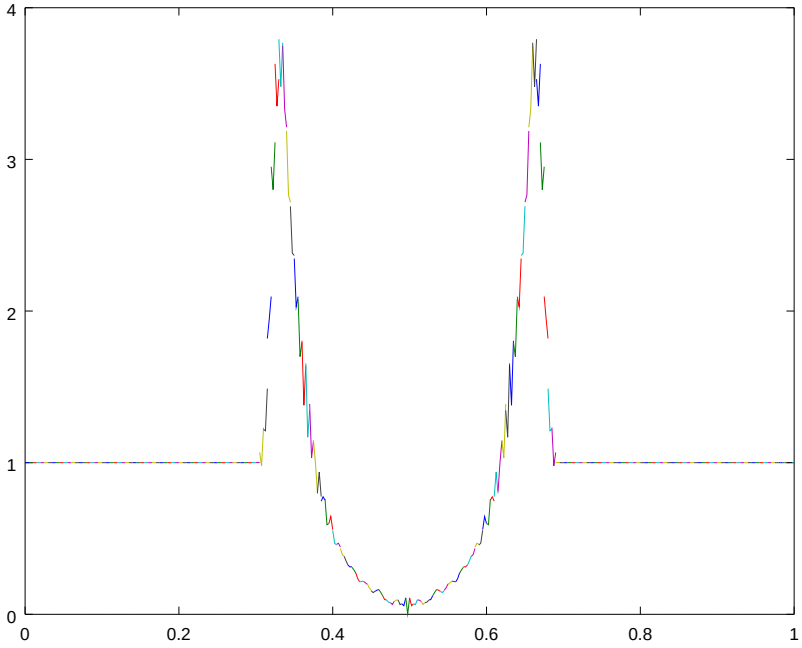
Julius-Maximilians-Universität Würzburg
Institut für Mathematik
Lehrstuhl für Mathematik VI
Angewandte Analysis

Bachelorarbeit

Positivitätserhaltendene nodale discontinuous Galerkin Methoden hoher Konvergenzordnung für die eindimensionalen Eulergleichungen



Niklas Götz

Eingereicht am 22. Juli 2017

Betreuer:
Prof. Dr. Christian Klingenberg

Zusammenfassung

Diese Bachelorarbeit erläutert die Discontiuous Galerkin Methode zur numerischen Lösung hyperbolischer Differentialgleichungen, wobei der Fokus auf den Eulergleichungen liegt. Aufbauend auf Hesthaven und Warburton wird dieses Verfahren für den eindimensionalen Fall hergeleitet und beschrieben sowie anhand verschiedener Beispiele wichtige Elemente dieses Verfahrens demonstriert.
Ziel dieser Arbeit ist es, die Bedeutung des Limiters für die Leistungsfähigkeit dieser Methode zu demonstrieren. Hierzu werden der TVD-Limiter sowie verschiedene TVB-Limiter demonstriert sowie der Positivitätslimiter nach Zhang und Shu implementiert und anschließend die Verbesserung der Genauigkeit des Verfahrens durch seine Anwendung untersucht. Es kann dabei gezeigt werden, dass das Ausführen dieses Limiters vor dem konventionellen TVB-Limiter eine qualitativ hochwertigere Lösung ermöglicht. Außerdem wird die Möglichkeit eröffnet, die Eulergleichungen in hoher Konvergenzordnung auch für solche Fälle zu lösen, bei denen durch die numerische Lösung unphysikalische Werte angenommen werden könnten. Letzteres würde ansonsten zum Zusammenbruch der Simulation führen.
Hierdurch ist man in der Lage, trotz des zusätzlichen Rechenaufwands für das Limiting in kürzerer Zeit genauerer Lösungen für die Eulergleichungen zu finden. Auch die Exaktheit bei der Berechnung von Problemen mit Unstetigkeitsstellen wird verbessert.

Inhaltsverzeichnis

1 Einleitung

Numerische Simulationen stellen in der der Physik ein zentrales Werkzeug dar, um Theorien auf ihre Vereinbarkeit mit empirischen Beobachtungen zu überprüfen. Insbesondere wenn eine große Anzahl oder besonders komplizierte Berechnungen vorgenommen werden müssen, führt an der Numerik kein Weg mehr vorbei.
Die Rolle der Computersimulationen ist unter allen Gebieten der Physik in der Astrophysik am größten. Dies hat zahlreiche Gründe: Die enormen Raum-, aber auch Zeitskalen, auf denen die Wechselwirkungen auftreten, erfordern eine enorme Anzahl an Rechenschritten. Darüber hinaus gibt es aber auch gerade in diesen Skalen enorme Differenzen, da Objekte vom Ausmaß vieler tausend Lichtjahre oder auch nur einiger hundert Kilometer sein können. Entscheidend ist aber auch die Komplexität der Probleme. Bereits für das aus der klassischen Himmelsmechanik bekannte n-Körper-Problem existiert keine praktisch verwertbare Lösung der Differentialgleichungen mehr.
Die Notwendigkeit numerischer Behandlung ist noch viel größer bei partiellen Differentialgleichungen, allen voran den Eulergleichungen [1]:

$$\begin{aligned} &\frac{\partial \rho}{\partial t} + \frac{\partial \rho u}{\partial x} = 0, \\ &\frac{\partial \rho u}{\partial t} + \frac{\partial (\rho u^2 + p)}{\partial x} = 0, \\ &\frac{\partial E}{\partial t} + \frac{\partial (E + p) u}{\partial x} = 0. \end{aligned} \tag{1.1}$$

Hierbei handelt es sich um den inkompressiblen, eindimensionalen Fall in der Erhaltungsform, denn es handelt sich hier offensichtlich um ein System von Erhaltungssätzen. Die Erhaltungsgrößen sind die Dichte ρ, der Impuls ρu und die Energie E, wobei Energie und Druck über das Ideale Gasgesetz verknüpft sind durch

$$p = (\gamma - 1)(E - \frac{1}{2}\rho u^2), \quad c = \sqrt{\frac{\gamma p}{\rho}}. \tag{1.2}$$

γ ist dabei ein von der Struktur der Moleküle abhängiger Faktor, der sich über $\frac{f+2}{f}$ berechnet, mit f, der Anzahl der Freiheitsgrade des Moleküls.
Diese nichtlinearen hyperbolischen partiellen Differentialgleichungen sind ein Sonderfall der Navier-Stokes-Gleichungen, bei dem die Viskosität und die Wärmeleitung des Fluids vernachlässigt werden, bzw. in unserem Fall auch die Kompressibilität. Deshalb kann man sie ähnlich wie die Navier-Stokes-Gleichungen aus den Newtonschen Bewegungsgleichungen und der Kontinuitätsgleichung für ein festes Volumenelement gewinnen, oder später durch Linearisierung im Gleichgewichtszustand. [15]. Die Eulergleichungen können jedoch auch aus der Bernoulli-Gleichung $e = \frac{u^2}{2} + \frac{p}{\rho} + gz = const.$ hergeleitet werden.

Die Beschränkungen, die im Gegensatz zu den Navier-Stokes-Gleichungen vorliegen, sind einerseits durch die deutlich einfachere Behandlung gerechtfertigt, andererseits aber auch für viele Fälle der Astrophysik, wo die Viskosität kaum eine Rolle spielt, eine ausgezeichnete Näherung. Hier ist ihre korrekte Lösung in vielfältigen Bereichen von außerordentlicher Bedeutung. So ist beispielsweise die genaue Struktur der Gasströmungen mindestens von ebenso großer Bedeutung bei der Sternentstehung wie magnetische Felder [13]. Aber auch bei der Bildung von Galaxien und Galaxienhaufen ist spielen Gasströme eine entscheidende Rolle, was ihre genaue Berechnung für astrophysikalische Vorhersagen unabdingbar macht.
Diese wird zunehmend vereinfacht durch den stetigen, grob Moore's Law folgenden Anstieg der Rechenleistung von Computern. Um dies voll nutzen zu können, müssen moderne Algorithmen jedoch die Fähigkeit besitzen, möglichst gut parallelisierbar zu sein, da oftmals mit mehreren hunderttausend Kernen gleichzeitig gerechnet wird. Erschwert wird dies dadurch, dass die Geschwindigkeit des Datentransfers innerhalb der Rechenmaschine nicht ebenso stark anwächst wie die Rechenleistung selbst. Die auf Diskretisierung basierenden Algorithmen müssen also für eine hohe Rechengeschwindigkeit möglichst Informationen nur von Nachbarelementen verwenden.
In der Astrophysik spielen traditionell Finite-Volumen-Verfahren eine große Rolle. Vor dem Hintergrund der Parallelisierung haben diese jedoch den großen Nachteil, dass sie für hohe Konvergenzordnungen Informationen aus einer großen Anzahl an Nachbarelementen benötigen. Mit den direkten Nachbarn alleine kann man lediglich eine zweite Ordnung erreichen.
Das alternative Finite-Elemente-Verfahren umgeht diese Problematik, indem es nicht mit lokalen Mittelwerten, sondern mit lokalen Approximationen arbeitet und so mehr Freiheitsgerade pro Element bietet. Die global definierten Basisfunktionen, auf denen die Approximation basiert, stellen jedoch ein Hindernis bei der Berechnung da und erfordern aufwendige Schritte, welche die Rechenzeit stark vergrößern. Abhilfe schafft dem die mit dem Finite-Elemente-Verfahren verwandte discontinuous Galerkin (DG) Methode. Seit ihrer Einführung in den 70ern ([21]) wurden ihre Anwendungsbereiche auf zahlreiche verschiedene Probleme partieller Differentialgleichungen ausgeweitet. Auch in der Astrophysik gewinnt dieses Verfahren zunehmend an Bedeutung, wie sich an aktuellen Papern zeigt [22]. Die DG-Methode basiert auf einer Darstellung der Lösung durch Basisfunktionen innerhalb der einzelnen Zellen, sodass keine Rekonstruktion wie bei der Finite-Volumen-Methode notwendig ist. Sie benötigt lediglich Informationen über die Nachbarzellen, und ein Großteil der Berechnungen findet über Informationen innerhalb der Zelle statt, da nur der Flux zwischen den Zellen berechnet werden muss. Das macht die Methode sehr gut parallelisierbar.
Der grundlegende Algorithmus beinhaltet aber nicht die physikalischen Beschränkungen. Wenn wir zu den Eulergleichungen zurückkehren, so sind negative Werte für Druck und Dichte offensichtlich unphysikalisch - dies widerspricht jedoch nicht der DG-Methode. Darüber hinaus müssen an Unstetigkeitsstellen auch Oszillationen unterdrückt werden, die bei numerischen Approximationen hoher Ordnung entstehen und ebenfalls unphysikalisch sind. Beide Probleme können durch Limiting gelöst werden. Die Herausforderung ist jedoch, trotz Limitings Konvergenz hoher Ordnung zu erhalten. Diesem Problem wird

sich diese Arbeit widmen.
Hierzu wird zuerst in Kapitel 2 das grundlegende Verfahren erläutert, wobei ich mich an der Notation und den Ausführungen der exzellenten Einführung von Hesthaven und Warburton orientiere [10]. Ich werde dabei auch die Beispielimplementierung von Hesthaven und Warburton erläutern und eigene Ergebnisse mit dieser aufführen. Im Anschluss daran wird in Kapitel 3 ein Fokus auf nichtlineare Probleme, zu denen auch die Eulergleichungen gehören, gelegt. Hier werden wir im Kontext von unstetigen Lösungen zum ersten Mal dem Limiting begegnen. In Kapitel 4 schließlich betrachten wir, neben einer Modifikation des TVB-Limiters, den Positivitätslimiter von Zhang und Shu [28]. Ich werde eine Implementierung desselben aufführen und anhand von Anwendungsbeispielen seine Vorzüge im Hinblick auf die Konvergenzordnung des Verfahrens aufzeigen.

2 Grundlagen des discontinuous Galerkin Verfahrens

In diesem Kapitel soll die Grundstruktur der Discontinous Galerkin Methode erläutert werden. Dabei liegt ein besonderer Schwerpunkt auf der praktischen Implementierung. Im folgenden orientiere ich mich teilweise an den Definitionen, Herleitungen und der Notation von Kapitel 1, 2, 3 und 4 in [10].

2.1 Kernelemente am Beispiel der skalaren Advektionsgleichung

Wir beginnen unsere Betrachtung mit der skalaren Advektionsgleichung

$$\frac{\partial u}{\partial t} + \frac{\partial f(u)}{\partial x} = 0, \qquad x \in [L, R] = \Omega \tag{2.1}$$

und gehen vorerst von einem linearen Fluss $f(u) = au$ aus. Wir wählen passende Anfangsbedingungen $u(x, 0) = u_0(x)$. Die Randbedingungen sind an der Oberfläche festgelegt, an der der Fluss eintritt, d.h.

$$u(L, t) = g(t) \quad \text{für} \quad a \geq 0, \qquad u(R, t) = g(t) \quad \text{für} \quad a \leq 0$$

Um diese Gleichung numerisch lösen zu können, müssen wir eine Diskretisierung unseres Problems vornehmen. Dazu wollen wir im ersten Schritt Ω durch K disjunkte Elemente, $x \in [x_l^k, x_r^k] = D^k$ annähern. Auf jedem dieser Teilbereiche werden wir unsere Lösung lokal durch Polynome der Ordnung $N = N_p - 1$ darstellen.

$$x \in D^k : u_h^k(x, t) = \sum_{n=1}^{N_p} \hat{u}_n^k(t)\psi_n(x) = \sum_{i=1}^{N_p} u_h^k(x_i^k, t) l_i^k(x) \tag{2.2}$$

In der ersten, genannt modale Form, drücken wir die Approximation u_h durch eine lokale Basis aus Polynomen aus, während wir in der zweiten, der nodalen Form, in jedem der K Teilbereiche N_p Punkte wählen (welche wir als Gitter auffassen können) und das Polynom über die interpolierenden Lagrangepolynome $l_i^k(x)$ darstellen. Die globale Lösung ist damit letztlich die direkte Summe der lokalen Lösungen. Wir definieren nun das Residuum

$$R_h(x, t) = \frac{\partial u_h}{\partial t} + \frac{\partial a u_h}{\partial x}. \tag{2.3}$$

Damit ergibt sich:

$$0 = \int (\frac{\partial u}{\partial t} + \frac{\partial f(u)}{\partial x}) dx = \int (\frac{\partial u}{\partial t} + \frac{\partial f(u)}{\partial x}) \psi_h dx =$$
$$\int (R_h - R_h + \frac{\partial u}{\partial t} + \frac{\partial f(u)}{\partial x}) \psi_h dx = \int R_h \psi_h dx$$

Hierfür führten wir einen globalen Raum V_h von Testfunktionen ψ_h ein, welchen wir definieren als $V_h = \oplus_{k=1}^{K} V_h^k = \oplus_{k=1}^{K} \text{span}\{\psi_n(D^k)\}$. Dabei nehmen wir an, dass für $(N_p, K) \to \infty$ ψ_h gegen ψ gleichmäßig konvergiert, d.h. unabhängig vom Auswertungspunkt. V_h ist damit der Raum der stückweise glatten Funktionen auf Ω_h. Der letzte Schritt folgte daraus, dass die Differenz aus dem Residuum und der eigentlichen Differentialgleichung in der L^2-Projektion verschwindet. Welche Gestalt die Testfunktionen tatsächlich haben, lassen wir dabei offen. Später werden wir eine konkrete, für unsere Zwecke vorteilhafte Wahl vornehmen.
Damit obige Bedingung erfüllt wird, müssen wir also fordern, dass das Residuum orthogonal zu allen Testfunktionen in V_h ist. Wir haben hierbei unsere Testfunktionen identisch zu unserer Polynombasis gewählt. Dies resultiert in folgender Gleichung für jedes unserer K Elemente:

$$\int_{D^k} R_h(x,t)\psi_n(x)dx = 0, \qquad 1 \leq n \leq N_p \tag{2.4}$$

Damit haben wir gleich viele Bestimmungsgleichungen wie Unbekannte in jedem Element. Da wir jedoch noch jede Wahl an Basis- bzw. Testfunktionen zugelassen haben, haben wir sowohl die Einhaltung der Randbedingungen und die Frage, wie wir die globale aus den lokalen Lösungen zurückgewinnen können, bisher vernachlässigt.
Wir nehmen nun an, dass die ψ_h glatt sind, aber über die Grenzen Elemente hinweg nicht eingeschränkt sind, also insbesondere auch nicht stetig an den Oberflächen. Mit partieller Integration folgt:

$$\int_{D^k} (\frac{\partial u_h^k}{\partial t}\psi_n - a u_h^k \frac{d\psi_n}{dx})dx = -[a u_h^k \psi_n]_{x_l^k}^{x_r^k} = -\int_{\partial D^k} \hat{\boldsymbol{n}} \cdot a u_h^k \psi_n dx, \quad 1 \leq n \leq N_p$$

$\hat{\boldsymbol{n}}$ ist dabei der nach außen zeigende, lokale Normalenvektor. Aufgrund der mangelnden Bedingungen an unsere Lösung ist die der Wert an den Rändern unserer Elemente nicht eindeutig. Wir müssen also festlegen, welchen oder welche Werte wir wählen. Dies verschieben wir auf später und arbeiten vorerst mit dem numerischen Flux $(au_h)^*$. Wir erhalten das semidiskrete Schema

$$\int_{D^k} (\frac{\partial u_h^k}{\partial t}\psi_n - a u_h^k \frac{d\psi_n}{dx})dx = -\int_{\partial D^k} \hat{\boldsymbol{n}} \cdot (a u_h^k)^* \psi_n dx, \quad 1 \leq n \leq N_p \tag{2.5}$$

Diese $K \times N_p$ Gleichungen erlauben es, die globale Lösung zu erhalten. Obige Formulierung bezeichnen wir als schwache Form. Bei weiterer partieller Integration erhalten wir die starke Form als:

$$\int_{D^k} R_h(x,t)\psi_n dx = \int_{\partial D^k} \hat{\boldsymbol{n}} \cdot (a u_h^k - (au_h)^*)\psi_n dx, \quad 1 \leq n \leq N_p \tag{2.6}$$

Wir erinnern uns daran, dass wir die Polynombasis für unsere u_h^k identisch mit den Testfunktionen gewählt haben. Deshalb können wir durch das Ausführen der Integration die schwache und starke Form folgendermaßen umformulieren:

$$\hat{\boldsymbol{M}}^k \frac{d}{dt}\hat{\boldsymbol{u}}_h^k - (\hat{\boldsymbol{S}}^k)^T a\hat{\boldsymbol{u}}_h^k = -(au_h)^*\boldsymbol{\psi}(x_r^k) + (au_h)^*\boldsymbol{\psi}(x_l^k) \tag{2.7}$$

$$\hat{\boldsymbol{M}}^k \frac{d}{dt}\hat{\boldsymbol{u}}_h^k - \hat{\boldsymbol{S}}^k a\hat{\boldsymbol{u}}_h^k = (au_h^k - (au_h)^*)\boldsymbol{\psi}(x_r^k) - (au_h^k - (au_h)^*)\boldsymbol{\psi}(x_l^k) \tag{2.8}$$

Dabei benutzen wir die lokale Massenmatrix und die lokale Steifheitsmatrix

$$\hat{\boldsymbol{M}}_{ij}^k = (\psi_i, \psi_j)_{D^k}, \quad \hat{\boldsymbol{S}}_{ij}^k = (\psi_i, \frac{d\psi_j}{dx})_{D^k}.$$

$\hat{\boldsymbol{u}}_h^k$ ist der Vektor unserer N_p lokalen Koeffizienten zu den jeweiligen Basispolynomen im Vektor $\boldsymbol{\psi}$. Wir wollen nun ein Beispiel geben, wie der numerische Flux gewählt werden kann. Motiviert wird dies von der Energiemethode für unsere Advektionsgleichung, aus der folgt:

$$\frac{d}{dt}||u||_\Omega^2 = -a(u^2(R) - u^2(L)) \tag{2.9}$$

Auch unsere numerische Approximation soll ein ähnliches Verhalten zeigen. Mit kurzem Nachrechnen folgt für jedes lokale Element:

$$\frac{d}{dt}||u_h^k||_{D^k}^2 = -a[(u_h^k)^2]_{x_l^k}^{x_r^k} + 2[u_h^k(au_h^k - (au)^*)]_{x_l^k}^{x_r^k}$$

Wir wollen ebenso wie bei der exakten Lösungen Beschränktheit und damit Stabilität garantieren. Dies erreichen wir, indem wir fordern, dass die Summe aller Zeitableitungen der Normquadrate kleiner als 0 ist. Dies ist sichergestellt, wenn wir zeigen, dass für jedes k der rechtsseitige Term, welcher die Kopplung über die Ränder darstellt, kleiner als 0 ist. Dies bedeutet zum Beispiel für die Schnittstelle zwischen den Elementen k und $k+1$:

$$\begin{aligned} &\hat{\boldsymbol{n}}^- \cdot (au_h^2(x_r^k) - 2u_h(x_h^k)(au)^*(x_r^k)) + \\ &\hat{\boldsymbol{n}}^+ \cdot (au_h^2(x_l^{k+1}) - 2u_h(x_l^{k+1})(au)^*(x_l^{k+1})) \leq 0 \end{aligned} \tag{2.10}$$

$\hat{\boldsymbol{n}}^-$ steht dabei antilinear zu $\hat{\boldsymbol{n}}^+$. Eine häufige Wahl für den numerischen Flux ist

$$f^* = (au)^* = \{\{au\}\} + |a|\frac{1-\alpha}{2}[[u]] \tag{2.11}$$

wobei wir definieren:

$$\{\{au\}\} = \frac{u^- + u^+}{2} = \frac{u^{innen} + u^{aussen}}{2} \quad [[u]] = \hat{\boldsymbol{n}}^- u^- + \hat{\boldsymbol{n}}^+ u^+ \tag{2.12}$$

Eine kurze Rechnung (Einsetzen von Gleichung 2.11 in Gleichung 2.10 mit Ausnutzung von Gleichung 2.12) zeigt dann, dass der Beitrag zur Energie aller inneren Ränder $-|a|(1-\alpha)[[u_h]]^2$ beträgt und somit für $0 \leq \alpha \leq 1$ kleiner gleich 0 ist, was die Stabilität garantiert. Darüber hinaus zeigt sich, dass wir für $\alpha = 1$ einen zentralen Flux haben, was dazu führt, dass bei periodischen Randbedingungen die Energie erhalten ist, während wir für $\alpha = 0$ einen upwind-Flux vorliegen haben.

2.2 Erweiterung auf den nichtlinearen, nichtskalaren Fall

Wir gehen nun von einem System von Erhaltungsgesetzen mit nicht zwingend linearen Fluss aus:

$$\frac{\partial \boldsymbol{u}}{\partial t} + \frac{\partial \boldsymbol{f}(\boldsymbol{u})}{\partial x} = 0, \quad x \in [L, R], \tag{2.13}$$

versehen mit der Anfangsbedingung $\boldsymbol{u}(x, 0) = \boldsymbol{u}_0(x)$.
Unser vorhandenes Schema lässt sich grundsätzlich auf gleiche Weise verwenden. Bei der Herleitung des Schemas haben wir nämlich an keiner Stelle die Linearität des Flusses verwendet. Wir müssen jedoch beachten, dass wir jetzt nicht nur u approximieren, sondern auch $f(u)$.
Von entscheidender Bedeutung ist jedoch weiterhin die Wahl des numerischen Fluxes. Neben seiner Eindeutigkeit wollen wir im Hinblick auf den großen Erfolg dieser Bedingung bei den Finite-Volumen-Verfahren fordern, dass $f*(a, b)$ monoton in dem Sinne ist, dass es im ersten Argument nicht fallend und im zweiten nicht steigend ist. Im skalaren Fall ist ein einfacher Flux, der dies erfüllt, der Lax-Friedrichs-Flux:

$$f^{LF}(a, b) = \frac{f(a) + f(b)}{2} + \frac{C}{2}\hat{\boldsymbol{n}} \cdot (a - b) \tag{2.14}$$

C ist dabei nicht fest vorgegeben. Wir stellen folgende Bedingung an C, welche uns die Monotonie garantiert:

$$C \geq \max_{\inf u_h(x) \leq s \leq \sup u_h(x)} |f_u(s)|$$

Verallgemeinert auf den nichtskalaren Fall erhalten wir folgendes Äquivalent:

$$\boldsymbol{f}^* = \{\{\boldsymbol{f}_h(\boldsymbol{u}_h)\}\} + \frac{C}{2}[[\boldsymbol{u}_h]] \tag{2.15}$$

mit der Konstante

$$C = \max_h |\lambda(\hat{\boldsymbol{n}} \cdot \frac{\partial \boldsymbol{f}}{\partial \boldsymbol{u}})|$$

wobei $\lambda(\cdot)$den Eigenwert einer Matrix bezeichnet.
Beschränken wir uns wieder auf den linearen Fall, so können wir erneut auf einfache Weise die Stabilität zeigen. Unser System hat nun die Form

$$\frac{\partial \boldsymbol{u}}{\partial t} + \boldsymbol{A}\frac{\partial \boldsymbol{u}}{\partial x} = 0 \tag{2.16}$$

Wir gehen davon aus, dass $\boldsymbol{A}$ reell diagonalisierbar ist, d.h. dass es sich um ein hyperbolisches System handelt; wir können also $\boldsymbol{A} = \boldsymbol{R}\boldsymbol{\Lambda}\boldsymbol{R}^{-1}$ schreiben. Mit der Definition $\boldsymbol{v_i} = \boldsymbol{R}^{-1}\boldsymbol{u_i}$ folgt dann unser Schema

$$\boldsymbol{M}^k \frac{d\boldsymbol{v}_i}{dt} + \boldsymbol{\Lambda}_{ii}\boldsymbol{S}\boldsymbol{v}_i = \frac{1}{2}\int_{\partial D^k} \boldsymbol{\Lambda}_{ii}\hat{\boldsymbol{n}}[[v_j]]dx, \tag{2.17}$$

wobei wir einen zentralen Flux verwendet haben, wie man auf der rechten Seite der Gleichung sieht. Dieses Schema entspricht dem für unsere Advektionsgleichung, für die

wir bereits Stabilität gezeigt hatten. Wir erhalten Stabilität nun für unser System durch folgende Rechnung:

$$||\boldsymbol{R}\boldsymbol{v}_h(t)||^2_{\Omega,h} \leq ||\boldsymbol{R}||^2_{\Omega,h} \exp(\alpha_h t)||\boldsymbol{v}_h(0)||^2_{\Omega,h} \leq$$
$$||\boldsymbol{R}||^2_{\Omega,h}||\boldsymbol{R}^{-1}||^2_{\Omega,h} \exp(\alpha_h t)||\boldsymbol{u}_h(0)||^2_{\Omega,h} \leq C_h \exp(\alpha_h t)||\boldsymbol{u}_h(0)||^2_{\Omega,h}$$

In der Tat gilt Stabilität, wenn $||\boldsymbol{R}||^2_{\Omega,h}||\boldsymbol{R}^{-1}||^2_{\Omega,h}$ beschränkt ist; dies ist aber bereits über die Diagonalisierbarkeit sichergestellt.

2.3 Implementierung

Wir wollen nun die erforderlichen Schritte aufzeigen, die nötig sind, um dieses Verfahren im eindimensionalen, skalaren Fall durchzuführen. Wir werden dabei auch direkt die Implementierung in Matlabcode aufzeigen, welche in diesem Unterkapitel zum Teil [10] entnommen sind.
Zuerst wollen wir uns dabei den Basisfunktionen zuwenden, die wir bisher nicht näher bestimmt haben. Um ein leistungsfähiges Verfahren mit guten Resultaten zu erhalten, wollen wir natürlich eine möglichst gute Konditionierung haben. Betrachten wir hierzu in der modalen Darstellung die Abbildung der unbekannten Approximationskoeffizienten auf die eigentliche Lösung:

$$\boldsymbol{M}\hat{\boldsymbol{u}} = \boldsymbol{u}$$
$$\boldsymbol{M}_{ij} = (\psi_i, \psi_j) \quad \hat{\boldsymbol{u}} = [\hat{u}_1, \ldots, \hat{u}_{N_p}]^T \quad \boldsymbol{u}_i = (u, \psi_i)$$

Für die Kondition bedeutet dies, dass $\boldsymbol{M}$ gut konditioniert sein muss. Bei einer simplen Monombasis ist dies nicht der Fall, wie sich leicht nachrechnen lässt. Wir verwenden stattdessen die orthogonalen, normierten Jacobi-Polynome, bei der $\boldsymbol{M}$ zur Einheitsmatrix wird. Wir wollen eine Verbindung zwischen der modalen und nodalen Form etablieren, da unsere Massenmatrix in der nodalen Form aufgrund der Basis aus den interpolierenden Lagrangepolynomen schlecht konditioniert ist. Betrachten wir die Darstellung

$$u(\xi_i) = \sum_{n=1}^{N_p} \hat{u}_n \tilde{P}_{n-1}(\xi_i)$$

wobei die ξ_i paarweise verschiedene Gitterpunkte sind. In diesem Fall gilt $u(\xi_i) = u_h(\xi_i)$ und wir erhalten die modale und nodale Form als

$$u_h(r) = \sum_{n=1}^{N_p} \hat{u}_n \tilde{P}_{n-1}(r) = \sum_{n=1}^{N_p} u(\xi_i) l_i(r) \tag{2.18}$$

mit den interpolierenden Lagrangepolynomen

$$l_i(r) = \prod_{\substack{j=1 \\ j \neq i}}^{N_p} \frac{r - \xi_i}{\xi_i - x_j}$$

Wir führen folgende Darstellung des Problems ein:

$$\boldsymbol{V}\hat{\boldsymbol{u}} = \boldsymbol{u}$$
$$\boldsymbol{V}_{ij} = \tilde{P}_{j-1}(\xi_i) \quad \hat{\boldsymbol{u}} = \hat{u}_i \quad \boldsymbol{u}_i = u(\xi_i)$$

Wobei $\boldsymbol{V}$die sog. Vandermondematrix ist. Aufgrund der Eindeutigkeit der Polynominterpolation gilt $\boldsymbol{V}^T\boldsymbol{l}(r) = \tilde{\boldsymbol{P}}(r)$.
Wir rufen in Erinnerung, dass folgender Term minimiert werden soll:

$$||u - u_h||_\infty \leq (1 + \Lambda_n)||u - u^*||_\infty \tag{2.19}$$

wobei u^* die beste Approximation der Ordnung N darstellt und Λ die Lebesque-Konstante ist, welche natürlich nur von unseren Gitterpunkten abhängt. Sie beschreibt das Verhältnis unseres Approximationsfehlers zu dem, der optimal wäre. Demzufolge können wir unser Ergebnis verbessern, wenn wir die Lebesque-Konstante minimieren, d.h., wir wollen die l_i möglichst klein wählen, da gilt:

$$\Lambda_n = \max_{x\in[a,b]} \sum_{j=0}^{n} |l_j(x)| \tag{2.20}$$

Wenn wir nun mit der Cramerschen Regel die Abbildung der Lagrangepolynome mittels der Vandermondematrix auf die Jacobipolynome nach den l_i auflösen wollen, zeigt sich, dass die Minimierungsforderung äquivalent dazu ist, zu fordern, dass die Determinante unserer Vandermonde-Matrix maximal ist. Im Eindimensionalen wird dies von den Legendre-Gauss-Lobatto-Quadraturpunkten (LGL) erfüllt, definiert über die Nullstellen von $(1 - r^2)\tilde{P}'_N(r)$.
Die Verbindung zwischen nodaler und modaler Form erhalten wir nun mittels folgender Relationen:

$$\boldsymbol{u} = \boldsymbol{V}\hat{\boldsymbol{u}} \quad \boldsymbol{V}^T\boldsymbol{l}(r) = \tilde{\boldsymbol{P}}(r) \quad \boldsymbol{V}_{ij} = \tilde{P}_j(r_i) \tag{2.21}$$

In Zukunft wollen wir uns auf die nodale Darstellung konzentrieren.
Zur tatsächlichen Implementierung verwenden wir die rekursive Definition der Jacobi-Polynome. Die LGL erhalten wir über die bekannten Algorithmen der Gauß-Quadratur. Um sie in jedem Element unseres Gitters verwenden zu können, projezieren wir den Intervall [-1,1] auf die Größe der einzelnen Elemente. Die Vandermonde-Matrix lässt sich dann leicht mit den errechneten Werten an diesen Punkten befüllen.
Wir wollen nun zeigen, wie wir mit diesen Vorbereitungen die lokalen Operatoren aus dem ersten Abschnitt, $\boldsymbol{M}^k$ und $\boldsymbol{S}^k$ berechnen können. Wir betrachten die Massenmatrix in einem Gitterelement:

$$\boldsymbol{M}^k_{ij} = \int_{x_l^k}^{x_r^k} l_i^k l_j^k dx = \frac{h^k}{2}\int_{-1}^{1} l_i l_j dr = \frac{h^k}{2}\boldsymbol{M}_{ij} \tag{2.22}$$

Mit der oben hergestellten Verbindung zwischen modaler und nodaler Form, insbesondere die Verbindung zwischen Langrange- und Jacobi-Polynomen über die Vandermonde-

Matrix, folgt weiter:

$$\boldsymbol{M}_{ij} = \int_{-1}^{1} \sum_{n=1}^{N_p} (\boldsymbol{V}^T)^{-1}_{in} \tilde{P}_{n-1}(r) \sum_{m=1}^{N_p} (\boldsymbol{V}^T)^{-1}_{jm} \tilde{P}_{m-1}(r) = \sum_{n=1}^{N_p} (\boldsymbol{V}^T)^{-1}_{in} (\boldsymbol{V}^T)^{-1}_{jn}$$

Daraus folgt:

$$\boldsymbol{M}^k = \frac{h^k}{2} \boldsymbol{M} = \frac{h^k}{2} (\boldsymbol{V}\boldsymbol{V}^T)^{-1} \tag{2.23}$$

Wir können also $\boldsymbol{M}$ bequem durch die bereits berechnete Vandermonde-Matrix ausdrücken. Fahren wir nun mit $\boldsymbol{S}$ fort. Wir definieren:

$$\boldsymbol{D}_{r,(i,j)} = \frac{dl_j}{dr}|_{r_i} \quad \rightarrow \boldsymbol{V}^T \boldsymbol{D}_r^T = (\boldsymbol{V}_r)^T \quad \boldsymbol{V}_{r,(i,j)} = \frac{d\tilde{P}_j}{dr}|_{r_i} \tag{2.24}$$

Die nötige Ableitung der Jacobipolynome lässt sich über Rekursionsformeln leicht berechnen. Wir sehen nun:

$$(\boldsymbol{M}\boldsymbol{D}_r)_{(i,j)} = \sum_{n=1}^{N_p} \boldsymbol{M}_{in} \boldsymbol{D}_{r,(n,j)} = \int_{-1}^{1} l_i(r) \sum_{n=1}^{N_p} \frac{dl_j}{dr}|_{r_n} l_n(r) dr = \int_{-1}^{1} l_i(r) \frac{dl_j(r)}{dr} dr = \boldsymbol{S}_{ij}$$

In unserem Schema trat auch ein Oberflächenintegral auf. Im 1D-Fall kann man dies implementieren, indem man mit einem $N_p \times 2$-Array multipliziert.
Wir wollen uns nun mit der Implementierung des Gitters beschäftigen.
Dazu haben wir zuerst zwei Datenstrukturen: den K+1-Vektor $\boldsymbol{VX}$, der die Koordinaten der Endpunkte unserer Elemente enthält (Unterteilung in gleich große Gebiete), und die Matrix $\boldsymbol{EToV}$, welche in jeder Zeile die Nummern der beiden Endpunkte eines Elements enthält.

```
function [Nv, VX, K, EToV] =MeshGen1D(xmin,xmax,K)

Nv = K+1;

VX = (1:Nv);
for i =1:Nv
  VX(i) =(xmax-xmin)*(i-1)/(Nv-1) +xmin;
end

EToV =zeros(K, 2);
for k =1:K
  EToV(k,1) =k; EToV(k,2) =k+1;
end
return
```

Um nun die Auswertungspunkte innerhalb dieser Elemente zu bestimmen, berechnen wir die LGL-Punkte für eine Simulation mit Polynomen N-ter Ordnung im Standardintervall [-1,1] und bilden diese dann in unser Element ab. So erhalten wir den Vektor der Koordinaten der Gitterpunkte $\boldsymbol{x}$. Gleichzeitig bestimmen wir mit diesen Punkten

auch unsere Massen- und Steifheitsmatrix in Zeile 3 und 4. Wir speichern uns ebenso die Koordinaten der Randpunkte der Elemente. Die Definition der Normalenvektoren ist in 1D noch trivial.

```
1 r = JacobiGL(0,0,N);
2
3 V  = Vandermonde1D(N, r); invV =inv(V);
4 Dr = Dmatrix1D(N, r, V);
5
6 LIFT =Lift1D();
7
8 va = EToV(:,1)'; vb =EToV(:,2)';
9 x = ones(N+1,1)*VX(va) +0.5*(r+1)*(VX(vb)-VX(va));
```

Desweiteren definieren wir die $2K \times N_\nu$-Matrix $\boldsymbol{FToV}$ (N_νAnzahl aller Gitterpunkte), welche jedem Punkt zuordnet, ob und zu welchem Element er ein Randpunkt ist, und die Berührungsmatrix $\boldsymbol{FToF} = (\boldsymbol{FToV})(\boldsymbol{FToV})^T$. Eine 1 in dieser Matrix zeigt an, dass die beiden Punkte an berührenden Flächen sind (oder identische). Man beachte im Code in Zeile 9, dass hier ausgenutzt wird, dass es sich um eine dünnbesetzte Matrix handelt. Desweiteren erzeugen wir noch die Matrix $\boldsymbol{EToE}$ und $\boldsymbol{EToF}$, die angeben, welche Elemente sich berühren und welche Oberflächen die Elemente haben. $\boldsymbol{EToE}$ enthält an der Stelle (k,i) die Nummer des Elements, an das das k-te Element mit der i-ten Oberfläche anschließt, während $\boldsymbol{EToF}$ an dieser Stelle die Nummer der Oberfläche des anderen Elements (die Nummer entspricht in 1D 1 für die linke und 2 für die rechte Oberfläche) hat.

```
1  function [EToE, EToF] =Connect1D(EToV)
2
3  Nfaces =2;
4
5  K = size(EToV,1); TotalFaces =Nfaces*K; Nv =K+1;
6
7  vn = [1,2];
8
9  SpFToV =spalloc(TotalFaces, Nv, 2*TotalFaces);
10 sk = 1;
11 for k=1:K
12   for face=1:Nfaces
13      SpFToV( sk, EToV(k, vn(face))) =1;
14      sk = sk+1;
15   end
16 end
17
18 SpFToF =SpFToV*SpFToV' -speye(TotalFaces);
19
20 [faces1, faces2] =find(SpFToF==1);
```

```
21
22 element1 =floor( (faces1-1)/Nfaces ) +1;
23 face1   =   mod( (faces1-1), Nfaces ) +1;
24 element2 =floor( (faces2-1)/Nfaces ) +1;
25 face2   =   mod( (faces2-1), Nfaces ) +1;
26
27 ind =sub2ind([K, Nfaces], element1, face1);
28 EToE     = (1:K)'*ones(1,Nfaces);
29 EToF     = ones(K,1)*(1:Nfaces);
30 EToE(ind) =element2; EToF(ind) =face2;
31 return
```

Im letzten Schritt erzeugen wir Vektoren der Randpunkte an den Oberflächen, für die inneren (***vmapM***) und äußeren (***vmapP***) Werte mit Hilfe der bereits definierten Matrizen, wobei die Werte in den Vektoren ihre Nummer unter der Gesamtheit aller Auswertungspunkte angeben. Wir verwenden dabei auch die Matrix ***Fmask***, die elementweise die Randpunkte enthält (da jedes Element ja den gleichen Satz Auswertungspunkte hat). Wenn ein Element sowohl innere wie auch äußere Informationen enthält (wie es initialisiert wurde), so wird dies als Randelement erkannt (Zeile 23). Die Randpunkte am Rande unseres Gitters werden in ***vmapB*** gespeichert, welcher die Nummer unter allen Auswertungspunkten enthält. ***mapB*** enthält hingegen die Nummer unter allen Randpunkten.

```
1  function [vmapM, vmapP, vmapB, mapB] =BuildMaps1D
2
3  Globals1D;
4  [...]
5  for k1=1:K
6    for f1=1:Nfaces
7      vmapM(:,f1,k1) =nodeids(Fmask(:,f1), k1);
8    end
9  end
10
11 for k1=1:K
12   for f1=1:Nfaces
13     k2 =EToE(k1,f1); f2 =EToF(k1,f1);
14     vidM =vmapM(:,f1,k1); vidP =vmapM(:,f2,k2);
15     x1  = x(vidM); x2 =x(vidP);
16     D = (x1 -x2 ).^2;
17     if (D<NODETOL) vmapP(:,f1,k1) =vidP; end;
18   end
19 end
20
21 vmapP =vmapP(:); vmapM =vmapM(:);
22
23 mapB =find(vmapP==vmapM); vmapB =vmapM(mapB);
24
```

```
25 mapI =1; mapO =K*Nfaces; vmapI =1; vmapO =K*Np;
26
27 return
```

Wir haben nun erreicht, dass unser Gitter definiert ist, inklusive der Zuordnung der korrekten Grenzflächen und Randgebiete. Darüber hinaus konnten wir das Schema der Disontinuous-Galerkin-Methode mit Matrizen ausdrücken. Damit ist es uns möglich, unser Problem auf folgende Gleichung zu reduzieren:

$$\frac{d\boldsymbol{u}_h}{dt} = \boldsymbol{L}_h(\boldsymbol{u}_h, t) \tag{2.25}$$

Diese gewöhnliche Differentialgleichung können wir offensichtlich mit Runge-Kutta-Methoden lösen. Es gibt dabei verschiedene Möglichkeiten, wie solche, die besonders wenig Rechenleistung erfordern, solche, die besonders stabil sind und wiederum welche, die wenig Speicherplatz erfordern. Wir entscheiden uns für letzteres. In vierter Ordnung lautet dieses Verfahren (die Koeffizienten werden vorab definiert; diese wurden bezogen aus [3]):

$$\begin{aligned}
&\boldsymbol{p}^{(0)} = \boldsymbol{u}^n, \\
&i \in [1, \ldots, 5] : \begin{cases} \boldsymbol{k}^{(i)} = a_i \boldsymbol{k}^{(i-1)} + \Delta t L_h(\boldsymbol{p}^{(i-1)}, t^n + c_i \Delta t) \\ \boldsymbol{p}^{(i)} = \boldsymbol{p}^{(i-1)} + b_i \boldsymbol{k}^{(i)} \end{cases} \\
&\boldsymbol{u}_h^{n+1} = \boldsymbol{p}^{(5)}
\end{aligned}$$

Bei der tatsächlichen Berechnung wird in einer Driverfunktion die Ordnung des Verfahrens, der zu untersuchende Bereich in Raum und Zeit und die Anfangsbedingungen initialisiert. Daraufhin wird nach dem erläuterten Verfahren das Gitter erstellt. Wir können dabei weite Teile des Codes für jedes beliebige eindimensionale Problem verwenden und müssen nur geringe Änderungen vornehmen. Unser Codebeispiel ist dabei ab jetzt unsere bekannte Advektionsgleichung, welche mit der Anfangsbedingung $sin(x)$ im Intervall [0,2] bis zum Zeitpunkt $t = 10$ in 10 Zellen in 8. Ordnung gelöst wird.

```
1 Globals1D;
2 N = 8;
3 [Nv, VX, K, EToV] =MeshGen1D(0.0,2.0,10);
4 StartUp1D;
5 u = sin(x);
6 FinalTime =10;
7 [u] = Advec1D(u,FinalTime);
```

Nun beginnt die Berechnung der numerischen Lösung. Dies übernimmt ein gesondertes Programmmodul, das lediglich die initialisierte Funktion und die Endzeit erhält. Dazu bestimmen wir erst den Schritt, der in der Zeit stattfindet. Wir gehen im folgenden wieder, wie zu Beginn, von der Advektionsgleichung aus. Hier können wir den Zeitschritt gemäß der CFL-Bedingung wählen, welche folgende Form annimmt:

$$\Delta t \leq \frac{C}{a} \min_{k,i} \Delta x_i^k \tag{2.26}$$

wobei gilt $\Delta x_i^k = x_{i+1}^k - x_i^k$.
Diese Wahl liegt darin begründet, dass der Zeitschritt nicht zu groß werden darf, damit das Verfahren nicht instabil wird. Wir können die Ausgangsgleichung zur Zeitintegration auch als Eigenwertgleichung aufstellen, wobei λ die Eigenwerte von L_h sind und einen negativen Realteil haben:

$$u_t = \lambda u$$

Δt soll klein genug sein, dass das gesamte Eigenwertspektrum in den stabilen Bereich der komplexen Ebene um $\lambda \Delta t$ passt. Durch numerische Experimente (siehe [10]) zeigt sich folgende Abschätzung:

$$max(|\lambda_N|) \leq \frac{3}{2} \max_i(\Delta_i r)^{-1} \rightarrow \Delta t \leq C(N) \min(\Delta x) \tag{2.27}$$

C enthält die Information über die Größe des stabilen Bereichs und skaliert mit $\mathcal{O}(1)$ und wird für unsere Berechnungen in diesem Beispiel auf 0,75 gesetzt. Im allgemeinen, nichtskalaren Fall kann anstatt a das Maximum der reellen Eigenwerte gewählt werden. Die erwähnten numerischen Experimente legen auch nahe, dass die CFL-Konstante mit $\frac{1}{N^2}$ skaliert, weshalb in allgemeinen Fällen von einem festen Wert für die Konstante abzuraten ist.
Die konkrete Zeitintegration, bei der wir a auf 2π festlegen und die Runge-Kutta-Koeffizienten aus Zeile 16,18 und 19 extern abgespeichert haben, findet wie folgt statt:

```
1  function [u] =Advec1D(u, FinalTime)
2
3  Globals1D;
4  time =0;
5
6  resu =zeros(Np,K);
7
8  xmin =min(abs(x(1,:)-x(2,:)));
9  CFL=0.75; dt  = CFL/(2*pi)*xmin; dt =.5*dt;
10 Nsteps =ceil(FinalTime/dt); dt =FinalTime/Nsteps;
11
12 a = 2*pi;
13
14 for tstep=1:Nsteps
15     for INTRK =1:5
16         timelocal =time +rk4c(INTRK)*dt;
17         [rhsu] =AdvecRHS1D(u, timelocal, a);
18         resu =rk4a(INTRK)*resu +dt*rhsu;
19         u = u+rk4b(INTRK)*resu;
20     end;
21
22     time =time+dt;
23 end;
24 return
```

In diesem Schema benötigen wir L_h, was der rechten Seite unseres DG-Verfahrens entspricht und in ***AdvecRHS1D*** berechnet wird. Diese nimmt für die Advektionsgleichung in 1D folgende Form an (starke Form):

$$\frac{d\boldsymbol{u}_h^k}{dt} = -a(\boldsymbol{M}^k)^{-1}\boldsymbol{S}\boldsymbol{u}_h^k + (\boldsymbol{M}^k)^{-1}\left[\boldsymbol{l}^k(x)(au_h^k - (au)^*)\right]_{x_l^k}^{x_r^k} \tag{2.28}$$

Dieses müssen wir nun für jeden Zeitschritt 5 mal berechnen, was von einer eigenen Funktion übernommen wird. Dabei legen wir den α-Faktor fest, mit dem wir das Verhältnis zwischen zentralen und upwind-Flux festlegen können. Wir erinnern uns:

$$(au)^* = \{\{au\}\} + |a|\frac{1-\alpha}{2}[[u]] \tag{2.29}$$

Genau diese Differenzen, die im Fluxterm vorkommen, werden in Zeile 7 berechnet, indem der Vektor $\boldsymbol{u}$ an den an den Elementgrenzen ausgewertet wird und wir damit den Flux erhalten. Für die Grenzen des betrachteten Gebiets nutzen wir zusätzlich noch die Randbedingungen aus, welche die exakte Lösung bei $x = 0$ ist, da aufgrund unserer Wahl von a dies jene Intervallgrenze ist, durch die der Fluss eintritt. Zuletzt müssen wir nur noch für den ersten Teil der obigen Gleichung unseren Differentiationsoperator auf $\boldsymbol{u}$ anwenden und nachskalieren sowie den Oberflächenintegraloperator auf den zweiten Term anwenden.

```
1  function [rhsu] =AdvecRHS1D(u,time, a)
2
3  Globals1D;
4
5  alpha=1;
6  du = zeros(Nfp*Nfaces,K);
7  du(:) =(u(vmapM)-u(vmapP)).*(a*nx(:)-(1-alpha)*abs(a*nx(:)))/2;
8
9  uin =-sin(a*time);
10 du (mapI) =(u(vmapI)- uin ).*(a*nx(mapI)-(1-alpha)*abs(a*nx(mapI)))/2;
11 du (mapO) =0;
12
13 rhsu =-a*rx.*(Dr*u) +LIFT*(Fscale.*(du));
14 return
```

Nun muss nur noch die Zeitintegration hinreichend oft ausgeführt werden, um letztlich den Wert unserer Lösungsfunktion an den $K \times (N+1)$-Auswertungspunkten zu erhalten.

2.4 Erste Anwendungen und Analyse

Wir wollen das erarbeitete Grundverfahren nun auch zum ersten Mal auf seine Leistungsfähigkeit prüfen. Hierzu wollen wir beim Advektionsproblem bleiben und die Konvergenzrate in der L^2-Norm für verschiedene Werte von N betrachten. Die Berechnung

des L^2-Fehlers folgt dabei [9]. Da wir die exakte Lösung kennen, können wir diese an den gleichen Punkten berechnen wie auch unsere Approximation. Da diese Punkte die LGL-Puntke sind, wird die nötige Integration trivial, da wir die einzelnen quadrierten Differenzen lediglich gewichten müssen, um eine Gauß-Quadratur auszuführen. In folgender Tabelle ist der Fehler und die Konvergenzrate für das homogene Advektionsproblem für K=10 und K=20 aufgeführt. Es ist zu beachten, dass für höhere Ordnungen die CFL-Konstante deutlich niedriger gewählt werden muss, da unsere Runge-Kutta-Integration lediglich vierter Ordnung ist.
Die Ergebnisse in Tabelle 2.1 stimmen sehr gut mit denen in [10] und in [17] überein und legen eine Abhängigkeit des Fehlers von Zellgröße h und Ordnung N der Form

$$||u - u_h||_{\Omega,h} \leq Ch^{N+1} \tag{2.30}$$

nahe. Es ist zu beachten dass demzufolge eine große Genauigkeit auf zwei Wege erreicht werden kann: Über das Vergrößern von K und über das Vergrößern von N. Es ist von der konkreten Problemstellung abhängig, welcher Weg am effizientesten ist.

Tab. 2.1: Fehler und Konvergenzraten für das homogene Advektionsproblem

N	Fehler für K=10	Fehler für K=20	Konvergenzrate
2	0.0063168	7.9490e-004	2.9910
4	6.5325e-006	1.8487e-007	5.1430
6	3.0741e-009	2.8343e-011	6.7610

In einem weiteren Schritt wird noch die Zeitabhängigkeit des Approximationsfehlers untersucht. Hierzu kehren wir zum Fall N=2 und K=10 zurück und betrachten das Problem zu den Zeitpunkten t=π,0.1π und 0.01π.

Tab. 2.2: Zeitabhängigkeit des Fehlers für das homogene Advektionsproblem

Fehler für $t = \frac{\pi}{1}$	Fehler für $t = \frac{\pi}{10}$	Fehler für $t = \frac{\pi}{100}$
0.0063196	0.0050875	0.0047635

Auch hier stützen unsere Beobachtungen die These von [10], dass der Fehler mit

$$||u - u_h||_{\Omega,h} \simeq (c_1 + c_2 T)h^{N+1} \tag{2.31}$$

skaliert (eine theoretische Begründung dieser Relation findet sich in Kapitel 4.5 von [10]). Abbildung 2.1 bestätigt unsere geringen Werte für den Fehler im Fall N=2, aber auch die deutliche Verbesserung bei Halbierung der Zellgröße, wie sich an den Übergängen zwischen den einzelnen Zellen in der Grafik erkennen lässt.
Im nächsten Schritt wollen wir zeigen, dass sich das Verfahren problemlos auch auf inhomogene Situationen erweitern lässt. Wir betrachten dazu den Quellterm $h(x,t) = x - 3t$. Um unser Verfahren daran anzupassen, müssen lediglich zwei Änderungen vorgenommen

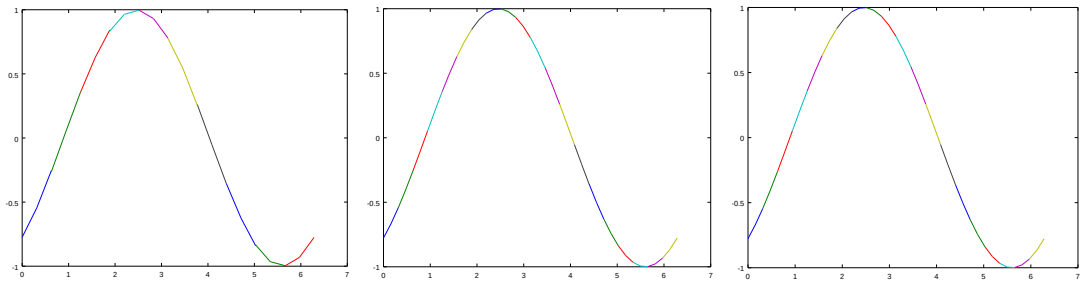


Abb. 2.1: Homogenes Advektionsproblem, N=2, $t = \pi$; v.l.n.r. : K=10, K=20, exakte Lösung

werden. Einerseits muss die Randbedingung an die veränderte exakte Lösung angepasst werden. Andererseits muss auch der Fluxterm modifiziert werden. Hierzu betrachten wir unsere neue starke DG-Formulierung:

$$\int_{D^k} (\frac{\partial u_h^k}{\partial t}\psi_n - a u_h^k \frac{d\psi_n}{dx})dx = \int_{\partial D^k} \hat{\boldsymbol{n}} \cdot (a u_h^k - (au_h)^*)\psi_n dx + \int_{D^k} h(x,t)\psi_n, \quad 1 \leq n \leq N_p$$

Wir sehen, dass damit in unserer Massen-und Steifheitsmatrixformulierung folgt:

$$\hat{\boldsymbol{M}}^k \frac{d}{dt}\hat{\boldsymbol{u}}_h^k - \hat{\boldsymbol{S}}^k a \hat{\boldsymbol{u}}_h^k = (a u_h^k - (au_h)^*)\boldsymbol{\psi}(x_r^k) - (a u_h^k - (au_h)^*)\boldsymbol{\psi}(x_l^k) + \hat{\boldsymbol{M}}^k h(x,t)$$

Daraus folgt, dass wir lediglich in ***AdvectRHS1D*** unseren Quellterm hinzuaddieren müssen. Fehler und Konvergenzrate ändern sich hierdurch höchstens marginal (in unseren Testläufen lediglich im Rahmen des Rundungsfehlers), für die Werte sei deshalb auf den homogenen Fall verwiesen. Die Qualität der Ergebnisse lässt sich auch an der beigefügten grafischen Darstellung nachvollziehen.

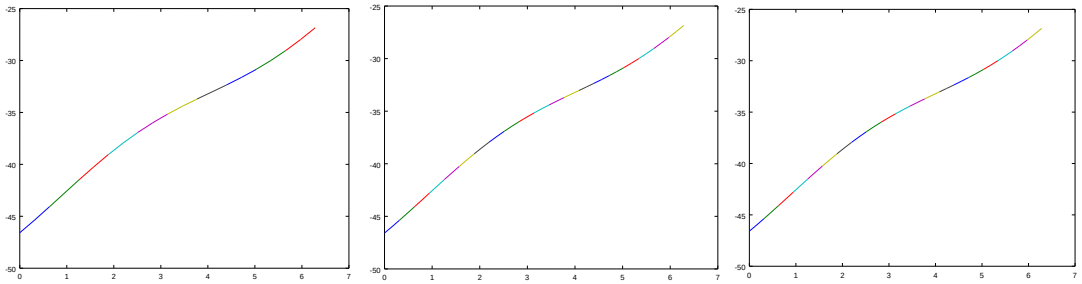


Abb. 2.2: Inhomogenes Advektionsproblem, N=2, $t = \pi$; v.l.n.r. : K=10, K=20, exakte Lösung

In einem vorerst letzten Schritt wollen wir zeigen, dass sich auch Systeme von linearen hyperbolischen partiellen Differentialgleichungen lösen lassen. Diese sollen die Form

$$\frac{\partial \boldsymbol{u}}{\partial t} + \boldsymbol{A}\frac{\partial \boldsymbol{u}}{\partial x} = 0 \tag{2.32}$$

haben und zur Vereinfachung zweidimensional sein. Da unser Flux ja entlang der Normalen betrachtet wird, interessiert uns der Operator $\Pi = (\hat{n}_x \boldsymbol{A}_1 + \hat{n}_y \boldsymbol{A}_2)$. Dieser sei reell diagonalisierbar mit der Matrix $\boldsymbol{B}$, was äquivalent dazu ist, dass das System hyperbolisch ist. Wir unterscheiden zwischen positiven und negativen Eigenwerten, welche zu den charakteristischen Variablen gehören, die sich mit bzw. gegen die Normale ausbreiten.
Für ein glattes Π lässt sich ein upwind-Flux über

$$(\Pi \boldsymbol{u})^* = \boldsymbol{B}(\Lambda^+ \boldsymbol{B}^{-1} \boldsymbol{u}^- + \Lambda^- \boldsymbol{B}^{-1} \boldsymbol{u}^+)$$

gewinnen.
Für den allgemeineren Fall, in dem Π nicht stetig ist, muss ein anderer Weg gewählt werden. Angenommen, wir haben die Eigenwerte $\lambda_1 = \lambda$ und $\lambda_2 = -\lambda$. So können wir die Rankine-Hugoniot-Bedingung ausnutzen (siehe hierzu auch [11]) und erhalten

$$-\lambda_i[\boldsymbol{u}^- - \boldsymbol{u}^+] + [(\Pi \boldsymbol{u})^- - (\Pi \boldsymbol{u})^+] = 0. \tag{2.33}$$

Unseren Flux erhalten wir daraus, dass wir noch einen Zwischenzustand einbauen.
Wir betrachten nun das System

$$\begin{aligned} \frac{\partial u}{\partial x} + \frac{\partial v}{\partial t} &= 0 \\ \frac{\partial v}{\partial x} + \frac{\partial u}{\partial t} &= 0 \end{aligned}$$

mit den Anfangsbedingungen $u_0 = \sin(x)$ und $v_0 = 0$.
Wir finden offensichtlich die Eigenwerte 1 und -1. Dann kann man mit Rankine-Hugoniot folgern:

$$\begin{aligned} (\boldsymbol{q}^* - \boldsymbol{q}^-) + (\Pi \boldsymbol{q})^* - (\Pi \boldsymbol{q})^- &= 0, \\ (\boldsymbol{q}^* - \boldsymbol{q}^+) + (\Pi \boldsymbol{q})^* - (\Pi \boldsymbol{q})^+ &= 0, \end{aligned}$$

woraus mit

$$(\Pi \boldsymbol{q})^\pm = \hat{\boldsymbol{n}} \begin{bmatrix} u^\pm \\ v^\pm \end{bmatrix}$$

folgt das gilt:

$$(\Pi \boldsymbol{q})^* = \hat{\boldsymbol{n}} \left(\begin{bmatrix} \{\{u\}\} \\ -\{\{v\}\} \end{bmatrix} + \frac{1}{2} \begin{bmatrix} [[u]] \\ [[v]] \end{bmatrix} \right), \tag{2.34}$$

was mit der Definition $\Pi = (\hat{n}_x \boldsymbol{A}_1 + \hat{n}_y \boldsymbol{A}_2)$ den numerischen Flux ergibt. In unserer starken Formulierung der DG-Methode müssen wir nun lediglich die Terme

$$\begin{aligned} u^- - u^* &= \frac{1}{2}(\hat{\boldsymbol{n}}[[v]] - [[u]]) \\ v^- - v^* &= \frac{1}{2}(\hat{\boldsymbol{n}}[[u]] - [[v]]) \end{aligned} \tag{2.35}$$

einbauen, über die das Oberflächenintegral ausgeführt wird. Zusammen mit den periodischen Randbedingungen

```
ubc = -u(vmapB); du (mapB) =u(vmapB) -ubc;
vbc =  v(vmapB); dv (mapB) =v(vmapB) -vbc;
```

muss unsere DG-Verfahren lediglich simultan für u und v ausgeführt werden, was ebenfalls nur einer geringen Änderung des Algorithmus bedarf und erneut die Stärke der Methode zeigt.
Auch hier erreichen wir die gleichen Konvergenzraten und sehen das identische Verhalten zu vorhin, und das sowohl für u als auch für v. Zur Verdeutlichung sei noch ein Plot der numerischen Lösung beigefügt.

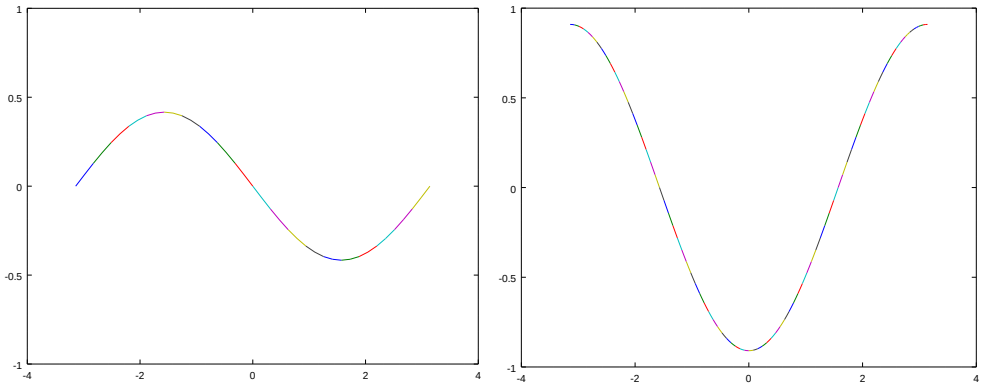


Abb. 2.3: Zweidimensionales, lineares System von Erhaltungsgleichungen, N=2, t = 2, K=20; links u, rechts v

3 Modifikationen für den nichtlinearen Fall

Bei den eindimensionalen Eulergleichungen handelt es sich um nichtlineare hyperbolische partielle Differentialgleichungen. Auch wenn sich, wie in Kapitel 2.2 gezeigt, das Verfahren grundsätzlich auch auf diesen Fall erweitern lässt, werden wir mit zusätzlichen Herausforderungen konfrontiert, welche im folgenden erläutert werden sollen. Wir folgen dabei stellenweise den Ausführungen in Kapitel 5 von [10].

3.1 Kurzer Exkurs zu Erhaltungsgleichungen

Wir betrachten wieder unsere Advektionsgleichung

$$\frac{\partial u}{\partial t} + \frac{\partial f(u)}{\partial x} = 0, \quad x \in [L, R] = \Omega, t \geq 0 \quad u(x, 0) = u_0(x),$$

wobei $f(u)$ die als konvex angenommene Fluss-Funktion ist. Wie zuvor sollen die Randbedingungen an dem Bereich vorgegeben sein, wo der Fluss eintritt. Wir haben hiermit eine skalare Erhaltungsgleichung, da folgende Bedingung erfüllt ist:

$$\frac{d}{dt} \int_a^b u(x)dx = f(u(a)) - f(u(b)) \tag{3.1}$$

Diese Identität zwischen der zeitlichen Änderung des Raumintegrals und der Differenz zwischen eintretendem und austretendem Fluss ist von fundamentaler Bedeutung innerhalb der Physik. Zahlreiche Größen, wie Masse, Ladung, Wärme etc. können dadurch beschrieben werden. Auch unsere Eulergleichungen sind Erhaltungsgleichungen.
Bisher haben wir einen linearen Fluss $f = au$ betrachtet. Für diesen lässt sich mittels der Methode der Charakteristiken zeigen, dass die Lösung dann glatt ist, wenn dies auch für die Anfangsbedingung der Fall ist (siehe auch [11])
Wir setzen $x = \psi(t; x_0)$ mit $\frac{d}{dt}\psi(t; x_0) = a$ und $\psi(0; x_0) = x_0$. Dies hat die Lösung

$$\psi(t; x_0) = at + x_0$$

Mit der Kettenregel folgern wir, dass $\frac{d}{dt}u(\psi(t; x_0), t) = 0$ gelten muss, woraus direkt folgt, dass $u(\psi(t; x_0), t) = u_0(x)$ gilt. Mit der Definition von ψ erhalten wir die Lösung

$$u(x, t) = u_0(x - at). \tag{3.2}$$

Aber dies war ja gerade unsere Aussage. Auch für nichtkonstante Koeffizienten $a(x, t)$ lässt sich dies zeigen:

$$\frac{d}{dt}\psi(t; x_0) = a(\psi(t; x_0), t) \quad \psi(0; x_0) = x_0$$

woraus sich wieder $\frac{d}{dt}u(\psi(t;x_0),t) = 0$ und damit $u(\psi(t;x_0),t) = u_0(x)$ folgern lässt. Letztlich erhalten wir das gleiche Ergebnis. Die Eindeutigkeit dieses Ergebnisses, was identisch dazu ist, dass für $u_0 = 0$ nur die 0 selbst Lösung sein kann, lässt sich folgendermaßen zeigen:
Sei $\eta(u)$ eine differenzierbare Funktion, so folgt (Multiplikation mit der Ableitung von η und Kettenregel rückwärts):

$$0 = \frac{\partial}{\partial t}\eta(u) + a\frac{\partial}{\partial x}\eta(u) = \eta(u)_t + (a\eta(u))_x - a_x\eta(u)$$

Sei $\eta(0) = 0$ und ansonsten positiv und die Ableitung von a beschränkt in Raum und Zeit. Es folgt:

$$\frac{d}{dt}\int_{\mathbb{R}} \eta(u(x,t))dx \leq C\int_{\mathbb{R}} \eta(u(x,t))dx$$

und daraus mit Gronwall:

$$\int_{\mathbb{R}} \eta(u(x,t))dx \leq e^{Ct}\int_{\mathbb{R}} \eta(u_0(x))dx,$$

was unsere Aussage liefert. Damit wissen wir, dass wir durch unser Verfahren im Falle linearer Koeffizienten ein eindeutiges Ergebnis erhalten können. Auch auf Fälle mit einem System von Gleichungen lässt sich dies erweitern.
Bei nichtlinearen Flüssen ist dies allerdings nicht mehr möglich. Man betrachte hier die Burgers-Gleichung mit $f(u) = \frac{1}{2}u^2$, so folgt für unsere charakteristischen Pfade $\frac{d}{dt}\psi(t) = f'(u(\psi,t), \psi(0) = x_0$. Betrachten wir nun $\kappa(t) = u(\psi,t)$, also die Lösung entlang unseres charakteristischen Pfades, so zeigt sich nach der Kettenregel

$$\frac{d\kappa}{dt} = \frac{\partial u}{\partial t} + \frac{1}{2}\frac{\partial u^2}{\partial \psi} = 0. \tag{3.3}$$

Sie ist also konstant. Wenn sich diese Pfade nicht schneiden, so haben wir eine eindeutige Lösung. Verwenden wir jedoch als Anfangsbedingung $u_0 = -sin(\pi x)$ im Intervall [-1,1] mit verschwindender Lösung am Rand, so bewegt sich die Information links von 0 nach rechts und umgekehrt. Aufgrund der Massenerhaltung bildet sich bei 0 ein sogenannter Schock, die Lösung ist nicht mehr stetig. Demzufolge ist auch die Ableitung nicht mehr definiert. Wir arbeiten deshalb nun mit schwachen Lösungen, also solche, die

$$\int_0^\infty \int_{-\infty}^\infty (u(x,t)\frac{\partial \phi}{\partial t} + f(u)\frac{\partial \phi}{\partial x})dxdt = 0$$

erfüllen sowie auch konsistent mit den Anfangsbedingungen sind. Allerdings gibt es so nun mehrere Lösungen, die Eindeutigkeit geht also verloren. Für unsere Burgers-Gleichung lässt sich beispielsweise zeigen, dass es zu den Anfangsbedingungen $u_0(x) = 0, x \leq 0$ und $u_0(x) = 1, x \geq 0$ zwei Lösungen gibt (eine kann über die Rankine-Hugoniot-Bedingungen gewonnen werden, die andere ist die klassische Lösung). Es stellt sich nun die Frage, welche Lösung unser Verfahren finden soll und auf welche Art dies zu erreichen

ist.
Wir betrachten die viskose Lösung $u^\epsilon(x,t)$, welche

$$\frac{\partial u^\epsilon}{\partial t} + \frac{\partial f(u^\epsilon)}{\partial x} = \epsilon \frac{\partial^2 u^\epsilon}{\partial x^2} \tag{3.4}$$

mit entsprechenden Anfangsbedingungen löst. Selbstverständlich ist die physikalische Lösung, die wir ja suchen, jene, die dem Grenzwert von ϵ gegen 0 entspricht. Ob dieser Grenzwert im allgemeinen existiert lässt sich nicht sagen.
Wir wollen nun die Entropiebedingung einführen (ausführlicher wird diese in [11] diskutiert). Wir definieren hierfür wieder eine konvexe Entropie $\eta(u)$ und den Entropiefluss

$$F(u) = \int_u \eta'(v) f'(v) dv.$$

Erfüllt unsere Lösung nun die Entropiebedingung

$$\frac{\partial \eta}{\partial t} + \frac{\partial}{\partial x} F(u) \leq 0 \tag{3.5}$$

so ist sie, falls der Fluss konvex ist, eindeutig. Um dies zu sehen muss lediglich das Differential nach x ausgeführt werden (Ableitung der Integralfunktion). Da der Integrand des Entropieflusses konvex ist, erhalten wir den gleichen Fall wie in Gleichung 3.1. Alle viskosen Lösungen erfüllen auch die Entropiebedingung, welche im Falle glatter Lösungen zu einer Gleichung wird. Weiterhin - unter der Voraussetzung eines konvexen Flusses - gilt: eine schwache Lösung, die die Entropiebedingung erfüllt, ist eindeutig [25].
Eine weitere Entropiebedingung, die sogenannte Lax-Entropiebedingung, lautet ([16]):

Satz 3.1.1. *Sei $u(x,t)$ eine schwache Lösung, S eine (x,t)-Kurve entlang der u eine Unstetigkeit hat. Sei $(x_0, u_0) \in S$ und u^- linker bzw. u^+ rechter Grenzwert von $u(x,t)$. Definiere*

$$s = \frac{f(u^-) - f(u^+)}{u^- - u^+}$$

Dieses s ist identisch zur Rankine-Hugoniot-Bedingung. u erfüllt die Entropie-Bedingung in $(x-0, t-0)$ genau dann, wenn

$$f'(u^-) \geq s \geq f'(u^+)$$

gilt. Die Unstetigkeit ist ein Schock und s die Schockgeschwindigkeit.

Wir müssen also darauf achten, wie wir die komplexeren nichtlinearen Flussterme approximieren, und uns bewusst sein, dass möglicherweise Glattheit verloren geht und damit auch die Eindeutigkeit, sofern wir nicht die Entropiebedingung durchsetzen.

3.2 Das nichtlineare Schema

Wir vertiefen nun das nichtlineare Schema, das bereits in Kapitel 2.2 angerissen wurde. Da der Weg von den lokalen Integralen hin zur vereinfachten Darstellung mittels der

Masse- und der Steifheitsmatrix bereits ausführlich diskutiert wurde und sich im nichtlinearen Fall nicht von den bisherigen Ausführungen unterscheidet, beginnen wir sofort mit der nichtlinearen starken bzw. schwachen Form

$$\boldsymbol{M}^k \frac{d}{dt}\boldsymbol{u}_h^k + \boldsymbol{S}\boldsymbol{f}_h^k = [\boldsymbol{l}^k(x)(f_h^k - f^*]_{x_l^k}^{x_r^k} \tag{3.6}$$

$$\boldsymbol{M}^k \frac{d}{dt}\boldsymbol{u}_h^k - \boldsymbol{S}^T\boldsymbol{f}_h^k = -[\boldsymbol{l}^k(x)f^*]_{x_l^k}^{x_r^k} \tag{3.7}$$

wobei unser Flux genauso definiert ist wie bei Gleichung 2.15. Wir erinnern uns an die Definition der auftretenden Vektoren als die Vektoren der nodalen Werte:

$$\boldsymbol{u}_h^k = [u_h^k(x_1^k), \ldots, u_h^k(x_{N_p}^k)]^T, \quad \boldsymbol{f}_h^k = [f_h^k(x_1^k), \ldots, f_h^k(x_{N_p}^k)]^T$$

Wir multiplizieren nun die schwache Form von rechts mit der konstanten Testfunktion $\phi = 1$, wodurch wir die Integrale ausführen können. Da die Ableitung durch die Steifheitsmatrix bei einer konstanten Testfunktion 0 ergibt, erhalten wir

$$\frac{d}{dt}\int_{x_l^k}^{x_r^k} u_h dx = f^*(x_l^k) - f^*(x_r^k)$$

Dies ist aber offenbar äquivalent zur Erhaltungseigenschaft bei unseren Erhaltungssätzen aus unserem vorherigen Abschnitt. Damit ist gezeigt, dass das Schema lokal konservativ ist.
Wenn wir nun noch über alle Zellen summieren, so sehen wir, dass bei der richtigen Wahl des Flux (konsistent und monoton, wie z.B. der Lax-Friedrichs-Flux) und periodischen Randbedingungen auch globale Erhaltung vorherrscht, da sich dann die inneren Fluxkomponenten aufheben.
Wir wollen nun untersuchen, welche Lösung unser Verfahren eigentlich liefert, und nutzen dafür nun eine allgemeine glatte Testfunktion der Form $\phi_h = \sum_{i=1}^{N_p} \phi(x_i^k, t) l_i^k(x)$. Diese nun anstelle der bisherigen Testfunktion in unsere schwache Form transponiert eingesetzt ergibt unter Ausnutzung der Skalarproduktnotation

$$\left(\phi_h, \frac{\partial}{\partial t}u_h\right)_D^k - \left(\frac{\partial \phi_h}{\partial x}, f_h\right)_D^k = -[\phi_h f^*]_{x_l^k}^{x_r^k}$$

woraus sich mit partieller Zeitintegration und Summation über alle Elemente leicht folgende Gleichung folgern lässt:

$$\begin{aligned} &\int_0^\infty \left[\left(\frac{\partial}{\partial t}\phi_h, u_h\right)_{\Omega,h} + \left(\frac{\partial \phi_h}{\partial x}, f_h\right)_{\Omega,h}\right] dt + (\phi_h(0), u_h(0))_{\Omega,h} \\ &= \int_0^\infty \sum_e \hat{\boldsymbol{n}}_e [[\phi_h(x_e^k) f^*(x_e^k)]] dt \end{aligned} \tag{3.8}$$

wobei e die Grenzflächen durchläuft. Als glatte Testfunktion konvergiert ϕ natürlich, wenn wir die Anzahl der Interpolationspunkte erhöhen, und mit dem gleichen Argument wie bei der konstanten Testfunktion können wir auch hier aussagen, dass die rechte Seite

der Gleichung verschwindet, wenn der Flux die erforderlichen Bedingungen erfüllt. Damit haben wir auch hier die Konservativität gezeigt. Unter der Voraussetzung, dass u_h fast überall gegen u konvergiert, können wir dann mit Abschnitt 2 von [2] auch folgern, dass es sich um eine schwache Lösung unseres Erhaltungssatzes handelt, und damit unter anderem sich der Schock mit der richtigen Geschwindigkeit bewegt.
Inspiriert von [12] wollen wir, entsprechend unserer Beobachtungen im vorigen Kapitel, versuchen, die Entropiebedingung einzuführen, und betrachten hierfür nun wieder Gleichung 3.7 und multiplizieren von links $\boldsymbol{u^T}$, um

$$\frac{1}{2}\frac{d}{dt}||u_h^k||_{D^k}^2 + \int_{D^k} u_h^k \frac{\partial}{\partial x} f_h^k dx = [u_h^k(x)(f_h^k - f^*)]_{x_l^k}^{x_r^k} \tag{3.9}$$

zu erhalten. Mit der konvexen quadratischen Entropiefunktion $\eta(u) = \frac{u^2}{2}$ und ihrem Entropiefluss $F'(u) = \eta' f'$, wobei offensichtlich gilt: $F(u) = f(u)u - \int_u f du = f(u)u - g(u)$, wobei g entsprechend definiert wurde. Damit erhalten wir für den zweiten Summanden von Gleichung 3.9:

$$\begin{aligned} &\int_{D^k} u_h^k \frac{\partial}{\partial x} f_h^k dx = \int_{D^k} \eta'(u_h^k) f'(u_h^k) \frac{\partial}{\partial x} u_h^k dx \\ &= \int_{D^k} F'(u_h^k) \frac{\partial}{\partial x} u_h^k dx = \int_{D^k} \frac{\partial}{\partial x} F(u_h^k) dx = \left[F(u_h^k)\right]_{x_l^k}^{x_r^k} \end{aligned} \tag{3.10}$$

Wir wollen auch für den nichtlinearen Fall Stabilität garantieren, was in Kapitel 2.2 noch nicht abgedeckt wurde. Eine hinreichende Voraussetzung hierfür wäre, wenn an jeder Grenzfläche

$$F(u_h^-) - F(u_h^+) - u_h^-(f_h^- - f^*) + u_h^+(f_h^+ - f^*) \geq 0$$

erfüllt ist. Mit der Definition unseres Entropieflusses folgt daraus

$$-g(u_h^-) + g(u_h^+) - f^*(u_h^+ - u_h^-) \geq 0.$$

Nach dem Mittelwerttheorem gilt für ein $\zeta \in [u_h^-, u_h^+]$:

$$g(u_h^+) - g(u_h^-) = g'(\zeta)(u_h^+ - u_h^-) = f(\zeta)(u_h^+ - u_h^-)$$

woraus folgt

$$(f(\zeta) - f^*)(u_h^+ - u_h^-) \geq 0.$$

Diese Bedingung ist von einem monotonen Flux, wie dem Lax-Friedrichs-Flux, den wir in Kapitel 2.2 ja mit dem Ziel der Monotonie konstruiert haben, erfüllt. Damit erhalten wir die nichtlineare Stabilität für unser Schema, ohne es in irgendeiner Weise modifizieren zu müssen.
[12] führt den konsistenten Entropieflux $\hat{F}(x) = f^*(x)u(x) - g(x)$ ein, was wir hier auch tun wollen. Mit ihm können wir die Zellentropiebedingung ausdrücken:

$$\frac{1}{2}\frac{d}{dt}||u_h^k||_{D^k}^2 - \int_{D^k} \frac{\partial u_h^k}{\partial x} f_h^k dx + [u_h^k f^*]_{x_l^k}^{x_r^k} = \frac{d}{dt}\int_{D^k} \eta(u_h^k) dx + \hat{F}(x_r^k) - \hat{F}(x_l^k) = 0$$

Diese ist nach Gleichung 3.10 zusammen mit der Glattheit der Lösung innerhalb der Zelle als strikte Gleichung erfüllt. Betrachten wir nun unser quadratische Entropie zusätzlich auch an einer Grenzfläche:

$$\frac{d}{dt}\int_{D^k}\eta(u_h^k)dx + \hat{F}(x_r^k) - \hat{F}(x_l^k) = \frac{d}{dt}\int_{D^k}\eta(u_h^k)dx + \hat{F}(x_r^k) - \hat{F}(x_r^{k-1}) + \hat{F}(x_r^{k-1}) - \hat{F}(x_l^k)$$
$$= \frac{d}{dt}\int_{D^k}\eta(u_h^k)dx + \hat{F}(x_r^k) - \hat{F}(x_r^{k-1}) + A^k = 0,$$

wobei (unter Ausnutzung von $f*(x_r^{k-1}) = f^*(x_l^k)$ und unseren Berechnungen von oben) gilt:

$$A^k = -f^*(x_r^{k-1})(u(x_r^{k-1}) - u(x_l^k)) + g(u_r^{k-1}) - g(u_l^k) = -(f^*(x_r^{k-1}) - f(\zeta))(u(x_r^{k-1}) - u(x_l^k)) \geq 0$$

Da wir nun auch die Grenzflächen berücksichtigt haben, ist somit die diskrete Zellentropiebedingung für die quadratische Entropie gezeigt:

$$\frac{d}{dt}\int_{D^k}\eta(u_h^k)dx + \hat{F}(x_r^k) - \hat{F}(x_r^{k-1}) \leq 0 \tag{3.11}$$

Aus ihr lässt sich auch die nichtlineare Stablität folgern (wie es in [12] getan wurde), aus Anschaulichkeitsgründen haben wir jedoch diese Reihenfolge gewählt.

3.3 Einführung zum Limiting

Unsere bisherigen Betrachtungen zu Stabilität und Eindeutigkeit der mit dem discontinuous Galerkin Verfahren gewonnenen numerischen Lösungen zu nichtlinearen Problem haben einen Punkt vernachlässigt: die Entwicklung von Unstetigkeiten und Schocks in der Lösung. Probleme beim Approximieren von Unstetigkeiten sind eine verbreitete Herausforderung, bekannt als Gibbs-Phänomen. Versucht man eine Unstetigkeit mit Polynomen darzustellen, so wird auch abseits von dieser die Genauigkeit punktweise auf erste Ordnung reduziert, punktweise Konvergenz geht verloren und um die Unstetigkeit herum treten Oszillationen auf.
Für unsere Euler-Gleichungen stellt dies aber ein enormes Problem da, denn bei Oszillationen, die nahe am Nullpunkt der Dichte oder des Drucks stattfinden, können diese negative Werte annehmen, was die Simulation zum Zusammenbruch führen kann. Demzufolge muss es unser Ziel sein, die Oszillationen zu unterbinden, ohne gleichzeitig die Genauigkeit der Berechnung zu stark negativ zu beeinflussen.
Hierzu werden wir unsere Methode um das Konzept des Limitings ergänzen, welches in [5] sehr gut eingeführt wird. Wir betrachten wieder die viskose Gleichung

$$\frac{\partial}{\partial t}u^\epsilon + \frac{\partial}{\partial x}f(u^\epsilon) = \epsilon\frac{\partial^2}{\partial x^2}u^\epsilon \tag{3.12}$$

und wollen, wie zuvor, eine eindeutige Lösung finden, wobei wir auch gleichmäßige Beschränktheit fordern, d.h. das Integral über den Betrag der Lösung im Raum muss

beschränkt sein. Letzteres gilt, sofern dies auch für die Anfangsbedingung gilt, wenn wir die Entropie als eben diesen Betrag der Lösung definieren. Mittels partieller Integration gilt folgender Zusammenhang:

$$-\int_\Omega (\eta'(u_x))_x u_t dx = \int_\Omega \frac{u_x}{|u_x|} u_{xt} dx = \frac{d}{dt}\int_\Omega |u_x| dx = \frac{d}{dt}||u_x||_{L^1}$$

Multiplizieren wir nun $-(\eta'(u_x^\epsilon))_x$ an unsere viskose Gleichung und führen eine Raumintegration aus, so erhalten wir damit:

$$\frac{d}{dt}||u_x^\epsilon||_{L^1} + \int_\Omega -(\eta'(u_x^\epsilon))_x f_x(u^\epsilon) dx = \int_\Omega -(\eta'(u_x^\epsilon))_x \epsilon u_{xx}^\epsilon dx.$$

Der zweite Term auf der linken Seite ist offensichtlich 0, was sich über partielle Integration ergibt und darüber, dass gilt: $\eta''(u)u = 0$. Für die rechte Seite sehen wir:

$$-\int_\Omega (\eta'(u_x^\eta))_x \epsilon u_{xx}^\epsilon dx = -\int_\Omega \epsilon \eta''(u_x^\epsilon)(u_{xx}^\epsilon)^2 dx \leq 0$$

Damit ist die gleichmäßige Beschränktheit gezeigt - allerdings nur für einen kontinuierlichen Fall und nicht für ein diskrete Methode. Wir wollen nun ein semidiskretes Schema betrachten, vorerst jedoch nur in erster Ordnung. Allgemein ist dieses von der Struktur:

$$h\frac{du_h^k}{dt} + f^*(u_h^k, u_h^{k+1}) - f^*(u_h^k, u_h^{k-1}) = 0 \tag{3.13}$$

Hierbei gehen wir von einem gleichmäßigen Zellabstand h aus und einem monotonen Flux. Den oben auftretenden Term $-(\eta'(u_x))_x$ nähern wir in erster Ordnung durch

$$v_h^k = -\frac{1}{h}\left[\eta'\left(\frac{u_h^{k+1} - u_h^k}{h}\right) - \eta'\left(\frac{u_h^k - u_h^{k-1}}{h}\right)\right]$$

Wenn wir nun mit diesem multiplizieren und über alle Zellen summieren, erhalten wir, unter Ausnutzung der Kettenregel und der Definition der diskreten totalen Variationsnorm

$$|u_h|_{TV} = \sum_{k=1}^{K} |u_h^{k+1} - u_h^k|$$

und dadurch, dass wir uns in Erinnerung rufen, dass die Entropie als Betrag der Lösung gewählt wurde, folgendes Ergebnis:

$$\frac{d}{dt}|u_h|_{TV} + \sum_{k=1}^{K} v_h^k (f^*(u_h^k, u_h^{k+1}) - f^*(u_h^k, u_h^{k-1})) = 0$$

Gehen wir nun wieder von unseren Lax-Friedrichs-Flux aus. Wir definieren

$$f^+(a) = \frac{1}{2}(f(a) + Ca), \quad f^-(b) = \frac{1}{2}(f(b) - Cb)$$

und erhalten so für $\hat{\boldsymbol{n}} = 1$ die Aufspaltung

$$f^*(a,b) = f^+(a) + f^-(b).$$

Beim anderen Vorzeichen der Normale vertauschen sich die Argumente. Wir können nun den Flux in steigende und fallende Komponenten spalten:

$$v_h^k(f^*(u_h^k, u_h^{k+1}) - f^*(u_h^k, u_h^{k-1})) = v_h^k(f^+(u_h^k) - f^+(u_h^{k-1}) + f^-(u_h^{k+1}) - f^-(u_h^k))$$

Über eine Fallunterscheidung lässt sich zeigen, dass dieser Term stets größer gleich 0 ist; daher folgt:

$$\frac{d}{dt}|u_h|_{TV} \leq 0$$

Die Eigenschaft, dass die Variation fallend ist, ist äquivalent zur gleichmäßigen Beschränktheit, welche wir zeigen wollten. Ähnliche Ergebnisse lassen sich selbstverständlich auch für andere monotone Fluxe erhalten.
Wenn wir dies aber nun auf höhere Ordnung erweitern, begegnen uns wieder die Oszillationen, wodurch die gerade gezeigte Beschränkung der totalen Varianz verletzt wird. Wir setzen nun zusätzlich voraus, dass in diesem Schema hoher Ordnung die Zellmittelwerte gleichmäßig beschränkt sind, und führen wieder die Notation u_l^k und u_r^k als linke bzw. rechte Grenzwerte der Lösung ein. Das Schema hat dann die Form

$$h\frac{d\bar{u}_h^k}{dt} + f^+(u_r^k, u_l^{k+1}) - f^*(u_l^k, u_r^{k-1}) = 0.$$

In [5] wird gezeigt, dass unter der Verwendung eines monotonen Fluxes und mit einem Eulerverfahren erster Ordung gilt

$$|\bar{u}^{n+1}|_{TV} - |\bar{u}^n|_{TV} + \Phi = 0,$$

wobei sich für Φ der komplexe Ausdruck

$$\begin{aligned}\Phi = &\sum_{k=1}^{K}\left(\eta'(\bar{u}^{k+1/2,n}) - \eta'(\bar{u}^{k+1/2,n}\right)(p(u^{k+1,n}) - p(u^{k,n})) \\ &+ \frac{\Delta t}{t}\sum_{k=1}^{K}\left(\eta'(\bar{u}^{k-1/2,n}) - \eta'(\bar{u}^{k+1/2,n})\right)(f^+(u_r^{k,n}) - f^+(u_r^{k-1,n})) \\ &- \frac{\Delta t}{t}\sum_{k=1}^{K}\left(\eta'(\bar{u}^{k+1/2,n}) - \eta'(\bar{u}^{k-1/2,n})\right)(f^-(u_l^{k+1,n}) - f^-(u_l^{k,n}))\end{aligned}$$

ergibt, wobei wir definiert haben: $\eta'(\bar{u}^{k+1/2,n}) = \eta'\left(\frac{\bar{u}^{k+1,n}-\bar{u}^{k,n}}{h}\right)$ und $p(u^{k,n}) = \bar{u}^k - \frac{\Delta t}{h}f^+(u_r^{k,n}) + \frac{\Delta t}{h}f^-(u_l^{k,n})$. Die Varianz des MIttelwerts ist genau dann abnehmend (total variation diminishing in mean - TVDM), wenn Φ größer gleich 0 ist. Da $\eta'(u) = \text{sign}(u)$

gilt, folgt damit durch einsetzen und Ausnutzen des Verhaltens unserer Fluxkomponenten:

$$\text{sign}(\bar{u}^{k+1,n} - \bar{u}^{k,n}) = \text{sign}(p(u^{k+1,n}) - p(u^{k,n})), \tag{3.14}$$
$$\text{sign}(\bar{u}^{k,n} - \bar{u}^{k-1,n}) = \text{sign}(u_r^{k,n} - u_r^{k-1,n}), \tag{3.15}$$
$$\text{sign}(\bar{u}^{k+1,n} - \bar{u}^{k,n}) = \text{sign}(u_l^{k+1,n} - u_l^{k,n}). \tag{3.16}$$

Diese Gleichungen sind im Allgemeinen jedoch nicht zwingend erfüllt. Um dennoch eine korrekte Lösung zu erhalten, müssen wir unser Verfahren um den Slope Limiter II erweitern. Dieser soll einerseits die Erhaltungsbedingungen erfüllen, die Genauigkeit nicht reduzieren und natürlich unsere obigen drei Bedingungen erfüllen. Die formale Genauigkeit unserer Methode lässt sich jedoch nur sehr schwer beibehalten. So kann es vorkommen, dass der Limiter lokale Extrema als Oszillationen klassifiziert und manipuliert, was die Genauigkeit stark reduziert (im Allgemeinen auf erste Ordnung). Ein Slope Limiter wirkt, im Gegensatz zu Flux Limitern, nicht auf unseren Flux, sondern auf unsere Lösung selbst.
Wir wollen nun einen simplen Slope Limiter konstruieren, der unsere Bedingungen erfüllt. Hierfür definieren wir die minmod-Funktion

$$m(a_1, \ldots, a_m) = \begin{cases} s \min_{1 \le i \le m} |a_i|, & |s| = 1 \\ 0, & \text{sonst} \end{cases} \qquad s = \frac{1}{m} \sum_{i=1}^{m} \text{sign}(a_i) \tag{3.17}$$

Diese Funktion prüft also, ob alle Argumente das gleiche Vorzeichen haben, und gibt in diesem Fall das betragsmäßig kleinste Argument aus. [10] implementiert dies auf folgende Weise. Dabei wird von der vektoriellen Struktur von Matlab Gebrauch gemacht, sodass laufzeitsparend gleich ein ganzer Datensatz manipuliert werden kann. Zuerst wird unsere Rückgabevariable mit 0 initiiert, dann jene Elemente ausgewählt, bei denen die Summe über den Argumentvektor 0 ist und mit dem Minimum im Vektor besetzt.

```
function mfunc =minmod(v)


m = size(v,1); mfunc =zeros(1,size(v,2));
s = sum(sign(v),1)/m;

ids =find(abs(s)==1);
if(~isempty(ids))
  mfunc(ids) =s(ids).*min(abs(v(:,ids)),[],1);
end
return;
```

Wir modifizieren unsere Randzustände an den Grenzflächen dann folgendermaßen:

$$v_l^k = \bar{u}_h^k - m(\bar{u}_h^k - u_l^k, \bar{u}_h^k - \bar{u}_h^{k-1}, \bar{u}_h^{k+1} - \bar{u}_h^k), \tag{3.18}$$
$$v_r^k = \bar{u}_h^k - m(\bar{u}_r^k - u_h^k, \bar{u}_h^k - \bar{u}_h^{k-1}, \bar{u}_h^{k+1} - \bar{u}_h^k), \tag{3.19}$$

Wenn wir nun diese anstatt u_r^k und u_l^k in unsere obigen Bedingungen einsetzen, so sehen wir, dass sie aufgrund des Verhaltens unserer minmod-Funktion erfüllt werden. Für die erste Gleichung muss man einen hinreichend kleinen Zeitschritt fordern. Bei den anderen sieht man, dass Einsetzen der limitierten Grenzflächenfluxe das Signum der linken Seite unserer Bedingung plus die Differenz zweier minmod-Funktionen (noch innerhalb der Signumfunktion) ergibt. Jene mit positivem Vorzeichen hat entweder in allen Argumenten das gleiche Vorzeichen wie unsere linke Seite oder ist 0, da die linke Seite in ihr vorkommt. Die mit negativem Vorzeichen hat ebenfalls die linke Seite in ihren Argumenten, kann also das Vorzeichen des Gesamtterms nicht ändern, da die minmod-Funktion ja nur das kleinste Argument ausgibt.
Man kann den Limiter noch etwas anders definieren, indem man die Lösung stückweise linear darstellt über

$$u_h^k(x) = \bar{u}_h^k + (x - x_0^k)(u_h^k)_x.$$

Hiermit kann man den MUSCL (Monotone Upstream-centered Scheme for Conservation Laws) Limiter definieren (siehe auch [26]):

$$\Pi^1 u_h^k(x) = \bar{u}_h^k + (x - x_0^k) m((u_h^k)_x, \frac{\bar{u}_h^{k+1} - \bar{u}_h^k}{h}, \frac{\bar{u}_h^k - \bar{u}_h^{k-1}}{h}) \tag{3.20}$$

Auch dieser erfüllt unsere Voraussetzungen für den Limiter, da es sich letztlich nur um eine Linearisierung des vorherigen Limiters handelt. Dieses lineare Limiting auf einer Zelle wird auf folgende Weise implementiert:

```
1  function ulimit =SlopeLimitLin(ul,xl,vm1,v0,vp1);
2
3  Globals1D;
4
5  ulimit =ul; h =xl(Np,:)-xl(1,:);
6  x0 = ones(Np,1)*(xl(1,:) +h/2);
7  hN = ones(Np,1)*h;
8
9  ux = (2./hN).*(Dr*ul);
10 ulimit =ones(Np,1)*v0+(xl-x0).*(ones(Np,1)*minmod([ux(1,:); (vp1-v0)./h;
       (v0-vm1)./h]));
11 return
```

Der u_x-Term in Zeile 9 wird dabei über unsere Differentiationsmatrix berechnet, die auch bei der Berechnung der Steifheitsmatrix verwendet wird. Um das Limiting auf alle Zellen anzuwenden, berechnen wir über alle Zellen den Mittelwert und übergeben diese dann unseren ***SlopeLimitLin***. Die Berechnung des Zellmittelwerts nutzt dabei die Orthogonalität der Basispolynome unseres Verfahrens aus. Wir nutzen dabei die modale

Darstellung unserer approximierten Lösung:

$$\bar{u}_h^k = \frac{1}{h^k}\int_{D^k} u_h(x)dx = \frac{1}{h^k}\int_{[-1,1]} u_h(r)\frac{d}{dr}(\psi^k(r))dx = \sum_{n=1}^{N_p} \hat{u}_n^k(t)\int_{[-1,1]} \phi_n(r)dx =$$
$$\sum_{n=1}^{N_p} \hat{u}_n^k(t)\int_{[-1,1]} \phi_n(r)\phi_1(r)dx = \hat{u}_1^k(t)$$

Wir haben dabei die Abbildungsfunktion ψ benutzt, die unsere Zelle auf den Standardintervall abbildet, ebenso wie den Fakt, dass das erste normierte Legendrepolynom ϕ_1 der konstante Wert 1 ist. Die Orthogonalität bedingt dann, dass unsere Mittelwert einfach der erste Koeffizient ist. Unsere Koeffizienten erhalten wir über die Multiplikation unseres Datenvektors mit der invertierten Vandermonde-Matrix. Die Linearisierung erfolgt auf ähnliche Weise.

```
function ulimit =SlopeLimit1(u);

Globals1D;
ulimit =zeros(Np,K);

uh = invV*u;

ul = uh; ul(3:Np) =0;ul =V*ul;

uh(2:Np,:)=0; uavg =V*uh; v =uavg(1,:);

vk = v; vkm1 =[v(1),v(1:K-1)]; vkp1 =[v(2:K),v(K)];

ulimit =SlopeLimitLin(ul,x,vkm1,vk,vkp1);
return
```

Dieser Limiter wird nun in jeder Zelle unserer Lösung aktiv. Wir wollen aber nur Oszillationen limitieren. Daher wollen wir eine weitere Modifikation vornehmen, um die Leistung unseres Limiters zu verbessern. Wir folgen dabei wieder dem Vorgehen von [5]. Wir gehen dabei davon aus dass unsere approximierte Lösung stückweise ein Polynom N-ter Ordnung ist. Daraufhin limitieren wir wieder nur die beiden Randwerte mit unserer Formel von oben. Wenn hierbei der gleiche Wert erhalten wird wie ohne Limiting, wird nichts weiter an der Lösung verändert. Ansonsten wird unser Π1-Limiter auf der ganzen Zelle angewendet.
In der Implementierung ändert sich dabei nur die letzte Zeile vor dem return-Statement. Sie wird ersetzt im Programm ***SlopeLimitN*** durch unser beschriebenes Verfahren ersetzt, welches folgende Form annimmt:

```
ve1 = vk - minmod([(vk-ue1);vk-vkm1;vkp1-vk]);
ve2 = vk + minmod([(ue2-vk);vk-vkm1;vkp1-vk]);
ids = find(abs(ve1-ue1)>eps0 | abs(ve2-ue2)>eps0);

```

```
5  if(~isempty(ids))
6
7  uhl = invV*u(:,ids); uhl(3:Np,:)=0; ul =V*uhl;
8
9  ulimit(:,ids) =SlopeLimitLin(ul,x(:,ids),vkm1(ids),vk(ids),
10 vkp1(ids));
11 end
```

Dieses Verfahren verbessert die Genauigkeit in glatten Gebieten der Lösung erheblich (siehe hierzu auch die numerischen Tests im übernächsten Abschnitt), aber scheitert weiterhin an lokalen Extrema. Hierfür wollen wir die Limiting-Bedingung abschwächen, indem wir unsere minmod-Funktion auf folgende Weise mittels einer Konstante M modifizieren (siehe auch [7],[6]):

$$\bar{m}(a_1, \ldots, a_m) = m(a_1, a_2 + Mh^2 \operatorname{sign}(a_2), \ldots, a_m + Mh^2 \operatorname{sign}(a_m)).$$

Dies entspricht der Forderung, dass unsere Varianz nicht mehr fallend ist, sondern lediglich beschränkt (TVBM-Bedingung). M sollte möglichst der obere Grenze der zweiten Ableitung an den Extrema entsprechen. Hierdurch kann erreicht werden, dass der Limiter auch beim Auftreten von lokalen Extrema nicht mehr auf erste Ordnung beschränkt ist.
Bisher haben wir nicht weiter über die Zeitintegration geredet, sondern sind stets bei unseren Herleitungen von einem Euler-Verfahren erster Ordnung ausgegangen. Seine Konvergenz erster Ordnung ist natürlich im Allgemeinen nicht akzeptabel. Da jedoch ein allgemeines explizites Runge-Kutta-Verfahren eine konvexe Kombination von Euler-Verfahren erster Ordnung ist, solange die Koeffizienten bei einem Runge-Kutta-Schritt positiv sind, können wir unsere Ergebnisse auch darauf verallgemeinern. Besonders geeignet für unsere Probleme sind die strong stability-perserving (SSP) Verfahren, welche auch im Fall von Unstetigkeiten keine zusätzlichen Oszillationen durch die Zeitintegration hervorrufen. Wir verwenden für unsere Untersuchung ein Verfahren dritter Ordnung aus [8]. Dieses hat drei Berechnungsstufen, wobei Limiting bei jeder dieser Stufen angewandt werden muss.

3.4 Umsetzung für die eindimensionalen Eulergleichungen

Im folgenden möchte ich den in [10] verwendeten Code aufführen, welcher die Ergebnisse der bisherigen Ausführungen beinhaltet und Grundlage für die Modifikationen im kommenden Kapitel ist. Im demonstrierten Code werden die eindimensionalen Eulergleichungen für das sogenannte Sod-Problem behandelt. Dieses Shock-Tube-Problem hat unstetige Anfangsbedingungen und enthält Schocks. Es lautet:

$$\rho(x,0) = \begin{cases} 1 & x \leq 0,5 \\ 0,125, & x \geq 0,5 \end{cases} \qquad \rho u(x,0) = 0 \quad E(x,0) = \frac{1}{\gamma - 1} \begin{cases} 1 & x \leq 0,5 \\ 0,1, & x \geq 0,5 \end{cases}$$

Der Vorteil dieses Problems ist die Bekanntheit einer exakten Lösung, was es vereinfacht, das Verhalten unserer approximierten Lösung zu betrachten.

Als erstes sei die Driverfunktion gegeben:

```
Globals1D;

N = 6;

[Nv, VX, K, EToV] =MeshGen1D(0.0, 1.0, 250);

StartUp1D;
gamma =1.4;

MassMatrix =inv(V')/V;
cx = ones(Np,1)*sum(MassMatrix*x,1);
rho = ones(Np,K).*( (cx<0.5) +0.125*(cx>=0.5));
rhou =zeros(Np,K);
Ener =ones(Np,K).*((cx<0.5) +0.1*(cx>=0.5))/(gamma-1.0);
FinalTime =0.2;

[rho,rhou,Ener] =Euler1D(rho,rhou,Ener,FinalTime);
```

Wie zu erwarten unterscheidet sich diese kaum von jener, die bereits im ersten Kapitel für unsere Advektionsgleichung gezeigt wurde. Dies unterstreicht erneut die Flexibilität unserer Methode. Lediglich die Anfangsbedingungen müssen nun für alle drei Größen eingegeben werden. $\boldsymbol{cx}$ enthält dabei den Ort in globalen Koordinaten unserer Auswertungspunkte. Auch die Zeitintegration hat sich nur wenig verändert:

```
function [rho,rhou,Ener] =Euler1D(rho, rhou, Ener, FinalTime)

Globals1D;

gamma =1.4; CFL =1.0; time =0;

mindx =min(x(2,:)-x(1,:));

rho =SlopeLimitN(rho); rhou=SlopeLimitN(rhou); Ener=SlopeLimitN(Ener);

while(time<FinalTime)
   Temp = (Ener -0.5*(rhou).^2./rho)./rho;
   cvel =sqrt(gamma*(gamma-1)*Temp);
   dt = CFL*min(min(mindx./(abs(rhou./rho)+cvel)));
   if(time+dt>FinalTime)
      dt = FinalTime-time;
   end
.
   [rhsrho,rhsrhou,rhsEner] =EulerRHS1D(rho, rhou, Ener);
   rho1 =rho + dt*rhsrho;
   rhou1 =rhou +dt*rhsrhou;
   Ener1 =Ener +dt*rhsEner;

```

```
    rho1 =SlopeLimitN(rho1); rhou1 =SlopeLimitN(rhou1);
    Ener1 =SlopeLimitN(Ener1);

    [rhsrho,rhsrhou,rhsEner] =EulerRHS1D(rho1, rhou1, Ener1);
    rho2 =(3*rho +rho1 +dt*rhsrho )/4;
    rhou2 =(3*rhou +rhou1 +dt*rhsrhou)/4;
    Ener2 =(3*Ener +Ener1 +dt*rhsEner)/4;

    rho2 =SlopeLimitN(rho2); rhou2 =SlopeLimitN(rhou2);
    Ener2 =SlopeLimitN(Ener2);

    [rhsrho,rhsrhou,rhsEner] =EulerRHS1D(rho2, rhou2, Ener2);
    rho =(rho +2*rho2 +2*dt*rhsrho )/3;
    rhou =(rhou +2*rhou2 +2*dt*rhsrhou)/3;
    Ener =(Ener +2*Ener2 +2*dt*rhsEner)/3;

    rho =SlopeLimitN(rho); rhou=SlopeLimitN(rhou);

    time =time+dt;
end
return
```

Wir sehen dass die Zeitintegration aller Größen unabhängig voneinander geschieht und nach jedem Schritt ein Limiting vorgenommen wird. Bei der CFL-Bedingung, die in Kapitel 1 ausführlich besprochen wurde, ist die Konstante auf 1 festgelegt. Wir haben wieder ein stabilitätserhaltendes Runge-Kutta-Verfahren dritter Ordnung gewählt. Wie erwähnt haben wir vor jedem RK-Schritt unseren Limiter eingebaut, da das Sod-Problem, wie in den Anfangsdaten ersichtlich, Unstetigkeiten enthält. Wir limitieren dabei jede Größe einzeln. Tatsächliche Änderungen in unserem Verfahren sieht man erst im letzten Programmmodul:

```
function [rhsrho, rhsrhou, rhsEner] =EulerRHS1D(rho, rhou ,Ener)

Globals1D;

gamma =1.4;
pres =(gamma-1.0)*(Ener -0.5*(rhou).^2./rho);
cvel =sqrt(gamma*pres./rho); lm =abs(rhou./rho)+cvel;

rhof =rhou; rhouf=rhou.^2./rho+pres; Enerf=(Ener+pres).*rhou./rho;

drho =zeros(Nfp*Nfaces,K); drho(:) =rho(vmapM)- rho(vmapP);
drhou =zeros(Nfp*Nfaces,K); drhou(:) =rhou(vmapM)- rhou(vmapP);
dEner =zeros(Nfp*Nfaces,K); dEner(:) =Ener(vmapM)- Ener(vmapP);
drhof =zeros(Nfp*Nfaces,K); drhof(:) =rhof(vmapM)- rhof(vmapP);
drhouf=zeros(Nfp*Nfaces,K); drhouf(:) =rhouf(vmapM)-rhouf(vmapP);
dEnerf=zeros(Nfp*Nfaces,K); dEnerf(:) =Enerf(vmapM)-Enerf(vmapP);
LFc =zeros(Nfp*Nfaces,K); LFc(:) =max(lm(vmapP),lm(vmapM));
```

```
18
19 drhof(:) =nx(:).*drhof(:)/2.0-LFc(:)/2.0.*drho(:);
20 drhouf(:)=nx(:).*drhouf(:)/2.0-LFc(:)/2.0.*drhou(:);
21 dEnerf(:)=nx(:).*dEnerf(:)/2.0-LFc(:)/2.0.*dEner(:);
22
23 rhoin =1.000; rhouin =0.0;
24 pin =1.000; Enerin =pin/(gamma-1.0);
25 rhoout =0.125; rhouout =0.0;
26 pout =0.100; Enerout =pout/(gamma-1.0);
27
28 rhofin =rhouin; rhoufin=rhouin.^2./rhoin+pin;
29 Enerfin=(pin/(gamma-1.0)+0.5*rhouin^2/rhoin+pin).*rhouin./rhoin;
30 lmI=lm(vmapI)/2; nxI=nx(mapI);
31 drho (mapI)=nxI*(rhof (vmapI)-rhofin )/2.0-lmI*(rho(vmapI) -rhoin);
32 drhou(mapI)=nxI*(rhouf(vmapI)-rhoufin)/2.0-lmI*(rhou(vmapI)-rhouin);
33 dEner(mapI)=nxI*(Enerf(vmapI)-Enerfin)/2.0-lmI*(Ener(vmapI)-Enerin);
34 rhofout=rhouout; rhoufout=rhouout.^2./rhoout+pout;
35 Enerfout=(pout/(gamma-1.0)+0.5*rhouout^2/rhoout+pout).*rhouout./rhoout;
36 lm0=lm(vmap0)/2; nx0=nx(map0);
37 drho (map0)=nx0*(rhof(vmap0) -rhofout)/2.0-lm0*(rho (vmap0)- rhoout);
38 drhou(map0)=nx0*(rhouf(vmap0)-rhoufout)/2.0-lm0*(rhou(vmap0)-rhouout);
39 dEner(map0)=nx0*(Enerf(vmap0)-Enerfout)/2.0-lm0*(Ener(vmap0)-Enerout);
40
41 rhsrho =-rx.*(Dr*rhof) +LIFT*(Fscale.*drhof);
42 rhsrhou =-rx.*(Dr*rhouf) +LIFT*(Fscale.*drhouf);
43 rhsEner =-rx.*(Dr*Enerf) +LIFT*(Fscale.*dEnerf);
44 return
```

Hier muss nicht nur die Randbedingung für das Sod-Problem implementiert werden (Zeile 23-26). Es benötigt auch die Berechnung der maximalen Geschwindigkeit sowie des Drucks für den Lax-Friedrichs-Flux (Zeile 6 und 7). Während der Druck ja bereits in der Differentialgleichung auftritt, benötigen wir die maximale Geschwindigkeit, da diese der größte Eigenwert unserer Jacobi-Matrix ist. Darüber hinaus muss der Flux für jede der drei Größen entsprechend der partiellen Differentialgleichungen berechnet werden, sowohl zwischen den Zellen als auch am Rand. Dies erfordert umfangreiche Rechenschritte. Am Ende kann dann jedoch ganz analog zur Advektionsgleichung die rechte Seite unserer Zeitintegration berechnet werden.

3.5 Anwendung

Wir wollen nun die Leistung unseres Verfahrens im Falle der Eulergleichungen sowie unsere verschiedenen Limiter auch praktisch an Testproblemen demonstrieren und dabei Vorzüge wie auch Nachteile derselben anhand der numerischen Tests herausarbeiten. In einem ersten Schritt wollen wir zeigen, dass in allen drei konservativen Variablen die optimale Konvergenzrate durch unser Verfahren erreicht wird. Hierzu können wir nicht das oben aufgeführte Sod-Shock-Tube-Problem verwenden, denn dies enthält of-

fensichtlich Unstetigkeiten, welche ja bei einer polynomialen Interpolation wie in unserer DG-Methode die Genauigkeit grundsätzlich auf erste Ordnung reduzieren. Stattdessen wollen wir ein Plain-Wave-Problem nutzen. Hierzu seien die Anfangsbedingungen:

$$\rho(x,0) = 1 + \frac{1}{2}\sin(2\pi x) \quad \rho u(x,0) = \rho(x,0) \quad E(x,0) = \frac{1}{\gamma - 1} + \frac{1}{2}\rho$$

Dieses Problem ist offensichtlich glatt und benötigt kein Limiting. Wir erwarten in allen Variablen dieselbe Konvergenzordnung des $L2$-Fehlers wie zuvor bei unserem Advektionsproblem, d.h. eine Konvergenzordnung von $N+1$. In den folgenden Tabellen sind für verschiedene Approximationsordnungen der L^2-Fehler zwischen den Lösungen mit einer Auflösung von K=25 und K=50 sowie für K=50 und K=100 aufgeführt und die daraus berechnete Konvergenzrate eingetragen. Dabei ist für jede Größe eine eigene Tabelle aufgeführt.

Tab. 3.1: Konvergenzraten für das Plain-Wave-Problem, Dichte

N	L^2-Fehler zw. K=25 und K=50	L^2-Fehler zw. K=50 und K=100	Konvergenzrate
1	0.0017376	4.2476e-004	2.0324
2	1.1760e-004	1.5614e-005	2.9130
3	6.8850e-007	5.5283e-008	3.6385
4	1.8082e-008	5.8288e-010	4.9552

Tab. 3.2: Konvergenzraten für das Plain-Wave-Problem, Impuls

N	L^2-Fehler zw. K=25 und K=50	L^2-Fehler zw. K=50 und K=100	Konvergenzrate
1	0.0017376	4.2476e-004	2.0324
2	1.1760e-004	1.5614e-005	2.9130
3	6.8850e-007	5.5283e-008	3.6385
4	1.8082e-008	5.8288e-010	4.9552

Tab. 3.3: Konvergenzraten für das Plain-Wave-Problem, Energie

N	L^2-Fehler zw. K=25 und K=50	L^2-Fehler zw. K=50 und K=100	Konvergenzrate
1	8.6882-004	2.1238e-004	2.0324
2	5.8801e-005	7.8068e-006	2.9130
3	3.4425e-007	2.7642e-008	3.6385
4	9.0411e-009	2.9144e-010	4.9552

Dies zeigt auf eindrückliche Weise, dass unser Konvergenzverhalten auch für das komplexere System der Eulergleichungen erhalten bleibt. Die Fehler von ρ und ρu sind aufgrund der gleichen Anfangsbedingungen wie zu erwarten identisch. Wir sehen bei N=3

eine leicht suboptimale Konvergenzrate, welche nahelegt, dass für nichtlineare Polynominterpolationen ungerader Ordnung ein etwas schlechteres Verhalten vorliegt. Eine Untersuchung dessen würde hier zu weit führen.
Als nächstes wollen wir unsere beiden TVD-Limiter, ***SlopeLimit*1** und ***SlopeLimitN*** vergleichen. Letzterer ist eine modifizierte Version des ersten Limiters, bei dem das Limiting nicht überall vorgenommen wird, sondern nur in Gebieten, in denen Oszillationen festgestellt werden, sodass das Verhalten in glatten Gebieten der Lösung deutlich verbessert werden sollte. Wir wollen diese Annahme an einem konkreten Beispiel bestätigen. Wir wählen hierzu erneut unser Plain-Wave-Problem. Ein guter Limiter sollte das Ergebnis möglichst wenig beeinflussen, da hier ja kein Limiting erforderlich ist. Um Abweichungen besonders gut sichtbar zu machen, wählen wir eine lineare Approximation mit niedriger Zellauflösung.

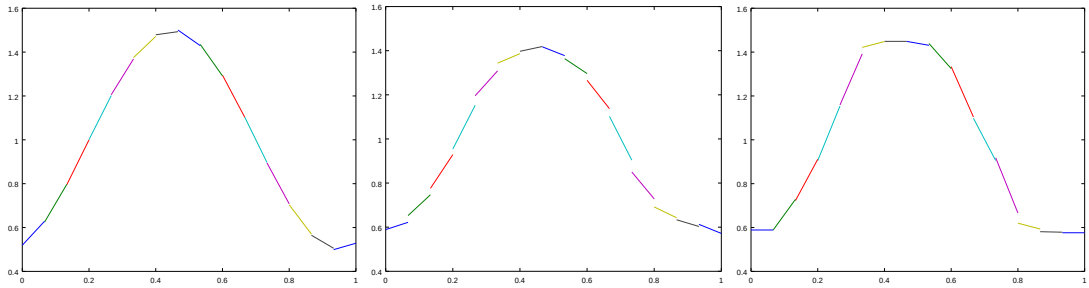


Abb. 3.1: Plain-Wave-Problem, N=1, $t = 0.2$, K=15; v.l.n.r. : unlimitiert, ***SlopeLimit*1**, ***SlopeLimitN***

Wir erkennen sehr deutlich die Vorzüge des verbesserten TVD-Limiters. in Abbildung 3.1 Nicht nur reicht die Konvergenzrate deutlich näher an die natürliche von grob 2 heran (1.3 bei ***SlopeLimit*1** vs. 1.6 bei ***SlopeLimitN*** - da der Limiter unser Polynom stückweise linearisiert, kann bis zu zweite Ordnung erreicht werden). Dies zeigt, dass die Verbesserung den Limiter deutlich weniger aggressiv macht. An den Abbildungen sehen wir auch, weshalb. Während zuvor auf dem gesamten Bereich der Lösung Abweichungen durch Limiting zu erkennen waren, insbesondere auch in monoton steigenden Gebieten, so wird durch den verbesserten Limiter lediglich im Bereich der Maxima und Minima eingegriffen.
Wir haben also bereits einen großen Fortschritt erreicht. Eine hohe Konvergenzordnung ist so jedoch noch nicht möglich. Hierfür müssen wir den Limiter weiter schwächen. Wir hatten dafür die Bedingung der verschwindenden Variation gelockert und lediglich ihre Beschränktheit erfordert. Dies führte zu unserem TVB-Limiter. Wir wollen zuerst am Plain-Wave-Problem zeigen, dass dieser zu besseren Ergebnissen und höherer Genauigkeit führt.
Bereits rein optisch erkennen wir enormen Verbesserung in Abbildung 3.2. Die Extrema zeigen eine deutlich verbesserte Auflösung, die deutlich mehr der unlimiterten Lösung entsprechen, obwohl wir mit M=20 noch relativ konservativ vorgegangen sind. Wir wollen jedoch die höhere Genauigkeit nicht nur optisch, sondern auch mit konkre-

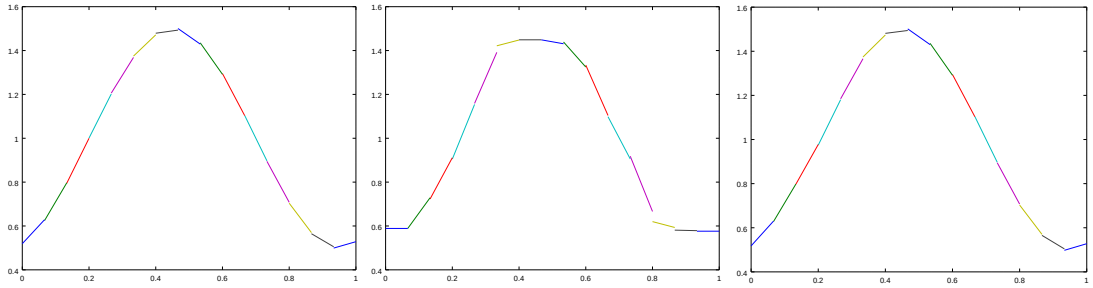


Abb. 3.2: Plain-Wave-Problem, N=1, t = 0.2, K=15; v.l.n.r. : unlimitiert, ***SlopeLimitN***, TVB-Limiter mit M=20 basierend auf ***SlopeLimitN***

ten Messdaten beweisen. Wir werden hierzu die L^2-Konvergenzrate des TVD-Limiters mit der des TVB-Limiters für verschiedenen Polynomapproximationsordnungen und verschiedene M vergleichen. Wir beschränken uns auf die Konvergenzrate der Dichte, da sich gezeigt hat, dass alle Konvergenzraten stark korreliert sind. Die Konvergenzraten werden mithilfe der exakten Lösung berechnet.

Tab. 3.4: Konvergenzraten für das Plain-Wave-Problem, Dichte, Vergleich TVD und TVB

N	TVD	TVB M=10	TVB M=25	TVB M=50	TVB M=75	TVB M=300
1	1.2585	1.2384	1.3096	1.4765	1.6169	2.0368
2	1.2169	1.2243	1.2909	1.5347	2.0312	2.9298
3	1.3139	1.4296	1.4743	2.3241	3.2798	3.6266

Die Daten in Tabelle 3.4 zeigen gut die Verbesserungen durch den TVB-Limiter. Sein Verhalten für niedrige M entspricht sehr stark dem des TVD-Limiters. Dies liegt daran begründet, dass der TVD-Limiter letztlich auch ein spezieller TVB-Limiter ist für M=0. Bei hohem M hingegen wird der Limiter immer seltener getriggert, da dies nur geschieht, wenn die Variation hinreichend groß ist. Für sehr große M wird also nie der Limiter ausgelöst. Dadurch kann für sehr hohe M der unlimitierte Fall mit maximaler Konvergenzordnung erreicht werden. Dies ist jedoch nur bei glatten Lösungen der Fall. Wie gezeigt wurde ist bei unstetigen Fällen Limiting notwendig. Daraus folgt, dass bei einer zu hohen Werten von M die Simulation zusammenbricht, was die maximale Genauigkeit begrenzt.
Wir wollen aber jedoch auch zeigen, dass der limitierende Effekt des TVB-Limiters vergleichbar ist zu dem des reinen TVD-Limiters. Hierzu kehren wir zum Sod-Shock-Tube-Problem zurück, welches wir bereits bei der Demonstration des Basiscodes für unsere eindimensionalen Eulergleichungen erwähnt haben. Da dieses Unstetigkeiten vorweist, ist hier Limiting notwendig. Daraus folgt, dass die Stabilität der Simulation beim TVB-Limiter zusammen bricht, wenn er nicht mehr aggressiv genug ist. Dies ist bei N=2 und K=200 für ein M ab ungefähr 700 der Fall. Eine maximale Genauigkeit ist also nicht nur wegen der Unstetigkeit unmöglich, sondern auch wegen der Notwendigkeit eines hin-

reichend aggressiven Limitings.

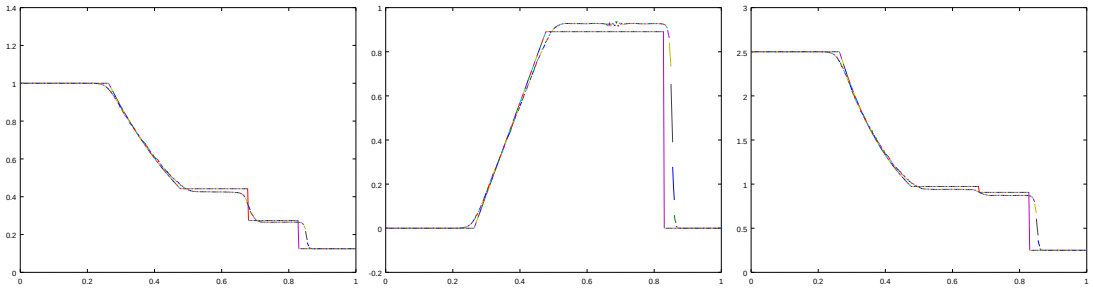


Abb. 3.3: Sod-Shock-Tube-Problem, N=2, $t = 0.2$, K=200; TVD; v.l.n.r. : Dichte, Geschwindigkeit, Energie; jeweils hinterlegt mit der analytischen Lösung

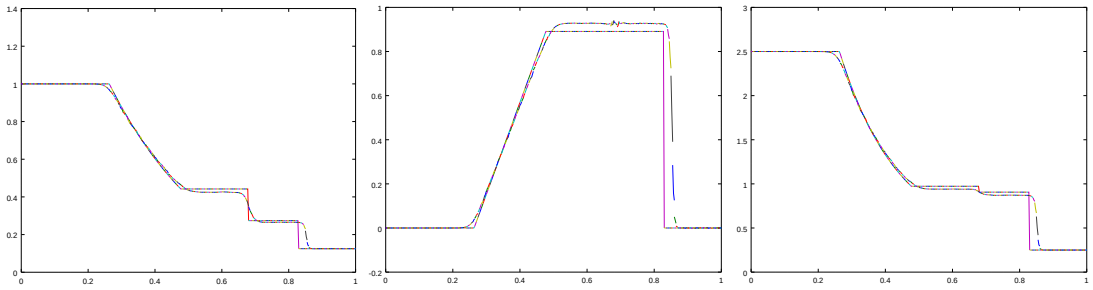


Abb. 3.4: Sod-Shock-Tube-Problem, N=2, $t = 0.2$, K=200; TVB mit M=20; v.l.n.r. : Dichte, Geschwindigkeit, Energie; jeweils hinterlegt mit der analytischen Lösung

Wir sehen in Abbildung 3.3 und Abbildung 3.4, dass der TVB-Limiter gleichwertige Ergebnisse liefert wie der TVD-Limiter. Es ergibt sich eine stabile Lösung mit recht hoher Genauigkeit, auch gerade an Stellen mit Unstetigkeiten. Wir beobachten bei beiden jedoch eine Verschiebung der Unstetigkeiten nach rechts, deren genaue Ursache noch unklar ist. Die Vorzüge des TVB-Limiters, nämlich die deutlich verbesserte Auflösung von Extrema, treten hier natürlich nicht zu Tage, im Gegensatz zum letzten Beispiel. Stattdessen sehen wir, dass diese nicht auf Kosten der Funktionalitäten eines TVD-Limiters gehen. Diese Aussage stimmt jedoch nur unter der Einschränkung, dass M nicht zu groß gewählt wird. Wie wir in Abbildung 3.5 sehen, ist der Nachteil von nur beschränkter Variation, dass Oszillationen immer noch auftreten können. Diese vermindern nicht nur die Genauigkeit. Sie sind auch unphysikalisch, und können nahe am Nullpunkt zu negativer Dichte oder Druck führen und damit, wie erwähnt, bei zu schwachem TVB-Limiter die Simulation unbrauchbar machen.
Desweiteren stellt sich das Problem, wie man jene Oszillationen, die in einer Lösung auftreten sollen, von solchen unterscheidet, die durch das TVB-Limiting entstehen. Ziehen wir also ein Situation in Betracht, bei der wir ein glattes Gebiet mit Oszillationen kleiner Amplitude erwarten, ebenso jedoch auch eine Unstetigkeit. Ein strenger TVB-Limiter würde diese unterdrücken. Bei einem schwachen hingegen würden durch die Approxima-

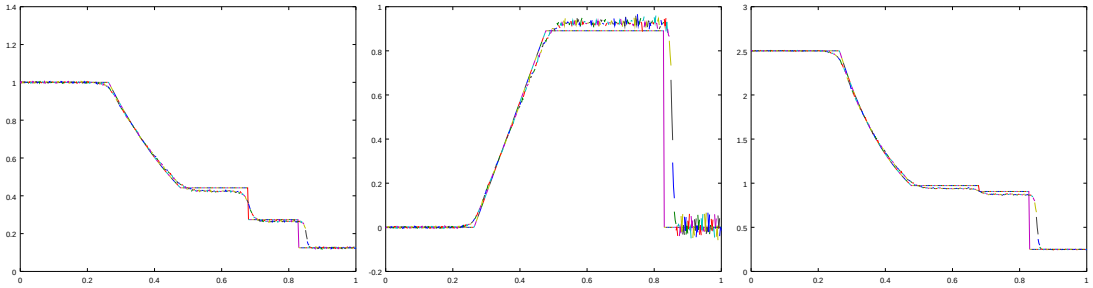


Abb. 3.5: Sod-Shock-Tube-Problem, N=2, $t = 0.2$, K=200; TVB mit M=300; v.l.n.r. : Dichte, Geschwindigkeit, Energie; jeweils hinterlegt mit der analytischen Lösung

tion der Unstetigkeit Oszillationen auftreten, die die physikalischen überlagern.
Ein Problem mit einer solchen Konfiguration ist das Shu-Osher-Problem (Beispiel 8 von [24]).

$$\rho(x,0) = \begin{cases} 3.857143 & x \leq -4 \\ 1 + 0.1\sin(5x), & x \geq -4 \end{cases} \qquad \rho u(x,0) = 0 \begin{cases} 2.629369\rho & x \leq -4 \\ 0 & x \geq -4 \end{cases}$$

$$E(x,0) = \frac{(\rho u)^2}{2\rho} \frac{1}{\gamma - 1} \begin{cases} 10.3333 & x \leq -4 \\ 1, & x \geq -4 \end{cases}$$

Mit unseren bisherigen Mitteln können wir dieses Problem nicht zufriedenstellend lösen. Ein optischer Vergleich in Abbildung 3.6 zeigt beim TVD-Limiter eine entscheidende Problematik. Während die breiten Oszillationen zu am linken und rechten Rand unsere Definitionsbereichs relativ gut abgebildet werden, sind die deutlich feineren, unregelmäßigen Oszillationen im Gebiet [0,2] der exakten Lösung nicht aufgelöst und verschwinden hinter einem Maximum. Unabhängig vom quantitativen Fehler ist dieses Verhalten qualitativ sehr schlecht, wenn eben gerade diese feinen Oszillationen untersucht werden sollen. Wir wollen sehen, ob unser TVB-Limiter ein besseres Verhalten zeigt.

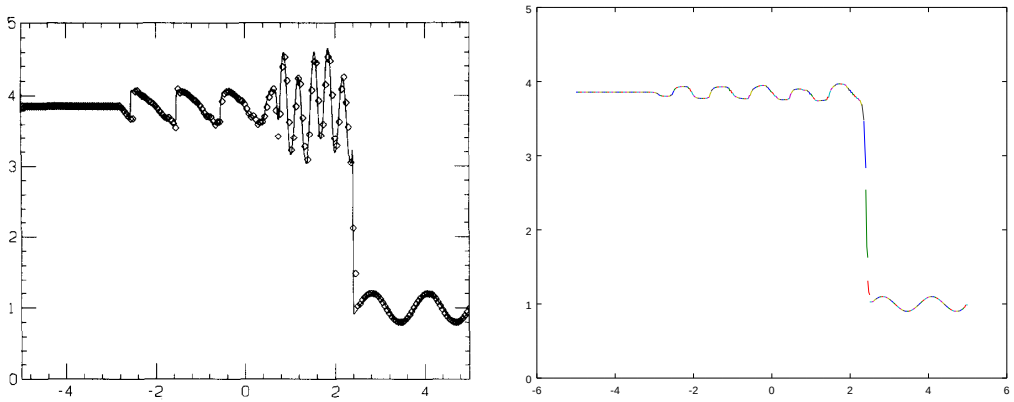


Abb. 3.6: Shu-Osher-Problem, N=2, $t = 1.8$, K=200; Dichte; v.l.n.r. : exakte Lösung aus [24], S. 109, überlagert mit einem anderen Verfahren; Ergebnis des TVD-Limiters

Doch auch Abbildung 3.7 zeigt kein zufriedenstellendes Verhalten. Im erwähnten Gebiet lassen sich keine feineren Strukturen erkennen. Zudem sind hier, wie auf der gesamten Lösung links der Unstetigkeitsstelle, sehr rasch feine Oszillationen, die von der Restvariation herrühren, die der TVB-Limiter erlaubt. Noch ehe also die physikalischen Oszillationen erhalten bleiben und nicht weglimitiert werden, tauchen neue, unphysikalische auf. Tatsächlich können wir mit dem bisherigen TVB-Limiter nicht den Punkt erreichen, an dem sie erhalten bleiben, da ab M=40 die Lösung nicht mehr stabil ist. Damit scheitern alle bisher erarbeiteten Werkzeuge daran, eine qualitativ hochwertige Lösung zu erzeugen. Desweiteren konnte noch nicht geklärt werden, wie wir sicherstellen, dass trotz lokaler Schwankungen bei Verwendung des TVB-Limiters zum Erreichen höherer Genauigkeit verhindert werden kann, dass die Lösung instabil wird, oder zumindest der instabile Bereich verringert werden kann. Diesen beiden Problemen wollen wir uns im nächsten Kapitel widmen.

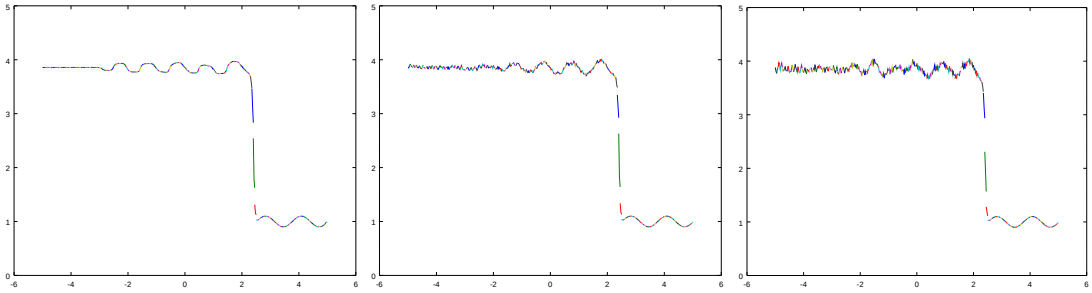


Abb. 3.7: Shu-Osher-Problem, N=2, t = 1.8, K=200; Dichte; v.l.n.r. : TVB mit M=5; M=15;M=30

4 Fortgeschrittenes Limiting

In diesem Kapitel widmen wir uns zwei Methoden, die die Leisungsfähigkeit unserer DG-Methode zusätzlich erweitern werden. Während wir mit Limting in charakteristischen Variablen das Shu-Osher-Problem deutlich besser lösen können werden, soll im zweiten Teil dieses Kapitels ein Positivitätslimiter eingeführt werden, der die physikalischen Anforderungen einer Lösung respektiert und damit die Stabilität verbessert, was letztlich auch höhere Genauigkeit ermöglicht.

4.1 Limiting in charakteristischen Variablen

Motiviert von Remark 2 in [14] sowie den Ausführungen in [7] und [6] wollen wir das Limiting in den charakteristischen Variablen vornehmen, im Gegensatz zu den Erhaltungsvariablen, die wir bisher verwendet haben. Hintergrund ist, dass sich gezeigt hat, dass hierdurch deutlich weniger lokale numerische Oszillationen in die Lösung eingeführt werden. Dies verbessert sowohl die Stabilität der Lösung als auch die Qualität.
Dazu gehen wir wie folgt vor: Wir suchen die Matrix R und ihre Inverse, welche die Matrix $J = \frac{\partial}{\partial u} f(\bar{u}_j)$ diagonalisiert, d.h. $R^{-1}JR = \Lambda$. Das bedeutet selbstverständlich, dass in R die Eigenvektoren von J und in Λ die Eigenwerte stehen. Die reelle Diagonalisierbarkeit ist garantiert, weil unsere Eulergleichungen hyperbolische partielle Differentialgleichungen sind.
Unser Limiting in den charakteristischen Variablen läuft nun derart ab, dass wir zuerst unsere Erhaltungsvariablen mittels R^{-1} in den Eigenraum projezieren, unseren TVD bzw. TVB-Limiter anwenden und dann wieder zurück auf unsere Erhaltungsvariablen mittels R abbilden.
Wie in den erwähnten Quellen ausgeführt wurde, hat sich gezeigt, dass das Limiting in den charakteristischen Variablen die auftretenden Oszillationen deutlich stärker reduziert. Die Matrizen, die wir für die Projektion anwenden, lauten:

$$R^{-1} = \begin{pmatrix} 1 & 1 & 1 \\ \bar{u}_h - \bar{c}_h & \bar{u}_h & \bar{u}_h + \bar{c}_h \\ \bar{H}_h - \bar{u}_h\bar{c}_h & \frac{1}{2}\bar{u}_h^2 & \bar{H}_h + \bar{u}_h\bar{c}_h \end{pmatrix}$$

$$R = \begin{pmatrix} \frac{\bar{m}_h}{2\bar{c}_h\bar{\rho}_h} + \frac{(\gamma-1)\bar{m}_h^2}{4\bar{c}_h^2\bar{\rho}^2} & -\frac{\bar{m}_h}{2\bar{c}_h\bar{\rho}_h} - \frac{(\gamma-1)\bar{m}_h}{2\bar{c}_h^2\bar{\rho}} & \frac{(\gamma-1)}{2\bar{c}_h^2} \\ 1 - \frac{(\gamma-1)\bar{m}_h^2}{2\bar{c}_h^2\bar{\rho}_h^2} & \frac{(\gamma-1)\bar{m}_h}{\bar{c}_h^2\bar{\rho}_h} & -\frac{(\gamma-1)}{\bar{c}_h^2} \\ -\frac{\bar{m}_h}{2\bar{c}_h\bar{\rho}_h} + \frac{(\gamma-1)\bar{m}_h^2}{4\bar{c}_h^2\bar{\rho}^2} & \frac{\bar{m}_h}{2\bar{c}_h\bar{\rho}_h} - \frac{(\gamma-1)\bar{m}_h}{2\bar{c}_h^2\bar{\rho}} & \frac{(\gamma-1)}{2\bar{c}_h^2} \end{pmatrix}$$

Mit

$$\bar{c}_h = \sqrt{\frac{\gamma \bar{p}_h}{\bar{\rho}_h}}, \quad \bar{H}_h = \frac{\bar{E}_h + \bar{p}_h}{\bar{\rho}_h} = \frac{\bar{c}_h^2}{\gamma - 1} + \frac{1}{2}\bar{u}_h^2$$

Diese lassen sich über Differenzierung der Flussmatrix errechnen. In der Implementierung bedeutet dies, dass wir die Projektion vor und nach ***SlopeLimitN*** anwenden müssen. Hierzu muss unser Limiter zuerst so modifiziert werden, dass er alle drei Größen gleichzeitig limitert, da diese nun miteinander verrechnet werden. Darüber hinaus muss die erste Projektion am Anfang des Limitierungsprozess durchgeführt werden, was dieser Programmteil übernimmt:

```
rhoh =invV*rho; rhoh(2:Np,:)=0; rhoavg =V*rhoh; vrho =rhoavg(1,:);
rhouh =invV*rhou; rhouh(2:Np,:)=0; rhouavg =V*rhouh; vrhou =rhouavg(1,:);
Enerh =invV*Ener; Enerh(2:Np,:)=0; Eneravg =V*Enerh; vEner =Eneravg(1,:);

ph=(gamma-1.0)*(vEner -0.5*((vrhou).^2)./vrho);
ch=sqrt(gamma*ph./vrho);
Hh=(vEner+ph)./vrho;
uvh=vrhou./vrho;

aa=((gamma-1)/4)*((uvh.^2)./(ch.^2))+(uvh./(2*ch));
ab=-((gamma-1)./(2*ch.^2)).*uvh-(1)./(2*ch);
ac=(gamma-1)./(2*(ch.^2));
ba=1-((gamma-1)./(2*ch.^2)).*(uvh.^2);
bb=(gamma-1)*uvh./(ch.^2);
bc=-2*ac;
ca=((gamma-1)/4)*((uvh.^2)./ch.^2)-uvh./(2*ch);
cb=-((gamma-1)./(2*ch.^2)).*(uvh)+(1)./(2*ch);
cc=ac;

for i=1:Np
  rhon(i,:)=aa.*rho(i,:)+ab.*rhou(i,:)+ac.*Ener(i,:);
  rhoun(i,:)=ba.*rho(i,:)+bb.*rhou(i,:)+bc.*Ener(i,:);
  Enern(i,:)=ca.*rho(i,:)+cb.*rhou(i,:)+cc.*Ener(i,:);
end
```

Dabei wird erst wieder der Zelltmittelwert gebildet und dann die Einträge unserer Matrix berechnet. Dabei ist zu beachten, dass die Mittelwerte Vektoren der Länge der Anzahl unserer Zellen sind; damit sind auch die Einträge der Matrix Vektoren. Zuletzt erfolgt die Abbildung auf unsere charakteristischen Variablen punktweise. Da wir mit Mittelwerten arbeiten, sind die Matrixeinträge für alle Punkte einer Zelle identisch.
Die Rückabbildung in unseren Raum der Erhaltungsvariablen findet am Ende unseres Limitingprozesses statt und funktioniert ganz ähnlich:

```
maa=mab=mac=ones(1,K);
mba=uvh-ch;
mbb=uvh;
mbc=uvh+ch;
```

```
mca=Hh-uvh.*ch;
mcb=0.5*uvh.^2;
mcc=Hh+uvh.*ch;

for i=1:Np

   rhos(i,:)=maa.*rho1(i,:)+mab.*rhou1(i,:)+mac.*Ener1(i,:);
  rhous(i,:)=mba.*rho1(i,:)+mbb.*rhou1(i,:)+mbc.*Ener1(i,:);
  Eners(i,:)=mca.*rho1(i,:)+mcb.*rhou1(i,:)+mcc.*Ener1(i,:);
end
```

Wir wollen nun wieder anhand des Shu-Osher-Problems zeigen, dass diese Modifikation unseres TVB-Limiters dazu in der Lage ist, dieses Problem in hinreichender Genauigkeit zu lösen und dabei alle Gebiete der Lösung exakt darzustellen. Dafür betrachten wir das Ergebnis des charakteristischen TVB-Limiters bei verschiedenen Werten für M.

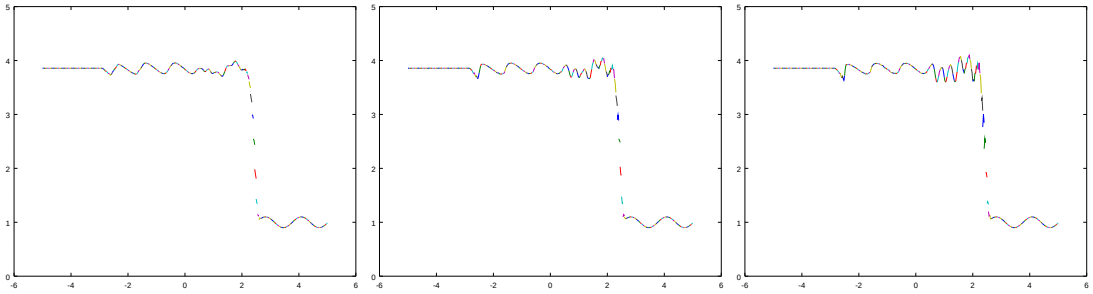


Abb. 4.1: Shu-Osher-Problem, N=2, t = 1.8, K=200; Dichte; v.l.n.r. : TVB in charakteristischen Variablen mit M=10; M=30;M=40

Es ist offensichtlich, dass der Limiter in charakteristischen Variablen mehr Rechenzeit benötigt, da bei jeder Limitierung eine Hin- und eine Rücktransformation vergenommen werden muss. Der Vergleich mit Abbildung 3.7 und insbesondere mit der exakten Lösung in Abbildung 3.6 zeigt jedoch die enormen Vorzüge dieses Limiters. So sehen wir, dass bereits die Abbildung der Oszillation im Bereich von negativen x auch bei niedrigem M deutlich näher an der exakten Lösung ist. Genauso sehen wir ein qualitativ gänzlich anderes Bild im Intervall [0,2], was dem der exakten Lösung sehr ähnlich ist. Am entscheidensten ist jedoch der Vergleich der beiden Fälle mit M=30 bei Abbildung 4.1 und Abbildung 3.7: die unphysikalischen, numerischen Oszillationen haben dramatisch abgenommen, die Lösung ist deutlich glatter geworden. Bei M=40 können wir mit unserem verbesserten Limiter sogar die einzelnen Peaks der exakten Lösung erkennen. Dies demonstriert die Überlegenheit des Limitings in charakteristischen Variablen.

4.2 Positivitätslimiter

Wir wollen nun einen völlig anderen Limiter betrachten, der speziell auf unsere Eulergleichungen angepasst ist. Wie bereits erwähnt ist es notwendig, dass Druck und Dichte zu jedem Zeitpunkt positiv sind. Dies ist nicht nur aufgrund der physikalischen Konsistenz von Nöten. Auch beim Blick in den Code von Kapitel 3.4 zeigt sich, dass bei negativem Druck im nächsten Rechenschritt komplexe Werte auftreten, was das Verfahren instabil werden lässt. Unser Limiter soll nun dafür sorgen, dass im $n+1$-Rechenschritt positive Werte auftreten, wenn die Werte im n-ten Schritt positiv waren.
Diese Leistung kann in vielen Fällen auch unser bisheriger TVB-Limiter übernehmen. Da unsere physikalische Lösung ja bereits positiv ist und dieser sicherstellt, dass wir auch diese berechnen und Oszillationen dämpft, welche in den negativen Bereich eindringen können, reicht er rein theoretisch aus. Durch einen Positivitätslimiter können wir jedoch M viel größer wählen und damit eine höhere Genauigkeit erreichen. Darüber hinaus können wir auch bei Fällen ohne Schocks sicherstellen, dass durch numerische Fehler keine negativen Werte angenommen werden.
Ein solcher Positivtätslimiter wurde in [28] vorgeschlagen. Wir wollen seine Herleitung vorstellen und dann die Implementierung betrachten.
Es sei $\boldsymbol{u} = (\rho, m, E)^T$ unser Vektor der betrachteten Größen. Die Druckfunktion hat die Gestalt $p(\boldsymbol{u}) = (\gamma - 1)(E - \frac{m^2}{2\rho})$. Die Hesse-Matrix dieser Funktion ist für positiven Druck negativ semidefinit. Damit handelt es sich um eine konkave Funktion. Wir können damit für $s \in [0, 1]$ abschätzen:

$$p(s\boldsymbol{u}_1 + (1-s)\boldsymbol{u}_2) \geq sp(\boldsymbol{u}_1) + (1-s)p(\boldsymbol{u}_2)$$

Die Menge an akzeptablen Werten Λ, d.h. solchen, bei denen Druck und Dichte positiv sind, ist somit konvex. Wir suchen ein Verfahren, das stets in Λ abbildet.
Gehen wir von einem Finite-Volumen-Verfahren erster Ordnung der Form

$$\boldsymbol{u}_j^{n+1} = \boldsymbol{u}_j^n - \lambda[f^*(\boldsymbol{u}_j^n, \boldsymbol{u}_{j+1}^n) - f^*(\boldsymbol{u}_{j-1}^n, \boldsymbol{u}_j^n)]$$

aus, bei dem $\boldsymbol{u}$ die approximierten Zellmittelwerte sind, n unsere Zeit- und j unsere Raumschritte zählt. λ ist das Verhältnis der Schrittweite von Zeit zu Raum. Das Verfahren und sein Flux f^* ist nun als positivitätserhaltend zu bezeichnen, wenn aus $\boldsymbol{u}_j^n \in \Lambda$ auch $\boldsymbol{u}_j^{n+1} \in \Lambda$ folgt. Dies gelte unter der Standard-CFL-Bedingung

$$\lambda||(|u| + c)||_\infty \leq \alpha_0 \tag{4.1}$$

wobei u unsere Gasgeschwindigkeit ist, die durch $\frac{m}{\rho}$ berechnet wird und c die Schallgeschwindigkeit des Gases. Die Summe von beiden ist der größte Eigenwert der Jacobi-Matrix des Systems. α_0 ist ein frei wählbare Parameter des Lax-Friedrichs-Flux. Im Anhang von [20] wird gezeigt, dass unser Lax-Friedrichs-Flux ein positivitätserhaltender Flux ist, sofern die CFL-Bedingung wie in unserem Fall gewählt ist. Wir betrachten nun allgemeines Finite-Volumen-Schema der MIttelwerte mit einem Eulerverfahren erster Ordnung in der Zeit:

$$\bar{\boldsymbol{u}}_j^{n+1} = \bar{\boldsymbol{u}}_j^n - \lambda[f^*(\boldsymbol{u}_{j+\frac{1}{2}}^-, \boldsymbol{u}_{j+\frac{1}{2}}^+) - f^*(\boldsymbol{u}_{j-\frac{1}{2}}^-, \boldsymbol{u}_{j-\frac{1}{2}}^+)] \tag{4.2}$$

$\bar{\boldsymbol{u}}_j^n$ ist hier der Zellmittelwert unserer exakten Lösung in der Zelle. Größen mit halbzahligem Index sind die Näherungen der Werte an den Grenzen der Zellen (in unserem Fall der erste und letzte LGL-Quadraturpunkt). [20] zeigt auch (in Theorem 1), dass die Positivität der Grenzpunkte unserer Zellen ausreicht, um bei diesem globale Positivität der Lösung für Druck und Dichte zu erhalten, wenn bei der verfeinerten CFL-Bedingung

$$\lambda||(|u|+c)||_\infty \leq a\alpha_0$$

α_0 so gewählt wurde, dass $\bar{u}_j^{n+1} - a(u^-_{j+\frac{1}{2}} + u^+_{j-\frac{1}{2}})$ ebenso positiv ist. a liegt dabei konstant zwischen 0 und 1. Außerdem werden Verfahren zweiter und dritter Ordnung darauf aufbauend konstruiert und gezeigt, dass dies nicht nur für Euler-Verfahren erster Ordnung gilt, sondern sich ebenso auf SSP-RK-Verfahren erweitern lässt. In [28] wird jedoch ein Vorgehen erarbeitet, dass für beliebige Ordnungen ohne weitere Änderungen funktioniert, was natürlich ein deutlicher Vorzug ist.
Sei nun $\boldsymbol{w}_j(x)$ unser polynomialer Vektor der Ordnung k unserer Lösung, sodass $\bar{\boldsymbol{u}}_j^n$ sein Zellmittelwert ist und für den Zellgrenzwert gilt: $\boldsymbol{u}^+_{j-\frac{1}{2}} = \boldsymbol{w}_j(x_{j-\frac{1}{2}}) = \boldsymbol{w}_j(\hat{x}_j^1)$. Selbiges auch für die andere Zellgrenze; $\hat{x}_j^i$ ist der i-te LGL-Quadraturpunkt.
Man kann nun zeigen, dass, wenn Gleichung 4.2 von den Zellmittelwerten unserer DG-Methode erfüllt wird und wenn $\boldsymbol{w}_j(\hat{x}_j^\alpha)$ stets in Λ liegt, dann gilt auch $\bar{\boldsymbol{u}}_j^{n+1} \in \Lambda$, wenn die CFL-Bedingung

$$\lambda||(|u|+c)||_\infty \leq \hat{\omega}_1\alpha_0 \tag{4.3}$$

erfüllt wird mit ω_1, dem ersten LGL-Koeffizienten (Theorem 2.1 von [28]). Dies wird bewiesen unter Verwendung der Exaktheit der Quadratur und der Zurückführung auf den obigen Fall. Das Ergebnis gilt, wie schon beim TVD-Limiter, auch für SSP-RK-Verfahren aufgrund der Konvexität von Λ.
In unserer CFL-Bedingung Gleichung 4.1 ist noch α_0 näher zu bestimmen, welches wir dann auch für den verfeinerten Fall verwenden können. Aus dem Lax-Friedrichs-Schema folgt, dass sich ergibt:

$$\boldsymbol{u}_j^{n+1} = (1-\lambda a_0)\boldsymbol{u}_j^n + \frac{\lambda a_0}{2}[\boldsymbol{u}_{j+1}^n - \frac{1}{a_0}\boldsymbol{f}(\boldsymbol{u}_{j+1}^n)] + \frac{\lambda a_0}{2}[\boldsymbol{u}_{j-1}^n - \frac{1}{a_0}\boldsymbol{f}(\boldsymbol{u}_{j-1}^n)]$$

Es gilt hierbei $a_0 = ||(|u|+c)||_\infty$. Naheliegende Wahl ist $\lambda a_0 \leq 1$. Dies wollen wir zeigen. Wir sehen, dass $\boldsymbol{u}_j^{n+1}$ eine konvexe Kombination aus drei Vektoren ist, von denen nur die hinteren beiden in Λ liegen müssen, damit es auch $\boldsymbol{u}_j^{n+1}$ tut - der vorderste stammt ja noch aus dem Zeitschritt, in dem nur erlaubte Werte angenommen wurden. Aus der mit $\frac{1}{a_0}$ modifizierten Flussgleichung für rho folgt direkt, dass die Dichte positiv bleibt. Kritisch ist jedoch der Druck, der aus mehreren Größen berechnet wird. Durch direktes Nachrechnen kann jedoch gezeigt werden, dass alle drei Vektoren im erlaubten Bereich liegen, und somit reicht es, zukünftig α_0 auf 1 zu setzen.
Zurück zu unserem Limiter. An diesen bzw. den durch ihn modifizierten Polynomen $\tilde{\boldsymbol{w}}_j(x)$ haben wir drei Anforderungen:

1. Bei glatten Lösungen soll gelten

$$||\tilde{\boldsymbol{w}}_j(x) - \boldsymbol{w}_j(x)|| = \mathcal{O}(\Delta x^{k+1}) \quad \forall x \in I_j$$

2. $\tilde{\boldsymbol{w}}_j(x_j^\alpha) \in \Lambda \quad \text{für} \quad \alpha = 1, 2, \ldots, N$

3. $\frac{1}{\Delta x} \int_{I_j} \tilde{\boldsymbol{w}}_j(x) dx = \bar{u}_j^n$

Wir definieren noch den Druckmittelwert, der sich einfach über die Mittelwerte der anderen Größen ergibt: $\bar{p}_j^n = (\gamma - 1)(\bar{E}_j^n - \frac{1}{2}(\bar{m}_j^n)^2/\bar{\rho}_j^n)$. Wir gehen davon aus, dass es einen Zeitpunkt n gibt, für den gilt: Es gibt ein $\epsilon \geq 0$ mit $\bar{\rho}_j^n \geq \epsilon$ und $\bar{p}_j^n \geq \epsilon$.
Im ersten Schritt wird die Dichte limitert. Hierzu ersetzt man $\rho_j(x)$ durch

$$\hat{\rho}_j(x) = \theta_1(\rho_j(x) - \bar{\rho}_j^n) + \bar{\rho}_j^n, \tag{4.4}$$

wobei gilt $\theta_1 = \min\left\{\frac{\bar{\rho}_j^n - \epsilon}{\bar{\rho}_j^n - \rho_{min}}, 1\right\}$ sowie $\rho_{min} = \min_\alpha \rho_j(\hat{x}_j^\alpha)$, also das Zellminimum der Dichte. Die Mittelwertbildung ergibt selbstverständlich noch denselben Mittelwert, aber es ist nun sichergestellt, dass $\hat{\rho}_j(\hat{x}_j^\alpha) \geq \epsilon$ an jedem Punkt innerhalb unserer Zelle. Die Mittelwerterhaltung lässt sich - neben direktem Nachrechnen - dadurch einsehen, dass quasi Dichte von anderen Punkten innerhalb der Zelle auf die Punkte verteilt wird, wo die Dichte negativ ist, sodass die Gesamtdichte erhalten bleibt.
Als letzte Anforderung fehlt noch die Genauigkeit. Der Beweis dieser ist im Grunde identisch zum Beweis von Lemma 2.4 in [27][1]. Durch Umformungen der Differenz zwischen limitierter und unlimitierter approximierter Lösung erhalten wir das Produkt $(\epsilon - \rho_{min})\frac{\bar{\rho}_j^n - \rho_j(x)}{\bar{\rho}_j^n - \rho_{min}}$. Die Definition von θ ergibt, dass $\rho_{min} \leq \epsilon$ gilt (ansonsten wird kein Limiting vorgenommen, da θ_1=1 gilt und dies ρ_j unverändert lässt), und wir wissen, dass $\epsilon - \rho_{min} = \mathcal{O}(\Delta x^{k+1})$, da ja $\rho_j(x)$ eine Approximation genau dieser Fehlergüte ist. Wir müssen nun nur noch zeigen, dass der zweite Term beschränkt ist. In [27] wird dies über die Äquivalenz von Normen in unserem Raum gezeigt. Damit ist die Genauigkeit dieses Teils des Limiters gezeigt.
Nun muss auch die Positivität des Drucks gewährleistet werden.
Es sei $\hat{\boldsymbol{w}}_j(x) = (\hat{\rho}_j(x), m_j(x), E_j(x))^T$ und $\hat{\boldsymbol{w}}_j^\alpha = \hat{\boldsymbol{w}}_j(\hat{x}_j^\alpha)$. Wir definieren:

$$\Lambda^\epsilon = \left\{ \boldsymbol{u} = \begin{pmatrix} \rho \\ m \\ E \end{pmatrix} \middle| \rho \geq \epsilon \quad \text{und} \quad p = (\gamma - 1)(E - \frac{1}{2}\frac{m^2}{\rho}) \geq \epsilon \right\} \tag{4.5}$$

$$\partial\Lambda^\epsilon = \left\{ \boldsymbol{u} = \begin{pmatrix} \rho \\ m \\ E \end{pmatrix} \middle| \rho \geq \epsilon \quad \text{und} \quad p = (\gamma - 1)(E - \frac{1}{2}\frac{m^2}{\rho}) = \epsilon \right\} \tag{4.6}$$

sowie

$$s^\alpha(t) = (1 - t)\bar{\boldsymbol{u}}_j^n + t\hat{\boldsymbol{w}}_j(\hat{x}_j^\alpha), \quad 0 \leq t \leq 1 \tag{4.7}$$

Die Konvexität von Λ^ϵ ist bereits über die Konvexität unseres Drucks gegeben, $\partial\Lambda^\epsilon$ ist ein Teil dessen Oberfläche. $s^\alpha(t)$ ist eine Gerade durch zwei Punkte. Liegt nun $\hat{\boldsymbol{w}}_j^\alpha$

[1] Man beachte, dass in diesem ein Fehler enthalten ist. Es muss heißen: "By the definition of θ in (2.8), $\theta = \left|\frac{M - \bar{u}_j^n}{M_j - \bar{u}_j^n}\right|$ implies that $\theta = \left|\frac{M - \bar{u}_j^n}{M_j - \bar{u}_j^n}\right| \leq 1$"

außerhalb von Λ^ϵ, d.h. gilt $p(\hat{\boldsymbol{w}}_j^\alpha) \leq \epsilon$, so gibt es einen Schnittpunkt von $s^\alpha(t)$ mit $\partial\Lambda^\epsilon$, den wir mit s_ϵ^α bezeichnen wollen, und mit t_ϵ^α jenes t, das in $s^\alpha(t)$ eingesetzt jenen Punkt ergibt, an dem $p(s^\alpha(t_\epsilon^\alpha)) = \epsilon$ gilt. Wir spielen etwas mit der Notation und definieren:

$$s_\epsilon^\alpha = \begin{cases} s^\alpha(t_\epsilon^\alpha), & \text{falls} \quad p(\hat{\boldsymbol{w}}_j^\alpha) \leq \epsilon \\ \hat{\boldsymbol{w}}_j^\alpha, & \text{falls} \quad p(\hat{\boldsymbol{w}}_j^\alpha) \geq \epsilon \end{cases} \tag{4.8}$$

Wir erhalten so einen neuen Vektor von Polynomen

$$\tilde{\boldsymbol{w}}_j(x) = \theta_2(\hat{\boldsymbol{w}}_j(x) - \bar{\boldsymbol{u}}_j^n) + \bar{\boldsymbol{u}}_j^n \tag{4.9}$$

mit

$$\theta_2 = \min_{\alpha=1,2,\ldots,N} \frac{||s_\epsilon^\alpha - \bar{\boldsymbol{u}}_j^n||}{||\hat{\boldsymbol{w}}_j^\alpha - \bar{\boldsymbol{u}}_j^n||}$$

Dies erhält natürlich wieder den Mittelwert. Wir wollen auch zeigen, dass $\tilde{\boldsymbol{w}}_j(x) \in \Lambda^\epsilon$ gilt an jedem der LGL-Quadraturpunkte. Dafür sehen wir zuerst, dass $\tilde{\boldsymbol{w}}_j(x)$ eine konvexe Kombination von $\hat{\boldsymbol{w}}_{\boldsymbol{j}}(\boldsymbol{x})$ und $\bar{\boldsymbol{u}}_j^n$ ist, und somit ist die Dichte nicht geringer als ϵ. Aus dem gleichen Grund ist auch der Druck zu $\tilde{\boldsymbol{w}}_j$ größer ϵ, wenn es auch der Druck zu $\hat{\boldsymbol{w}}_j$ ist.
Gilt nun aber, dass $p(\bar{\boldsymbol{u}}_j(\hat{x}_j^\alpha)) \leq \epsilon$, so folgt $p(s_\epsilon^\alpha) = \epsilon$ sowie $\tilde{\boldsymbol{w}}_j(\hat{x}_j^\alpha) = \frac{\theta_2}{t_\epsilon^\alpha} s_\epsilon^\alpha + (1 - \frac{\theta_2}{t_\epsilon^\alpha})\bar{\boldsymbol{u}}_j^n$ (direktes Mit Gleichung 4.7 gilt $t_\epsilon^\alpha = \frac{||s_\epsilon^\alpha - \bar{\boldsymbol{u}}_j^n||}{||\hat{\boldsymbol{w}}_j^\alpha - \bar{\boldsymbol{u}}_j^n||}$, sodass aufgrund der Minimumsbildung folgt $\theta_2 \leq t_\epsilon^\alpha$ Damit ist $\tilde{\boldsymbol{w}}_j(x)$ eine konvexe Kombination von $\hat{\boldsymbol{w}}_{\boldsymbol{j}}(\boldsymbol{x})$ und s_ϵ^α und deshalb der Druck größer als ϵ.
Zuletzt wollen wir zeigen, dass unser Limiter die Genauigkeit erhält. Hierzu betrachten wir die Norm $d(\boldsymbol{z}, G) = \min_{\boldsymbol{y}\in\Lambda} ||\boldsymbol{z} - \boldsymbol{y}||$. Wenn nun die exakte Lösung $\boldsymbol{r}(x, t^n)$ glatt ist und $d(\boldsymbol{r}(x, t^n), \Lambda^\epsilon) \geq M, \forall x, M \geq 0$, gilt (das heißt die Lösung liegt außerhalb des erlaubten Bereichs), so reicht zu zeigen dass $\theta_2 = 1 + \mathcal{O}(\Delta x^{k+1})$. Unter unseren Annahmen gilt $\theta_2 = ||s_\epsilon^\beta - \bar{\boldsymbol{u}}_j^n||/||\hat{\boldsymbol{w}}_j^\beta - \bar{\boldsymbol{u}}_j^n|| \leq 1$, wobei β so gewählt ist dass s_ϵ^β der Schnittpunkt mit dem Rand ist.
$\bar{u}_j^n$ ist eine Approximation $(k+1)$-Ordnung des Zellmittelwerts der exakten Lösung $r(x, t^n)$, deshalb gilt $d(\bar{u}_j^n, \Lambda^\epsilon) \geq M + \mathcal{O}(\Delta x^{k+1}) \geq \frac{M}{2}$. Es gilt auch $||s_\epsilon^\beta - \hat{\boldsymbol{w}}_j^\beta|| = \mathcal{O}(\Delta x^{k+1})$, da $\hat{\boldsymbol{w}}_j^\beta$ eine $(k+1)$-Approximation zum Schnittpunkt ist. Damit folgt:

$$1 - \theta_2 = 1 - \frac{||s_\epsilon^\beta - \bar{\boldsymbol{u}}_j^n||}{||\hat{\boldsymbol{w}}_j^\beta - \bar{\boldsymbol{u}}_j^n||} = \frac{||s_\epsilon^\beta - \hat{\boldsymbol{w}}_j^\beta||}{||\hat{\boldsymbol{w}}_j^\beta - \bar{\boldsymbol{u}}_j^n||} \leq \frac{||s_\epsilon^\beta - \hat{\boldsymbol{w}}_j^\beta||}{d(\bar{\boldsymbol{u}}_j^n, \Lambda^\epsilon)} = \mathcal{O}(\Delta x^{k+1})$$

Wir haben ausgenutzt, dass $\hat{\boldsymbol{w}}_\epsilon^\beta, s_\epsilon^\beta, \bar{\boldsymbol{u}}_j^n$ auf einer Linie liegen. Damit ist die Exaktheit gezeigt.
Wir fassen dass Limiting-Verfahren für unsere DG-Methode zusammen:
Zum Zeitpunkt n haben wir das Lösungspolynom $\boldsymbol{w}_j = (\rho_j(x), m_j(x), E_j(x))^T$ des Grades k, und der Zelldurchschnitt sei $\bar{\boldsymbol{u}}_j^n = (\bar{\rho}_j^n, \bar{m}_j^n, \bar{E}_j^n)^T \in \Lambda$, so ist unser Algorithmus für das Eulerverfahren (und auch für das SSP-RK-Verfahren, dann in jedem Teilschritt angewendet):

- Initiiere die kleine Zahl $\epsilon = \min_j\{10^{-13}, \bar{\rho}_j^n, p(\bar{\boldsymbol{u}}_j^n)\}$
- In jeder Zelle wird zuerst die Dichte abgeändert. Wir suchen in jeder Zelle das Minimum der Dichte über $\min_{\alpha=1,\dots,N} \rho_j(\hat{x}_j^\alpha)$. $\hat{\rho}_j(x)$ erhalten wir dann über Gleichung 4.4 und bekommen $\hat{\boldsymbol{w}}_j = (\hat{\rho}_j(x), m_j(x), E_j(x))^T$
- Als nächstes wird der Druck modifiziert. Wenn $p(\hat{\boldsymbol{w}}_j^{\boldsymbol{\alpha}}) \leq \epsilon$, dann lösen wir die folgende quadratische Gleichung

$$p[(1 - t_\epsilon^\alpha)\bar{\boldsymbol{u}}_{\boldsymbol{j}}^{\boldsymbol{n}} + t_\epsilon^\alpha \hat{\boldsymbol{w}}(\hat{x}_j^\alpha)] = \epsilon \tag{4.10}$$

  ansonsten setzten wir t_ϵ^α auf 1. θ_2 ergibt sich dann als $\min_{\alpha=1,\dots,N} t_\epsilon^\alpha$. Wir erhalten somit $\tilde{\boldsymbol{w}}_j(x)$ über Gleichung 4.9

Es ist zu beachten, dass bei unstetigen Situationen der TVD/TVB-Limiter zuerst ausgeführt werden muss. Der TVB-Limiter profitiert vom Positivitätslimiter insofern, als dass ein deutlich größeres M gewählt werden kann, was die Genauigkeit der Lösung deutlich verbessert.
Eine Implementierung des Limiters sieht folgendermaßen aus:

```
function [rhol,rhoul,Enerl]=PLimit(rho,rhou,Ener)
Globals1D;
gamma=1.4;

rhoh =invV*rho; rhoh(2:Np,:)=0; rhoavg =V*rhoh; vrho =rhoavg(1,:);
rhouh =invV*rhou; rhouh(2:Np,:)=0; rhouavg =V*rhouh; vrhou =rhouavg(1,:);
Enerh =invV*Ener; Enerh(2:Np,:)=0; Eneravg =V*Enerh; vEner =Eneravg(1,:);
rhol1=zeros(Np,K);


eps=min([10^-13,min(vrho),min((gamma-1.0)*(vEner-0.5*(vrhou.^2)./vrho))]);

for j=1:K

 minrho=10^20;
 for alph=1:Np
    if rho(alph,j)<minrho
      minrho=rho(alph,j);
    end
  end
    theta1=min((vrho(j)-eps),(vrho(j)-minrho));
    rhol1(:,j)=(theta1*(rho(:,j)-vrho(j)))./(vrho(j)-minrho+10^-13)+vrho(j);
end

p=(gamma-1.0)*(Ener -0.5*((rhou).^2)./rhol1);

tf=1;
for j=1:K

```

```
30   for alph=1:Np
31     if(p(alph,j)<eps)
32
33             diffrho =rhol1(alph,j)-vrho(j);
34             diffrhou=rhou(alph,j)-vrhou(j);
35             diffEner =Ener(alph,j)-vEner(j);
36             eta =2.0*diffrho*diffEner -diffrhou^2;
37             beta =2.0*diffrho*(vEner(j) -eps/(gamma-1.0))+
                   2.0*vrho(j)*diffEner- 2.0*vrhou(j)*diffrhou;
38             delta =2.0*vrho(j)*vEner(j)- vrhou(j)^2-
                   2.0*eps*vrho(j)/(gamma-1.0);
39             beta=beta/eta ;
40             delta=delta/eta;
41             sr =sqrt(abs(beta^2 -4.0*delta) );
42             t1 =0.5*(-b1 -sr);
43             t2 =0.5*(-b1 +sr);
44
45        if(1*10^-14<t1&&t1<1+1*10^-14)
46          ts=t1;
47        else
48          ts=t2;
49       endif
50     else
51       ts=1;
52     endif
53
54     if (ts<tf)
55       tf=ts;
56     endif
57
58   end
59
60   for alpha=1:Np
61     rhol(alpha,j)=tf*(rhol1(alph,j)-vrho(j))+vrho(j);
62     rhoul(alpha,j)=tf*(rhou(alph,j)-vrhou(j))+vrhou(j);
63     Enerl(alpha,j)=tf*(Ener(alph,j)-vEner(j))+vEner(j);
64
65   end
66 end
67
68 return;
```

Im Vergleich zu ***SlopeLimitN*** wird auch hier, wie bereits beim Limiter in charakteristischen Variablen, in allen Größen gleichzeitig limitiert. Im ersten Schritt werden wieder die Mittelwerte berechnet. Daraufhin definieren wir ein ϵ, welches fortan die untere Grenze für unseren Druck und die Dichte darstellt.
Anschließend nehmen wir elementeweise das Dichtelimiting vor, wie im vorherigen Abschnitt dargestellt. Hierbei ist von entscheidender Bedeutung, dass keine Rundungsfehler

auftreten. Da wir ja sehr niedrige Werte erwarten, wenn wir einen Positivitätslimiter nutzen, ist es sehr wahrscheinlich, dass bei der Berechnung von θ_1 solch ein Fehler auftreten könnte. Dies könnte zu einem Scheitern des Limitings führen. Deshalb vermeiden wir die Division bei θ_1 selbst und führen sie erst nach der Multiplikation mit der Dichte aus. Mit der so erhaltenen limitierten Dichte wird der Druck limitert. Dazu berechnen wir punktweise den Druck. Hierzu wird geprüft, ob er unter ϵ liegt. Ist dies der Fall, so lösen wir Gleichung 4.10, indem wir beide analytische Lösungen berechnen. Auch hier achten wir darauf, Rundungsfehler zu vermeiden. Wir wählen diejenige aus, die in [0,1] liegt. Diese existiert, da es einen Schnittpunkt der Verbindungsgeraden zwischen den unlimiterten Werten und der Grenzfläche des erlaubten Bereichs geben muss. Wir speichern die gewählte Lösung als **ts** ab. Ist der Druck größer als ϵ, so setzen wir **ts**$= 1$. Wir suchen das Minimum der **ts** und limiteren damit alle unsere Größen. Damit ist der Limitingprozess beendet. Andere Implementierungen des Limiters finden sich beispielsweise bei [4]
Die Leistungsfähigkeit dieses Limiters soll anhand zweier Testprobleme demonstriert werden. Hierbei betrachten wir als erstes wieder unser Plain-Wave-Problem in modifizierter Form auf dem Bereich [0,1]:

$$\rho(x,0) = 1 + 0.99999999999\sin(x) \quad \rho u(x,0) = \rho(x,0) \quad E(x,0) = \frac{1}{\gamma - 1} + \frac{1}{2}\rho$$

Es handelt sich wieder um ein glattes Problem, d.h. wir erwarten Konvergenzraten von $N + 1$. Ohne den Positivitätslimiter lässt sich dieses Problem jedoch nicht lösen. Da die minimale Dichte 1×10^{-11} beträgt, führt selbst ein geringfügiger Fehler zu einer negativen Dichte, was die Simulation zusammenbrechen lässt. Erst durch sehr kleine Raum- und Zeitschritte kann dieses Problem ohne Positivitätslimiter gelöst werden, was natürlich einen enormen Mehraufwand an Rechenkosten erfordert. Durch den Positivitätslimiter wird dieses Problem nicht nur lösbar, es wird darüber hinaus auch die Genauigkeit vollständig erhalten. Um dies zu belegen, betrachten wir die Konvergenzraten in der L^2-Norm gegenüber der exakten Lösung sowohl für dieses Problem mit der Verwendung des Positivitätslimiters, wie auch des eng verwandten Problems

$$\rho(x,0) = 1 + 0.99\sin(x) \quad \rho u(x,0) = \rho(x,0) \quad E(x,0) = \frac{1}{\gamma - 1} + \frac{1}{2}\rho$$

welches ohne jeden Limiter auskommt.

Tab. 4.1: Fehler und Konvergenzraten für unlimiterte low-density-Problem

N	Fehler für K=25	Fehler für K=50	Konvergenzrate
2	1.4469e-004	2.0291e-005	2.8340
3	4.9478e-008	3.6123e-009	3.7756
4	9.1292e-010	3.0515e-011	4.9029

Wir wählen bewusst ein relativ grobes Gitter, da dies zu größeren Fehlern führt und damit der Positivitätslimiter häufiger getriggert wird. Die vorliegenden Daten zeigen beim Vergleich von Tabelle 4.1 und Tabelle 4.2 deutlich, dass der Positivitätslimiter es

Tab. 4.2: Fehler und Konvergenzraten für limiterte low-density-Problem

N	Fehler für K=25	Fehler für K=50	Konvergenzrate
2	1.5487e-004	2.1885e-005	2.8230
3	5.1435e-008	3.8102e-009	3.7548
4	9.1142e-010	3.3164e-011	4.9018

nicht nur ermöglicht, jene Probleme zu lösen, bei denen äußerst kleine Dichten oder Drücke auftreten, sondern gleichzeitig auch die Genauigkeit vollständig erhält. Dies bestätigt die zuvor getroffenen theoretischen Aussagen
Natürlich vergrößert sich durch den Positivitätslimiter die Rechenzeit. Allerdings geschieht dies in größerem Umfang nur dann, wenn das Limiting auch wirklich notwendig wird. Dies ermöglicht es, ohne nennenswerten Verlust an Geschwindigkeit und ohne Verlust der Genauigkeit den Positivtätslimiter zu nutzen, um sich gegen gegebenenfalls auftretende Bereich mit niedriger Dichte oder niedrigem Druck abzusichern.
Auch die qualitative Erhaltung der Genauigkeit der Simulation soll an folgenden Abbildungen demonstriert werden:

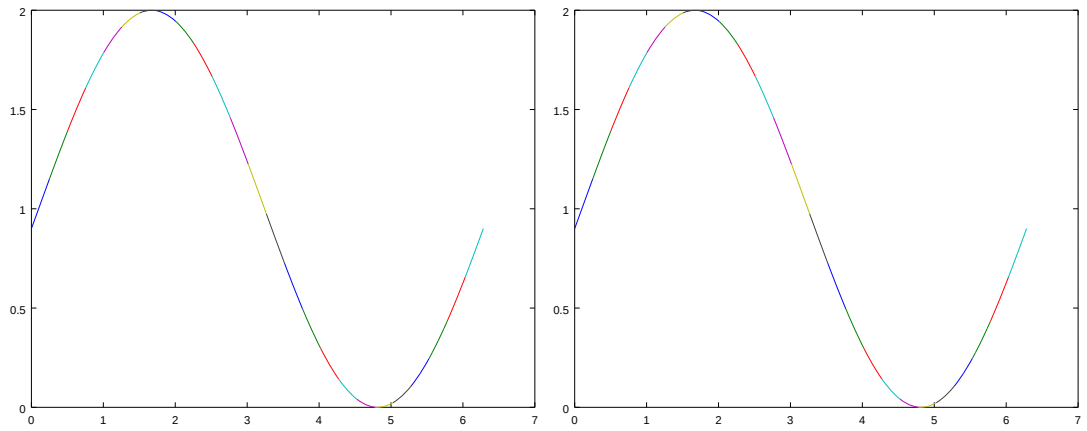


Abb. 4.2: low-density-Problem, N=4, $t = 0.1$, K=50; Dichte; links exakte Lösung, rechts Lösung mit Positivitätslimiter

Als nächstes wollen wir das sogenannte Sedov-Problem betrachten([23]). Dieses ist wie folgt definiert:

$$\rho(x,0) = 1 \quad \rho u(x,0) = 0 \quad E(x,0) = \begin{cases} 3200000 & x = 0 \\ 10^{-12}, & \text{sonst} \end{cases}$$

Bildlich gesprochen handelt es sich also um eine statische, gleichmäßige Dichteverteilung, bei der eine große Menge Energie in den Mittelpunkt injeziert wird. Während das theoretische Problem dies am Nullpunkt tut, ändern wir für die Diskretisierung das Problem so ab, dass auf jeden Punkte in der Zelle, welche den Nullpunkt enthält, die gleiche

Tab. 4.3: Richtwerte für M_{kr} bei N=2 für das Sedov-Problem

K	M_{kr} ohne Positivitätslimiter	M_{kr} ohne Positivitätslimiter
50	500	1100
100	1400	3200
200	4500	12000

Tab. 4.4: Richtwerte für M_{kr} bei K=50 für das Sedov-Problem

N	M_{kr} ohne Positivitätslimiter	M_{kr} ohne Positivitätslimiter
2	500	1000
3	500	700
4	400	800

Menge Energie gegeben wird.
Die Besonderheit des Sedov-Problems liegt darin, dass in den Verdünnungsgebieten hinter den Schocks, welche sich in der Dichteverteilung mit der Zeit bilden, Dichten sehr nahe an 0 entstehen. Auch beim Druck können Oszillationen zu Werten kleiner als 0 führen. Durch diese Eigenheit ist ohne einen Positivitätslimiter ein deutlich strengerer TVB-Limiter, d.h. ein deutlich niedrigerer Wert für M nötig, um Stabilität zu gewährleisten. Dies hat auch Auswirkungen auf die Güte der Lösung.
Wir wollen zeigen, dass, da der Positivitätslimiter die Gefahr eines negativen Drucks oder einer negativen Dichte, wie auch schon zuvor beim glatten low-density-Problem bannt, der TVB-Limiter deutlich sanfter gewählt werden kann. Er ist nur noch insofern nötig, als dass er die Lösung dieses unstetigen Problems ermöglicht. Aufgrund der Unstetigkeit können wir zwar keine deutlich bessere Konvergenz erwarten, jedoch aufgrund des sanfteren TVB-Limiters eine deutliche qualitative Verbesserung der Lösung. Als TVB-Limiter wollen wir unseren Limiter in den charakteristischen Variablen wählen, da sich dieser gegenüber dem klassischen als deutlich überlegen gezeigt hat.
Hierzu betrachten wir für verschiedene Situationen den Wert der Konstanten M (welche ja die Stärke des Limitings regelt), ab der unser Verfahren instabil wird. Richtwerte für den kritischen Wert von M seien in Tabelle 4.3 undTabelle 4.4 aufgeführt.
Wir machen gleich mehrere interessante Beobachtungen. In Tabelle 4.3 stellen wir zuerst fest, dass sich mit der Halbierung der Zellgröße der erlaubte Wert für M grob verdreifacht wird. Sein Ansteigen ist einleuchtend, sorgt doch ein feineres Gitter dafür, dass die Oszillationen deutlich weniger stark ausfallen können. Darüber hinaus sehen wir jedoch auch den großen Vorteil unserer Positivitätslimiters: durch seine Anwendung können wir bei nur geringer Vergrößerung der Rechenzeit den Wert für M mehr als verdoppeln. Der Rechenaufwand ist dabei deutlich niedriger als bei der Verfeinerung des Gitters, da für zusätzliche Zellen ja stets der gesamte Programmzyklus durchlaufen werden muss. Unsere Ergebnisse stimmen auch gut mit [28] überein, wo ja bei N=2 und K=800 ein stabiles Verfahren für M=20000 beobachtet werden konnte,
In Tabelle 3.2 hingegen sehen wir, dass wir den Limiter bei der Verwendung von Polynomen höherer Ordnung strenger wählen müssen. Auch dies entspricht unseren Erwartun-

gen, ist doch eine höhere Polynomordnung mit stärkeren Oszillationen verbunden. Auch hier erreicht unser Positivitätslimiter eine Verdopplung des kritischen Werts für M. Für N=3 ist der Beitrag des Positivitätslimiters deutlich geringer, ein Fall, der an anderer Stelle noch genauerer Untersuchung bedarf (wir hatten ja bereits bei N=3 verschlechterte Konvergenzraten beobachtet, siehe Kapitel 3.5). Ebenso wäre es lohnenswert, zu untersuchen, inwieweit für bestimmte Probleme und Werte für K und N M_{kr} vorhergesagt werden kann, um umfangreiche numerische Tests zu umgehen.
Zuletzt wollen wir noch einen qualitativen, optischen Vergleich der Lösungen nahe an den jeweiligen Werten für M_{kr} unternehmen und damit die Vorzüge der Wahl eines sanften TVB-Limiters und damit auch der Nutzung des Positivitätslimiters unterstreichen.

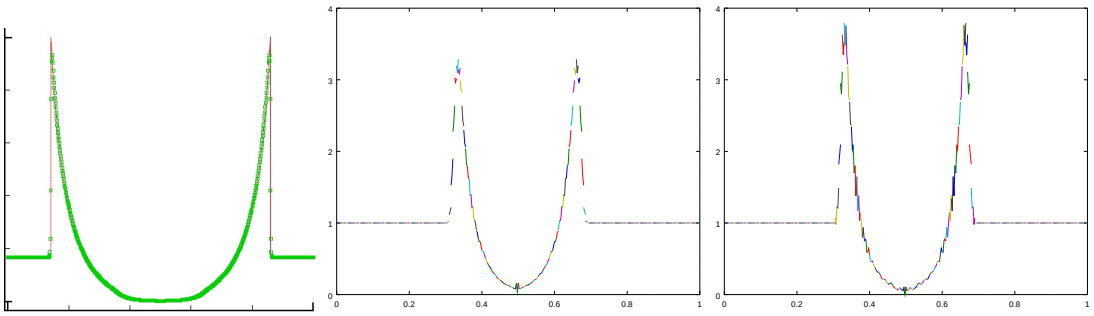


Abb. 4.3: Sedov-Problem, Dichte, N=2, $t = 0.1$, K=200; v.l.n.r. : exakte sowie Lösung mit sehr feinem Gitter aus [28]; ohne Positivitätslimiter bei M_{kr}; mit Positivitätslimiter bei M_{kr}

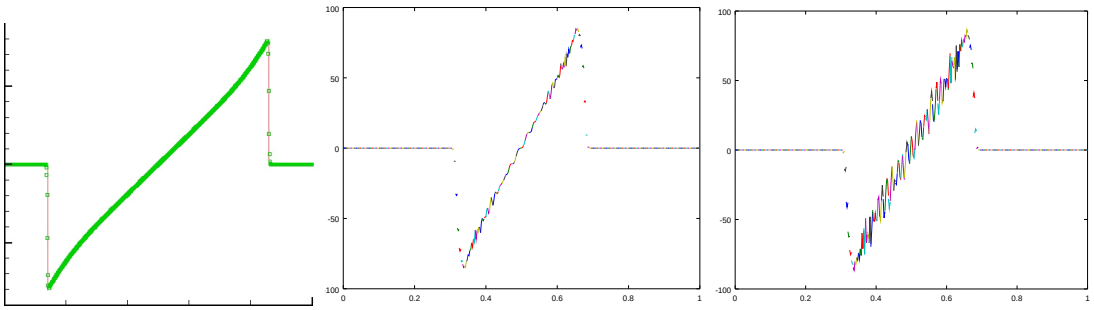


Abb. 4.4: Sedov-Problem, Geschwindigkeit, N=2, $t = 0.1$, K=200; v.l.n.r. : exakte sowie Lösung mit sehr feinem Gitter aus [28]; ohne Positivitätslimiter bei M_{kr}; mit Positivitätslimiter bei M_{kr}

Besonders deutlich wird der Nutzen des Positivitätslimiters bei 4.3. Dort kann die Unstetigkeit viel deutlicher abgebildet werden, die Peaks sind höher, ausgeprägter, dünner und der rechte Winkel wird viel stärker erreicht als bei der lediglich mit TVB-Limiter gerechneten Lösung, die ein niedrigeres M nutzen musste. Auch bei 4.5 erkennen wir eine bessere Auflösung der Unstetigkeit, was sich wieder gut an dem rechten Winkel an der Front der Druckwelle beobachten lässt sowie an der Auflösung der Peaks.

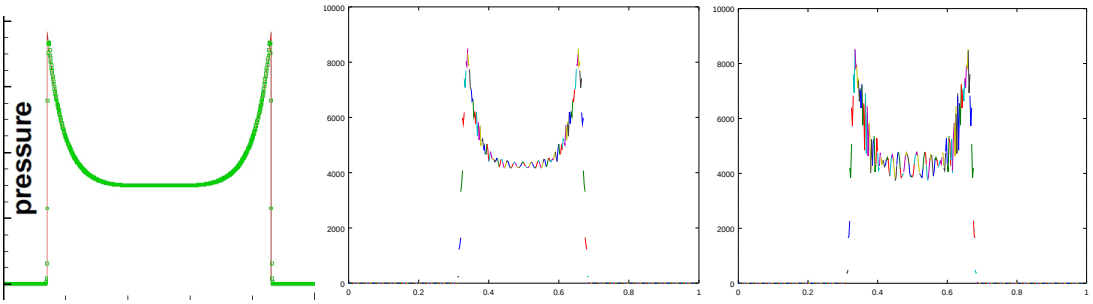


Abb. 4.5: Sedov-Problem, Druck, N=2, $t = 0.1$, K=200; v.l.n.r. : exakte sowie Lösung mit sehr feinem Gitter aus [28]; ohne Positivitätslimiter bei M_{kr}; mit Positivitätslimiter bei M_{kr}

Unser Positivitätslimiter ermöglicht die Nutzung eines sehr sanften TVB-Limiters. Auf die Auswirkungen desselben hat er jedoch keinen Einfluss. Genauso wie die Lösung von den Vorteilen eines schwächeren Limitings profitiert, so leidet sie auch unter seinen Nachteilen. Trotz der Verwendung des Limiters in charakteristischen Variablen werden immer noch Oszillationen eingeführt, die bei hohem M mitunter recht stark ausfallen können. Vor allem bei 4.4 sehen wir die negativen Auswirkungen dieser Oszillationen. Daher ist nicht zwingend sinnvoll, ein größtmögliches stabiles M zu wählen.

5 Schluss

Auch wenn das Sedov-Problem, wie alle verwendeten Beispiele, eine große Idealisierung darstellt, ist seine Bedeutung im Bereich der Physik dennoch äußerst groß. Die punktuelle Freisetzung großer Mengen von Energie erinnern an eine Explosion, und es wurde tatsächlich von Sedov und Taylor im Kontext von Atombombenexplosionen entwickelt. Heutzutage dient es jedoch primär der Beschreibung von Supernovae. Anhand der Vorhersagen durch numerische Berechnungen kann aus den beobachteten Überresten von Supernovae eine Aussage über Sterne getroffen werden, welche sie hervorgerufen haben. Auf diese Weise können Rückschlüsse auf die Entwicklung von Sternsystemen gezogen werden.
Hierzu sind selbstverständlich deutlich umfangreichere Simulationen nötig , als die Demonstrationsversuche in dieser Arbeit. Es ist zum Beispiel nötig, weitere Kraftkomponenten in unser Gleichungssystem einzuführen, wie Gravitation oder elektromagnetische Kräfte, die bei Sternen von großer Bedeutung sind. Dadurch wird unser Schema komplexer und muss modifiziert werden, auch wenn zumindest die Gravitation als Quellterm betrachtet werden kann und dessen Einbindung bereits demonstriert wurde. Dennoch ist auch hier noch umfangreiche Analyse im Gange, beispielsweise bei der Betrachtung von Gleichgewichtszuständen ([18]). Ein weiterer notwendiger, aber umfangreicher Schritt ist es, das Verfahren auf höhere Dimensionen zu erweitern. Hier muss noch deutlich mehr Arbeit in die Konstruktion des Gitters gesteckt werden. Allerdings ist das demonstrierte Verfahren der Gitterinitialisierung so gewählt worden, dass ein Teil davon direkt übernommen werden kann (siehe auch [10]).
Die höhere Komplexität geht natürlich auch mit höheren Berechnungskosten einher. Da diese oft der limitierende Faktor in der Genauigkeit der Simulation sind, ist es unabdingbar, diese zu minimieren. In der Einleitung wurden die Vorzüge der discontinuous Galerkin-Methode bezüglich dieses Faktors erwähnt. Die hier gezeigten Codes sind jedoch nicht optimiert. Sie sollten lediglich der Demonstration der mathematischen Eigenschaften sowie des grundsätzlichen Ansatzes, mitsamt der numerischen Feinheiten, dienen, jedoch nicht der Anwendung in großem Maßstab. So gibt es gleich mehrere Bereiche, in denen sie verbessert werden könnten. Insbesondere die Vorzüge des Positivitätslimiters zahlen sich erst dann wirklich aus, wenn man ihn effizient implementiert - ansonsten kann über feinere Zeit- und Raumschritte mit vergleichbaren Kosten ein ähnliches Ergebnis erreicht werden. Sollte man bei der Sprache Matlab bleiben, so kann eine deutliche Beschleunigung erreicht werden, wenn man stärker von der vektoriellen Struktur Gebrauch macht und von Schleifen Abstand nimmt. Es ist jedoch auch durchaus sinnvoll, über eine Implementierung in effizienteren Sprachen, wie Julia, Python, Fortran oder C++ nachzudenken. Von besonderem Vorzug kann dabei auch eine Parallelisierung der Rechenprozesse sein. Hierzu gibt es verschiedene Ansätze, darunter solche, die beispiels-

weise mit mehreren Gittern arbeiten ([19]). Desweiteren bietet es sich auch an, das Gitter weniger starr zu wählen, als es bei den vorgestellten Simulationen gewählt war. So kann man in Bereichen, die kritisch sind, feinere Gitter wählen, oder in glatten Gebieten höhere Polynomordnungen nutzen, um ein besseres Konvergenzverhalten zu erhalten. Solche Ansätze wurden unter anderen in [22] gewählt.
Auch unsere Limiter spielen erneut eine große Rolle. In dieser Arbeit wurde die Notwendigkeit des Limitings bei unstetigen Lösungen aufgrund des Bedarfs der Reduzierung von Oszillationen bei Unstetigkeiten hergeleitet. Darüber hinaus wurden verschiedene TVD und TVB-Limiter untersucht, welche sicherstellen, dass die totale Variation fällt oder beschränkt bleibt. Ihre Vor- und Nachteile im Kontext von Konvergenz und qualitativer Darstellung der Lösung wurden herausgearbeitet. Dabei konnte gezeigt werden, dass der TVD-Limiter die Konvergenz auf erste Ordnung beschränkt und glatte Extrema schlecht abbildet. Der TVB-Limiter kann diese Nachteile bis zu einem gewissen Grad umgehen, führt jedoch unphysikalische Oszillationen ein. Daraufhin wurde eine Modifikation des TVB-Limiters demonstriert, welche diese Oszillationen über das Limiting in charakteristischen Variablen verringert. Im letzten Schritt wurde, um auch Probleme mit niedrigem Druck oder Dichte lösen zu können, der Positivitätslimiter erarbeitet. Es konnte gezeigt werden, dass er die Behandlung bisher nicht lösbarer Probleme ermöglicht, optimale Konvergenz erhält und bei unstetigen Lösungen eine deutliche Abmilderung des TVB-Limiters ermöglicht.
Der Limiter in charakteristischen Variablen sowie der Positivitätslimiter können auch bei der Simulation einer Supernova verwendet werden. Es wurde ja bereits für den 1D-Fall erläutert, weshalb das Sedov-Problem als unstetiges Problem ohne einen TVB-Limiter nicht lösbar ist, und weshalb die Verwendung eines Limiters in charakteristischen Variablen sowie eines Positivitätslimiters die Lösung deutlich verbessert. Diese Limiter sind ebenfalls so umzubauen, dass sie in höheren Dimensionen arbeiten. [28] zeigt, dass dies für unseren Positivitätslimiter mit nur kleinem Mehraufwand möglich ist.
Die discontiuous Galerkin Methode bietet somit nicht nur zahlreiche Anwendungsgebiete und Vorzüge gegenüber anderen Verfahren, sondern auch reichlich Möglichkeiten zur Optimierung und Erweiterung.

Literaturverzeichnis

[1] Batchelor, G. K.: *An Introduction to Fluid Dynamics.* Cambridge : Cambridge Univ. Pr., 1967

[2] Carpenter, M. H. ; Gottlieb, D. ; Shu, C.-W.: On the Conservation and Convergence to Weak Solutions of Global Schemes. In: *Journal of Scientific Computing* (2003)

[3] Carpenter, M. H. ; Kennedy, C.A.: Fourth-order 2N-storage Runge-Kutta schemes. In: *NASA Report TM 109112* (1994)

[4] Chandrashekar, P.: *Quellcode zu DG-Methoden.* Website: https://bitbucket.org/cpraveen/. – Abgerufen am 14. Juli 2017.

[5] Cockburn, B.: Discontinuous Galerkin method for convection-dominated Problems. In: Barth, T. J. (Hrsg.) ; Deconinck, H. (Hrsg.): *High-Order Methods for Computational Physics.* New York : Springer, 1999

[6] Cockburn, B. ; Shu, C.-W.: Runge–Kutta Discontinuous Galerkin Methods for Convection-Dominated Problems. In: *Journal of Scientific Computing* (2001)

[7] Cockburn, Lin S. ; Shu, C.-W.: TVB Runge-Kutta Local Projection Discontnuous Galerkin Finite Element Method for Conservation Laws III: One-Dimensional Systems. In: *Journal of Computational Physics* (1989)

[8] Gottlieb, S. ; Shu, C.-W. ; Tadmor, E.: Strong stability preserving high order time discretization methods,. In: *SIAM Review* (2001)

[9] Gustavsson, K. ; Saffar Shamshir Gar, D.: *Verifying Numerical Convergence Rates.* Website: http://www.csc.kth.se/utbildning/kth/kurser/DN2255/ndiff13/. 2013. – Abgerufen am 15. Juni 2017.

[10] Hesthaven, J. S. ; Warburton, T.: *Nodal Discontinuous Galerkin Methods.* New York : Springer, 2008

[11] Holden, H. ; Risebro, N. H.: *Front Tracking for Hyperbolic Conservation Laws.* Berlin : Springer, 2015

[12] Jiang, G. S. D. ; Shu, C.-W.: On a cell entropy inequality for discontinuous Galerkin methods. In: *Mathematics of Computation* (1994)

[13] Klessen, R. S. ; Mac Low, M.-M.: Control of star formation by supersonic turbulence. In: *Reviews of Modern Physics* (2004)

[14] Krivodonova, L.: Limiters for high-order discontinuous Galerkin methods. In: *Journal of Computational Physics* (2007)

[15] Landau, L. D. ; Lifschitz, E. M.: *Lehrbuch der theoretischen Physik, Bd. 6, Hydrodynamik*. Haan-Gruiten : Harri Deutsch, 1991

[16] Lax, P.D.: Shock waves and entropy. In: *Proceeding of the Symposium at the University of Wisconsin* (1971)

[17] LeSaint, P. ; Raviart, P.A.: On a Finite Element Method for Solving the Neutron Transport Equation. In: *Mathematical Aspects of Finite Elements in Partial Differential Equations* (1974)

[18] Li, G. ; Xing, Y. J.: Well-Balanced Discontinuous Galerkin Methods for the Euler Equations Under Gravitational Fields. In: *Journal of Scientific Computing* (2016)

[19] Nastase, C. ; Mavriplis, D.: A Parallel hp-Multigrid Solver for Three-Dimensional Discontinuous Galerkin Discretizations of the Euler Equations. In: *45th AIAA Aerospace Sciences Meeting and Exhibit, Aerospace Sciences Meetings*, 2007

[20] Perthame, B. ; Shu, C.-W.: On positivity preserving finite volume schemes for Euler equations. In: *Numerische Mathematik* (1996)

[21] Reed, W. H. ; Hill, T.R.: Triangular mesh methods for the neutron transport equation. In: *Los Alamos Report LA-UR-73-479* (1973)

[22] Schaal, K. ; Bauer, A. ; Chandrashekar, P. ; Pakmor, R. ; Klingenberg, C. ; Springel, V.: Astrophysical hydrodynamics with a high-order discontinuous Galerkin scheme and adaptive mesh refinement. In: *MNRAS* (2015)

[23] Sedov, L.I.: *Similarity and Dimensional Methods in Mechanics*. New York : Academic Press, 1959

[24] Shu, C.-W. ; Osher, S.: Efficient Implementation of Essentially Non-Oscillatory Shock-Capturing Schemes, II. In: *Journal of Computational Physics* (1989)

[25] Smoller, J.: *Shock Waves and Reaction-Diffusion Equations*. New York : Springer, 1983

[26] Thomas, J. W.: *Numerical Partial Differential Equations: Conservation Laws and Elliptic Equations*. New York : Springer, 1999

[27] Zhang, X. ; Shu, C.-W.: On maximum-principle-satisfying high order schemes for scalar conservation laws. In: *Journal of Computational Physics* (2010)

[28] Zhang, X. ; Shu, C.-W.: On positivity preserving high order discontinuous Galerkin schemes for compressible Euler equations on rectangular meshes. In: *Journal of Computational Physics* (2010)

Tabellenverzeichnis

Abbildungsverzeichnis

Erklärung

Hiermit versichere ich die vorliegende Abschlussarbeit selbstständig verfasst zu haben, keine anderen als die angegebenen Quellen und Hilfsmittel benutzt zu haben, und die Arbeit bisher oder gleichzeitig keiner anderen Prüfungsbehörde unter Erlangung eines akademischen Grades vorgelegt zu haben.

Würzburg, den 22. Juli 2017

. . . . . .
Niklas Götz